02 a 05
de outubro 2017
Uberlândia - MG



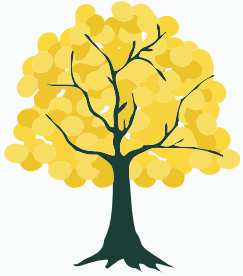
SBBD

Proceedings

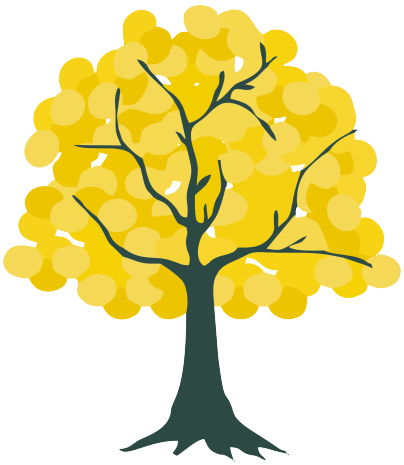
32nd Brazilian Symposium on Databases

Carmem S. Hara, Bernadette F. Lóscio,
Damires Y. S. Fernandes, Ana Carolina B. Salgado,
Humberto L. Razente, Maria Camila N. Barioni (Org.)

Sociedade Brasileira de Computação



SBBD2017



32nd BRAZILIAN SYMPOSIUM ON DATABASES
October 2nd to 5th, 2017
Uberlândia – MG – Brazil

PROCEEDINGS

Promotion
Sociedade Brasileira de Computação – SBC
Comissão Especial de Banco de Dados (CEBD) da SBC

Organization
Universidade Federal de Uberlândia – UFU

Steering Committee Chair
Javam de Castro Machado, UFC

Local Organization Chairs
Maria Camila Nardini Barioni, UFU
Humberto Luiz Razente, UFU

Program Committee Chairs
Carmem S. Hara, UFPR
Bernadette F. Lóscio, UFPE
Damires Y. S. Fernandes IFPB

ISSN: 2316-5170

Dados Internacionais de Catalogação na Publicação (CIP)

B827s Brazilian Symposium on databases (SBBD) (32. : 2017 : Uberlândia, MG, Brazil)
Proceedings [do] 32nd Brazilian Symposium on databases (SBBD), October, 2nd a 5th, 2017, Uberlândia, Minas Gerais; organizadores: Universidade Federal de Uberlândia, Javam de Castro Machado. - Uberlândia: SBC, 2017.

ISSN: 2316-5170
Inclui bibliografia.
341 p.: il.
Modo de acesso: http://www.sbbd.org.br/2017

1. Computação - Congressos. 2. Bases de Dados - Congressos. I. Hara, Carmem Satie. II. Lóscio, Bernadete Farias. III. Brazilian Symposium on databases (SBBD) (32.: 2017 : Uberlândia, MG, Brazil) IV. Universidade Federal de Uberlândia. V. Sociedade Brasileira de Computação. VI. Título.

CDU: 681.3(061.3)

Message from the Local Organization Committee Chairs

Welcome to the 32nd Brazilian Symposium on Databases and to Uberlândia, Minas Gerais! The Brazilian Symposium on Databases is the official database event of the Brazilian Computer Society (SBC) and the largest venue in Latin America for presentation and discussion of research results in the database domain. The 32^{nd} edition of the symposium (SBBD 2017) was held in Uberlândia, in the state of Minas Gerais, from October 2^{nd} to October 5^{th}, 2017. The local organization was performed by the Federal University of Uberlândia (UFU) through the Computing Faculty (FACOM). This year, for the first time, SBBD had the Brazilian Conference on Intelligent Systems (BRACIS) and the Symposium on Knowledge Discovery, Mining and Learning (KDMiLe) as co-located events providing a rich environment for the discussion of researches of their interrelated areas.

The SBBD 2017 program offers a variety of activities, suited for an audience ranging from undergraduate to Ph.D. students, database professionals, practitioners and researchers. The program includes: 3 invited talks and 3 tutorials, presented by distinguished speakers from Brazil, USA and France; 9 technical sessions; 3 short courses about hot topics in the area, presented by specialists in their research fields; demos and applications session; posters sessions; thesis and dissertations workshop; the biannual thesis and dissertations contest; 2 co-located workshops; the 1^{st} KDD-BR (Brazilian Knowledge Discovery in Databases) competition; and a panel.

The excellence of SBBD 2017 program is the result of the competence and effort of a large community, which we gratefully acknowledge. The various sections of these proceedings list in detail those that contributed to the SBBD 2017 edition. We thank the symposium chairs and our colleagues of the local organization committee who donated their precious time to made SBBD 2017 a reality. We also thank the Computing Faculty (FACOM) of the Federal University of Uberlândia (UFU). We are also grateful to the SBC board for their support and to the steering committee members for their help, advice and support. Further, we thank the program committee members and external reviewers for the high quality reviews, and the authors who submitted their papers to SBBD 2017. Finally, we are grateful to our sponsors. Without their support we would not be able to organize this annual event that brings together our community.

We hope you all enjoy SBBD 2017 in Uberlândia, Minas Gerais!

Maria Camila Nardini Barioni, UFU
Humberto Luiz Razente, UFU
SBBD 2017 Local Organization Committee Chairs

Table of Contents

SBBD Proceedings – Full Papers 8

SBBD Proceedings – Short Papers 196

Tutorials 318

Invited Talks 335

Editorial

It is a great pleasure to introduce the Proceedings of the Brazilian Symposium on Databases (SBBD) with the full and short papers accepted for presentation at the 32nd edition of the symposium. SBBD 2017 was held in Uberlândia, in the state of Minas Gerais, Brazil, from October 2nd to October 5th, 2017. It was organized by the Federal University of Uberlândia (UFU), and for the first time, SBBD was held in conjunction with the Brazilian Conference on Intelligent Systems (BRACIS) and the Symposium on Knowledge Discovery, Mining and Learning (KDMiLe).

SBBD is the official database event of the Brazilian Computer Society (SBC) and the largest venue in Latin America for presentation and discussion of research results in the databases domain. Along with technical sessions, SBBD includes invited talks, tutorials and short courses given by distinguished speakers from the national and international research communities. SBBD regularly promotes a demos and applications session, and a thesis and dissertations workshop as co-located events. This year, we also promoted the Thesis and Dissertation Contest, a biannually event started in 2015, in which the best Brazilian thesis and dissertations in the database area, defended in 2015 and 2016, were presented and competed for an award.

We also introduced some new activities to the event. For the first time, there were two co-located workshops: the Dataset Showcase Workshop, and the Databases meet Bioinformatics Workshop. Moreover, SBBD joined BRACIS and KDMile to promote the 1st Brazilian Knowledge Discovery in Databases competition (KDD-BR). The competition involved the classification of real images captured by a monitoring station in an astronomy observatory.

All papers presented in technical sessions during the event reported interesting results or proposed novel thought-provoking ideas in several subjects on the databases and related areas. For the 2017 edition, SBBD accepted five categories of submissions: JIDM articles, full papers, short papers, vision papers, and distinguished published papers.

Submissions to the JIDM category can be made throughout the year. The review process is conducted by the editorial board of JIDM, leaded by the editor-in-chief Caetano Traina Jr. Articles accepted by August 8th were invited to be presented at SBBD 2017. This year, only one JIDM article were presented during the event.

Full papers had two cycles of submissions: the first one with deadline in March and the second one with deadline in May. The review process involved a single review round, with a rebuttal phase. Authors were initially notified with the reviews and had a few days for answering the reviewers' comments during the rebuttal phase. After evaluating the rebuttal comments during the discussion period, a final decision was achieved. Out of 45 papers submitted to both the 1st and 2nd cycles, 15 were accepted as full papers (acceptance rate of 33%), and 2 as short papers.

Short and vision papers had a single cycle of submissions with deadline in July. There were no rebuttal phase. However, all program committee members were invited to participate in a discussion period to reach a decision. There were 38 submissions of short papers, out of which 17 were accepted (acceptance rate of 45%). There were no vision papers submissions.

Distinguished Published Papers is a new category of submissions introduced in SBBD 2017. It aimed to attract the best papers of the Brazilian community, published or accepted for publication by a first-class database conference, and give the authors the opportunity to present their work during the event. There were 2 submissions, but none of them were accepted by the SBBD Steering Committee, on the grounds of adherence to the call for papers.

The topics with more submissions among full and short papers, according to the author's selection from the topics of interest, were: Performance Evaluation and Benchmarking (18 submissions), NoSQL Databases (18 submissions), Data Analytics and Data Visualization (18 submissions), Algorithms and Techniques for Data Mining (14 submissions), Database Techniques to support Data Mining (9 submissions), Database Design and Data Semantics (9 submissions), Data on the Web (9 submissions), Information Retrieval Models and Techniques (8 submissions), and Data Management in Clouds (8 submissions).

JIDM articles, as well as full and short accepted papers were presented in technical sessions during the event. An invited paper has also been included in the technical program: "The collaboration network of the Brazilian Symposium on Databases". It analyzes SBBD's co-authorship network during its 30 years of history. The article has been published in the Journal of the Brazilian Computer Society, in its July 2017 issue.

The best full and short papers received award certificates during the event and will be invited to submit extended versions to JIDM. We have also awarded "Outstanding Reviewer Certificates" to the technical program committee members that excelled in the reviewing process. We do not have enough words to thank all committee members and external reviewers for their commitment and high quality reviews.

The Proceedings of SBBD are the result of the collective effort of a large community, which we gratefully acknowledge. We thank the SBBD 2017 local organization committee and its symposium chairs, who worked hard to guarantee an outstanding event. We are also grateful to the steering committee members for their help, advice and support. Lastly, but most importantly, we are grateful to the authors who submitted their work to SBBD 2017.

Carmem S. Hara, UFPR
SBBD 2017 Program Committee Chair

Bernadette Farias Lóscio, UFPE
Damires Yluska Souza Fernandes, IFPB
SBBD 2017 Short Papers Program Co-Chairs

32th Brazilian Symposium on Databases

full

October 2nd to 5th, 2017
Uberlândia – MG – Brazil

SBBD PROCEEDINGS

FULL PAPERS

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Uberlândia – UFU

Program Chair

Carmem Hara, UFPR

32nd Brazilian Symposium on Databases

October 2nd to 5th, 2017
Uberlândia – MG – Brazil

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Uberlândia – UFU

SBBD Steering Commitee

Agma Juci Machado Traina, USP
Bernadette Lóscio, UFPE
Caetano Traina Jr., USP
Carmem Hara, UFPR
Javam Machado, UFC
Mirella M. Moro, UFMG
Vanessa Braganholo, UFF

SBBD 2017 Commitee

Steering Committee Chair
Javam Machado, UFC

Local Organization Chairs
Maria Camila N. Barioni, UFU and Humberto L. Razente, UFU

Program Committee Chair
Carmem S. Hara, UFPR

Short papers Chairs
Bernadette Lóscio, UFPE and Damires Souza, IFPB

Demos and Applications Session Chair
Daniel de Oliveira, UFF

Short Courses Chair
Vaninha Vieira, UFBA

Workshop on Thesis and Dissertations in Databases Chair
Carina Dorneles, UFSC

Tutorials Chair
Ana Carolina Salgado, UFPE

Thesis and Dissertation Contest Chair
Vânia Vidal, UFC

Workshops Chair
Fernanda Baião (UNIRIO)

Local Organization Committee

Maria Camila N. Barioni, UFU
Humberto L. Razente, UFU
José Gustavo de Souza Paiva, UFU
Marcelo Zanchetta do Nascimento, UFU
Elaine Ribeiro de Faria Paiva, UFU
João Henrique de Souza Pereira, UFU

Full Papers Program Committee

Agma Traina, Universidade de São Paulo (ICMC/USP)
Alberto Laender, Universidade Federal de Minas Gerais (UFMG)
Alexandre Plastino, Universidade Federal Fluminense (UFF)
Altigran Soares da Silva, Universidade Federal do Amazonas (UFAM)
Ana Carolina Salgado, Universidade Federal de Pernambuco (UFPE)
André Santanchè, Universidade Estadual de Campinas (Unicamp)
Angelo Brayner, Universidade Federal do Ceará (UFC)
Bernadette Loscio, Universidade Federal de Pernambuco (UFPE)
Caetano Traina Júnior, Universidade de São Paulo (ICMC/USP)
Carina F. Dorneles, Universidade Federal de Santa Catarina (UFSC)
Carmem Hara, Universidade Federal do Paraná (UFPR), Chair
Celso Hirata, Instituto Tecnológico de Aeronáutica (ITA)
Clodoveu Davis, Universidade Federal de Minas Gerais (UFMG)
Cristina Ciferri, Universidade de São Paulo (ICMC/USP)
Daniel de Oliveira, Universidade Federal Fluminense (UFF)
Daniel Kaster, Universidade Estadual de Londrina (UEL)
Divyakant Agrawal, University of California, Santa Barbara (UCSB)
Duncan Ruiz, Pontifícia Universidade Católica do Rio Grande do Sul (PUC/RS)
Edleno Moura, Universidade Federal do Amazonas (UFAM)
Eduardo Ogasawara, Centro Federal de Educação Tecnológica Celso Suckow da Fon-

seca (CEFET/RJ)
Elaine Sousa, Universidade de São Paulo (USP)
Eli Cortez, Microsoft
Fabio Porto, Laboratório Nacional de Computação Científica (LNCC)
Fernanda Baião, Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Humberto Razente, Universidade Federal de Uberlândia (UFU)
Javam Machado, Universidade Federal do Ceará (UFC)
João Eduardo Ferreira, Universidade de São Paulo (USP)
Jonice Oliveira, Universidade Federal do Rio de Janeiro (DCC/IM/UFRJ)
José Palazzo Moreira de Oliveira, Universidade Federal do Rio Grande do Sul (UFRGS)
José Antonio Macêdo, Universidade Federal do Ceará (UFC)
Karin Becker, Universidade Federal do Rio Grande do Sul (UFRGS)
Marco Antonio Casanova, Pontifícia Universidade Católica do Rio de Janeiro (PUC/Rio)
Marcos Gonçalves, Universidade Federal de Minas Gerais (UFMG)
Maria Camila Nardini Barioni, Universidade Federal de Uberlândia (UFU)
Maria Claudia Cavalcanti, Instituto Militar de Engenharia (IME)
Marta Mattoso, Universidade Federal do Rio de Janeiro (COPPE/UFRJ)
Mirella M. Moro, Universidade Federal de Minas Gerais (UFMG)
Mirian Halfeld-Ferrari, Université d'Orleans (LIFO)
Pedro Eugenio Rocha Pedreira, Facebook Inc.
Renata Galante, Universidade Federal do Rio Grande do Sul (UFRGS)
Renato Fileto, Universidade Federal de Santa Catarina (UFSC)
Ricardo Ciferri, Universidade Federal de São Carlos (UFSCar)
Ricardo Torres, Universidade Estadual de Campinas (Unicamp)
Ronaldo Mello, Universidade Federal de Santa Catarina (UFSC)
Sergio Lifschitz, Pontifícia Universidade Católica do Rio de Janeiro (PUC/Rio)
Valéria C. Times, Universidade Federal de Pernambuco (UFPE)
Vanessa Braganholo, Universidade Federal Fluminense (UFF)
Vania Bogorny, Universidade Federal de Santa Catarina (UFSC)
Vaninha Vieira, Universidade Federal da Bahia (UFBA)
Viviane Moreira, Universidade Federal do Rio Grande do Sul (UFRGS)
Wagner Meira Jr., Universidade Federal de Minas Gerais (UFMG)
Wellington Martins, Universidade Federal de Goiás (UFG)

External Reviewers

Anderson Chaves Carniel (ICMC-USP)
Eduardo Gonçalves (ENCE-IBGE)
Flavio R. C. Sousa (UFC)
Leonardo Moreira (UFC)
Mariana Tasca (CTSMART)
Rebeca Schroeder Freitas (UDESC)
Troy Kohwalter (UFF)

Table of Contents (JIDM Papers)

Querying Provenance along with External Domain Data Using Prolog
Wellington Oliveira (UFF), Kary A. C. S. Ocaña (LNCC), Daniel de Oliveira(UFF) e Vanessa Braganholo (UFF)
https://seer.lcc.ufmg.br/index.php/jidm/article/view/1878

Table of Contents (Invited Papers)

The collaboration network of the Brazilian Symposium on Databases
Journal of the Brazilian Computer Society, to be published, December (2017)
Lucas Henrique Costa de Lima (UFMG), Gustavo Penha (UFMG), Laís Mota de Alencar Rocha (UFMG), Mirella M. Moro (UFMG), Ana Paula Couto da Silva (UFMG), Alberto H. F. Laender (UFMG), José Palazzo M. de Oliveira (UFRGS)
https://doi.org/10.1186/s13173-017-0059-6

Table of Contents (Full Papers)

A Framework for Online Clustering Based on Evolving Semi-Supervision 16
Guilherme Alves (UFU), Maria Camila N. Barioni (UFU), Elaine R. Faria (UFU)

A Spline-based Cost Model for Metric Trees 28
Marcos V. N. Bedo (ICMC-USP, UFF), Agma J. M. Traina (ICMC-USP), Caetano Traina Jr. (ICMC-USP)

Analyzing the Performance of Spatial Indices on Flash Memories using a Flash Simulator 40
Anderson Chaves Carniel (ICMC-USP), Tamires Brito da Silva (ICMC-USP), Kairo Luiz dos Santos Bonicenha (ICMC-USP), Ricardo Rodrigues Ciferri (UFSCar), Cristina Dutra de Aguiar Ciferri (ICMC-USP)

Explorando arquiteturas multi-core para processamento eficiente de consultas em sistemas de gerência de Big Data 52
Frank W. R. da Silva (UFF, UNEMAT), Victor T. de Almeida (UFF, Petrobras), Vanessa Braganholo (UFF)

Extração de Dados de Conferências a Partir da Web 64
Cássio Alan Garcia (UFRGS), Viviane P. Moreira (UFRGS)

Identificação do grafo de genealogia acadêmica de pesquisadores: Uma abordagem baseada na Plataforma Lattes 76
Rafael J. P. Damaceno (UFABC), Luciano Rossi (UFABC), Jesús P. Mena-Chalco (UFABC)

Ligações Semânticas Utilizando Predicados SKOS 88
Ricardo Ávila (UFC), Salomão Santos (UFC), David Araújo (UFC), Vânia Vidal (UFC), José Macêdo (UFC)

ParallelNACluster: Uma estratégia paralela de clusterização para o casamento de múltiplos catálogos 100
Vinícius Pires de Moura Freire (UFC, IFMA), Fabio Porto (LNCC), José A. F. de Macêdo (UFC)

Particionamento como Ação de Sintonia Fina em Bancos de Dados Relacionais 112
Ana Carolina Almeida (UERJ), Antony Seabra de Medeiros (PUC-Rio), Rogério Luís de Carvalho Costa (PUC-Rio), Sergio Lifschitz (PUC-Rio)

Relational graph data management on the edge: Grouping vertices' neighborhood with Edge-k 124
Lucas C. Scabora (ICMC-USP), Paulo H. Oliveira (ICMC-USP), Daniel S. Kaster (UEL), Agma J. M. Traina (ICMC-USP), Caetano Traina Jr. (ICMC-USP)

STACY: Um Novo Algoritmo para Automaticamente Classificar a Força dos Relacionamentos ao Longo dos Anos 136
Michele A. Brandão (UFMG), Pedro O. S. Vaz de Melo (UFMG), Mirella M. Moro (UFMG)

Um Survey sobre Soluções para Publicação de Dados na Web sob a Perspectiva das Boas Práticas do W3C 148
Lairson Emanuel R. de Alencar Oliveira (UFPE), Marcelo Iury S . Oliveira (UFPE, UFRPE), Bernadette Farias Lóscio (UFPE)

Uma Avaliação de Eficiência e Eficácia da Combinação de Técnicas para Deduplicação de Dados 160
Levy de Souza Silva (UFMG), Dimas Cassimiro Nascimento Filho (UFRPE), Mirella M. Moro (UFMG)

Uma Proposta de Perfil de Conjuntos de Dados na Web com Enriquecimento Semântico 172
Natacha Targino (UFPE), Damires Souza (IFPB), Ana Carolina Salgado (UFPE)

Workload-Aware RDF Partitioning and SPARQL Query Caching for Massive RDF Graphs stored in NoSQL Databases 184
Luiz Henrique Zambom Santana (UFSC), Ronaldo dos Santos Mello (UFSC)

paper:171369

A Framework for Online Clustering Based on Evolving Semi-Supervision*

Guilherme Alves[1], Maria Camila N. Barioni[1], Elaine R. Faria[1]

[1]Faculdade de Computação (FACOM/UFU)
Universidade Federal de Uberlândia, MG - Brasil

{guilhermealves,camila.barioni,elaine}@ufu.br

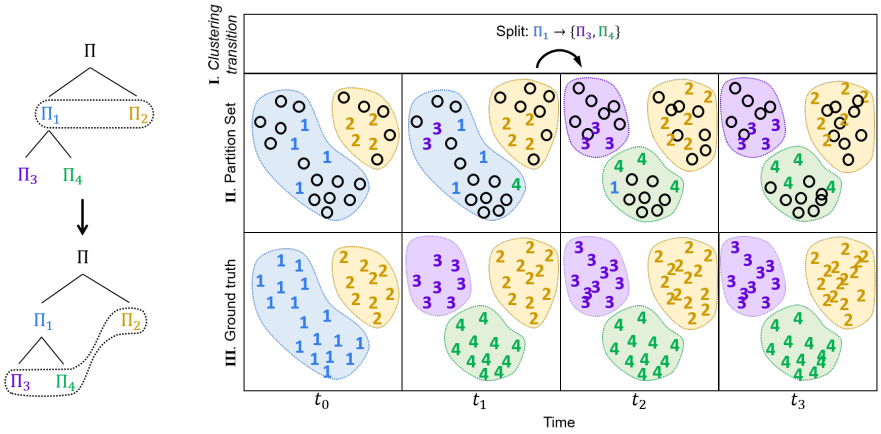
Abstract. *The huge amount of currently available data puts considerable constraints on the task of information retrieval. Automatic methods to organize data, such as clustering, can be used to help with this task allowing timely access. Semi-supervised clustering approaches employ some additional information to guide the clustering performed based on data attributes to a more suitable data partition. However, this extra information may change over time imposing a shift in the manner by which data is organized. In order to help cope with this issue, we propose the framework called* CABESS *(Cluster Adaptation Based on Evolving Semi-Supervision), for online clustering. This framework is able to deal with evolving semi-supervision obtained through user binary feedbacks. To validate our approach, the experiments were run over hierarchical labeled data considering clustering splits over time. The experimental results show the potential of the proposed framework for dealing with evolving semi-supervision. Moreover, they also show that our framework is faster than traditional semi-supervised clustering algorithms using lower standard semi-supervision.*

1. Introduction

The advent of ubiquitous computing is the one of the reasons most responsible for the tremendous increase in data generation. However, notwithstanding the valuable information contained in this huge amount of data, without the support of appropriate approaches the user may end up drowning in the data. Data mining approaches, such as clustering, aim at helping to obtain useful information from large datasets. Clustering approaches have been designed with the purpose of grouping data in order to detect patterns, to summarize information or help in the arrangement thereof [Barioni et al. 2014]. When there is some additional background knowledge available or a subset of labeled data, this additional information may be used to guide the clustering process to a desirable or more suitable data partition. Techniques that employ this additional information, referred to here as constraints, constitute the research area called semi-supervised clustering. Considering that the desired organization for the data may change over time, semi-supervised approaches may be useful for guiding clustering algorithms in the adaptation process.

The motivation for the work described herein is illustrated through the following example. Suppose that a marketing company aims to segment customers into groups, as well as monitor the evolution of these groups over time. In order to find these groups

*This work has been supported by CAPES, CNPq, and FAPEMIG.



(a) Grouping tree. (b) Execution results *vs.* ground truth over time.

Figure 1. An illustrative example. (a) Grouping trees showing the desired data partition highlighted by the dotted region in two different timestamps. (b) The evolution of the clustering structure as new data arrives and the constraint change over time. The numbers represent semi-supervision labels and the black circles represent the data instances without semi-supervision.

there is an available set of characteristics that represent each customer (such as monthly income, age) and additional information concerning the preferences of a small number of customers. Thus, a semi-supervised clustering algorithm is used at first. However, as time goes by, the preferences of the customers may change leading to new clustering structures. That is, at another time the preferences of the customers of a group change and cause the appearance of a new subgroup of costumers. Therefore, constraints, derived from the new preferences of the customers, impose a new clustering structure splitting a previous group into two newer groups. Noteworthy here is that more customers may still appear, but the characteristics do not change, that is, the representation of the customers does not change over time only their preferences.

The previously described scenario is an instance of the problem addressed herein. Figure 1 illustrates the evolution of the clustering structure as the constraints change over time. Figure 1a depicts a grouping tree, describing the different levels of the possible data organization. The desired organization is highlighted by the dotted regions. Thus, one observes that there are two possible types of targeted data organization. On the left-hand side, Figure 1b describes the following sets over time: (I) the cluster transitions detected, (II) the resulted partition set after a semi-supervised clustering, and (III) the ground truth, that is, the optimal partition set desired by a user at each timestamp. By analyzing the clustering results from the timestamp t_0 to t_3, one notes that the user criteria for partitioning the data has changed according to the level of hierarchy desired (Figure 1a). The feedbacks define constraints that guide the clustering process (see numbers as labels in Figure 1b).

Considering that the user constraints may change over time, the following clustering transitions may occur: birth, split or merge. The work described herein is focused on splits and assumes that the evolving semi-supervision can impact the modifications in the clustering structures over time, implying in new splits. Therefore, the main goal of our work is to provide a framework that is able to use and maintain semi-supervision

correctly to enable efficient and effective online clustering processes. The main contributions of the work described herein can be summarized as follows: (1) The introduction of CABESS (Cluster Adaptation Based on Evolving Semi-Supervision) (Section 3), a framework which aims at allowing efficient and effective online clustering using semi-supervision in the form of feedback. (2) The proposal of a strategy that extracts semi-supervision information from feedback given in the form of labels (Section 4.1). (3) An approach to keep the labels consistent over time (Section 4.2).

This paper is organized as follows. Section 2 presents the related works. Section 3 formalizes the problem addressed in this paper and describes our framework CABESS. Section 4 presents Pointwise CABESS, an instantiation of our framework, and its particularities. Section 5 explains the experimental setup and Section 6 presents and discusses the experimental results. The conclusions and the future works are discussed in Section 7.

2. Related Work

The strategies employed in the framework CABESS are related to the research performed on semi-supervised clustering with online learning and tracking clustering. Therefore, Section 2.1 briefly presents the fundamental concepts of batch and online semi-supervised clustering. Following on from this, Section 2.2 describes the related works regarding monitoring evolving clusters.

2.1. Semi-supervised Clustering

The semi-supervised clustering approaches described in the scientific literature provide different approaches in order to allow for the guiding of the clustering process, and the obtainment of meaningful clusters [Barioni et al. 2014]. Generally, traditional semi-supervised clustering algorithms are divided into two categories: similarity-adapting methods that adapt the similarity measure employed in the clustering process in order to satisfy the labels or constraints in the data; and methods that employ labels or constraints provided by the users to modify the clustering assignment step [Basu et al. 2008].

There is also a variety of ways to express and to obtain these constraints. A substantial part of the works described in the scientific literature explores the specification of these restrictions in the form of instance-level constraints [Bilenko et al. 2004], which are composed of two types of constraints: *must-link* (ML) and *cannot-link* (CL). A ML constraint indicates that two data instances must be in the same cluster. A CL constraint implies that two data instances must not be in the same cluster. Other means for expressing constraints are attribute-level constraints [El Moussawi et al. 2016], cluster-level constraints [Dubey et al. 2010], relative constraints [Liu et al. 2011], and labels [Castellano et al. 2013].

Several research works have extended classical clustering methods to be able to deal with additional information. MPCK-Kmeans [Bilenko et al. 2004], for example, is an extension of the widely used K-MEANS algorithm [Jain and Dubes 1988] that incorporates metric learning and semi-supervision in the form of ML and CL. C-DBScan [Ruiz et al. 2007] and SSDBScan [Lelis and Sander 2009] extended the well known DBScan algorithm [Ester et al. 1996]. While the first one deals with pairs of constraints, the latter receives a set of labeled instances as semi-supervision input. These methods require that

the entire dataset and all the semi-supervision information be available at the beginning of the data clustering process.

Examples of online approaches are described in [Castellano et al. 2013] and [Lai et al. 2014]. The first presents an extension of SSFCM for automatic image annotation based on semi-supervised clustering. It assumes that instances belonging to some clusters are available over time and can be clustered as chunks. The authors in [Lai et al. 2014] propose a semi-supervised clustering technique that improves the clustering effectiveness interactively from user feedbacks in the form of *positive*, *negative* and *displacement* feedback. A *positive* feedback means that a data instance was correctly assigned to the current cluster. A *displacement* feedback means that a data instance must not belong to the current cluster and points to the appropriate cluster. Although feedback arrives continually, [Lai et al. 2014] consider that it has all data instances at the beginning of the execution. Furthermore, all of these approaches assume that the user feedback is always coherent among different iterations.

2.2. Tracking and Evolving Clustering

An important research issue that has recently come to be explored is related to the understanding of the clusters behavior over time. The MONIC [Spiliopoulou et al. 2006] and MClusT [Oliveira and Gama 2010] frameworks are some well-known examples that deal with this issue. The first of these two frameworks models and tracks cluster changes in order to understand the nature of the change. The cluster transitions are formalized and an algorithm for detecting transitions is proposed. MClusT builds a bipartite graph for modeling the cluster transitions, where vertexes are clusters and edges represent the relationship between a pair of clusters. Thus, a user may well attain knowledge into what happened with instances of a cluster that split at a particular period. Other approaches have been proposed, such as [Pereira and Moreira 2016], but it is noteworthy that none of these explored the analysis of the evolution of constraints over time.

3. The CABESS Framework

We revisited the semi-supervised clustering model described in [Lai et al. 2014] in order to propose CABESS, a generic framework that aims at using a limited amount of additional information to provide efficient and effective online clustering. It also uses an approach that detects external clustering transitions in order to manage evolving semi-supervision over time. CABESS receives as input a massive sequence of data instances $(x_1, ..., x_n)$, $\mathcal{D} = \{x_i\}_{i=1}^{n}$, and a limited sequence of feedbacks $\mathcal{F} = \{f_i^j\}_{j=1}^{m}$, $|\mathcal{F}| \ll |\mathcal{D}|$. Each instance is described as a d-dimensional feature vector $x_i = [x_i^j]_{j=1}^{d}$ that belongs to a continuous feature space Ω. Each user feedback f_i^j relates to a data instance $x_i \in \mathcal{D}$ and expresses either two types of feedbacks: **positive** and **displacement**.

The clustering process performed by CABESS contains five steps (see Figure 2). Considering that no feedback information is available at first, the clustering process starts summarizing the dataset using a micro-clustering approach based on CF-Vectors (step 1). Among the approaches that can be used in these situations, one finds BIRCH [Zhang et al. 1996] and CluStream [Aggarwal et al. 2003]. A CF-Vector summarizes data of a group of N instances as a triplet $CF = (N, \overrightarrow{LS}, SS)$ where $\overrightarrow{LS}$ is the linear sum of the data instances and SS is the sum of the squared data instances. CF-vectors have important

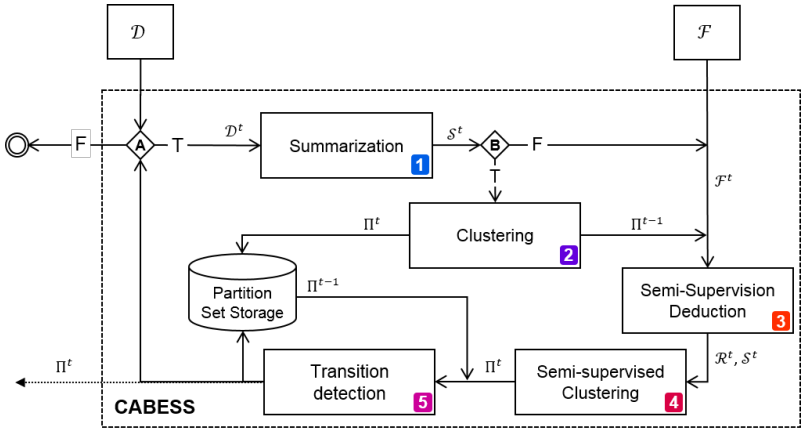


Figure 2. The framework CABESS. Test A verifies if new instances have been generated or the user has been satisfied with the cluster quality (feedback generation). Test B verifies if it is the first clustering performed.

properties that are useful in CABESS. It is easy to compute centroid, radius, and diameter of any group of the data instances from the sufficient statistics of a CF-vector. Moreover, a CF-vector also has properties that allow us to update the group information by using only the sufficient statistics, for example: adding a new data instance, merging two clusters, and subtracting a data instance from a group.

In the second step, CABESS performs a macro-clustering over the resulted summarized representation $\mathcal{S}^t$ using an unsupervised clustering algorithm. This step can employ either DBSCAN or K-MEANS algorithms. Following these steps, CABESS allows the user to provide instance-level feedback regarding a data partition set Π^t, which is used to adjust the clustering process performed at the next timestamp $(t+1)$. In order to do so, the instance-level feedbacks are used to deduce the summarized-level constraints $\mathcal{R}^t$ (step 3). As the deduction process is a contribution derived from this particular work, it is detailed in Sections 4.1 and 4.2.

After the semi-supervision deduction, a semi-supervised clustering algorithm is used to re-organize the data partitions (step 4). Examples of algorithms that are suitable to use in this step are SSDBSCAN and MPCK-MEANS. There arises the need to emphasize that the semi-supervision required by the semi-supervised clustering algorithm must be the same type as the semi-supervision deduced in step 3. Thus, if we choose SSDBSCAN, step 3 must deduce semi-supervision in the form of labels, as SSDBSCAN deals with labels. The last step is responsible for detecting transitions between the current partition set Π^t and the previous partition set Π^{t-1} (step 5). Both MONIC and MCLusT can be chosen for use in this step. The CABESS clustering process finishes when the resulted partition set satisfies the user and when there are no new instances generated in $\mathcal{D}$.

It is worth mentioning here that the clustering algorithms performed in steps 2 and 3 consider the summarized information obtained from the original data. After the clustering process, the user can provide feedback. Thus, a summarized data instance $s_1 \in \mathcal{S}^t$ of cluster $\Pi_i^t \in \Pi^t$, in fact, may represent more than one data instance. Hence, semi-supervision information involving s_1 imposes constraints on all instances represented by s_1.

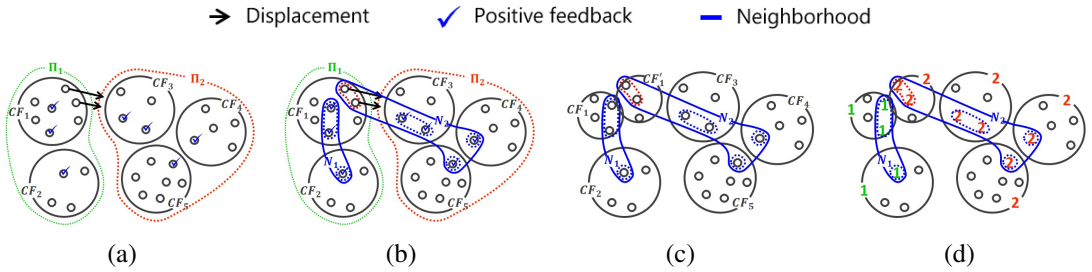


Figure 3. Example of the extraction of the labels from feedback (a,b) and the deduction labels from instance-level to summarized-level (c,d).

4. Pointwise CABESS

In order to illustrate an instantiation of the CABESS framework we will present the Pointwise CABESS. This CABESS instance employs the BIRCH algorithm in order to summarize the data (step 1). The other algorithms used in Pointwise CABESS were the DBSCAN in step 2 to perform the first clustering without any semi-supervision, then SSDBSCAN was adopted to detect clusters with summarized-level labels in step 4, and in the final step, MONIC was used to detect external transitions.

Pointwise CABESS deals with labels as semi-supervision information. In the next section, we explain how Pointwise CABESS extract instance-level labels and how it deduces the summarized-level labels.

4.1. Extracting labels from feedbacks

The summarized-level labels are obtained in two phases, firstly the instance-level labels are extracted from feedback (see Figures 3a, 3b) and then the summarized-level labels are deduced from instance-level labels (see Figures 3c, 3d).

Instance-level labels. The main idea behind the extraction process concerns to the concept of neighborhood. A neighborhood is defined as a set of instances that must be in the same cluster. CABESS computes the neighborhoods as in our main reference [Lai et al. 2014] by following two rules. The first states that all instances of the same cluster that received positive feedback at a previous timestamp are assigned to the equivalent neighborhood. The second states that all instances of different clusters that received displacement feedback at a previous timestamp are assigned to the neighborhood of the initial rule, since the the destination/actual cluster is the same as the instances in the initial rule. Following this, we assign identical labels for instances that are in the same neighborhood. One notes that in Fig. 3b the neighborhood associated with Π_1 will be assigned to label 1 at the end of the process (Fig. 3d)

Summarized-level labels. The deduction of summarized-level labels is performed as a propagation task. CABESS aims to assign labels to Cluster Feature Vectors (CFs) in Fig. 3c. For each summarized instance that contains labeled instances, we assign the instance-level label to the summarized-level label. If one of the summarized instances has labeled instances with different labels, then we need to split it in order to obtain purified summarized instances, i.e., summarized instances that contain only the same label in labeled instances. Noted in Fig. 3d is that CF_1 and CF_2 received label 1, as these summarized instances which are in neighborhood N_1.

Table 1. Datasets employed in the experiments.

Name	# instances	d	# classes	Reference	Type
DB_7	9,050	2	8	[Silva et al. 2015]	
SYN_3	5,000	2	3	streamMOA	Synthetic
SYN_4	10,000	3	5	streamMOA	
FROGS	1,484	8	4	[Colonna et al. 2016]	
IPEA	5,564	5	27	IPEA	Real
$KDD'99_5$	24,692	19	11	UCI	

4.2. Dealing with obsolete labels

Obsolete labels are labels assigned to instances, for which the clusters do not exist anymore. The data instances are still active but they are assigned to other clusters. These labels appear when an external cluster transition happens. For example, let us suppose there is a cluster Π_1^0 at the timestamp $t = 0$ and also that at the next time, Π_1^0 splits into two new clusters Π_{10}^1 and Π_{11}^1. Thus, the label associated with Π_1^0 is not valid at $t = 1$, because it does not help the semi-supervised clustering process.

In order to minimize the problem of obsolete labels, CABESS adopts a detector of transitions. This approach aims at allowing for better neighborhood management. Neighborhoods are responsible for generating new labels and removing obsolete labels. Thus, when a cluster survives both neighborhood and associated labels are preserved. When a split transition is detected, the label associated with the previous cluster is removed and the neighborhood is divided in order to create two or more neighborhoods and new labels are generated accordingly.

5. Experiments

The experiments were conducted using 6 datasets[1], 3 with real data and the other 3 with synthetic data. The details concerning each one are summarized on Table 1. All these datasets share a common feature that allows us to run our experiments, each of its data instances is multi-labeled where labels are in a hierarchical structure.

From the three synthetic datasets, with Gaussian distribution, $\mathbf{SYN}_3$ and $\mathbf{SYN}_4$ were generated using the `RandomRBFGenerator` available at `streamMOA`[2], an interface of MOA[3] (Massive Online Analysis) developed for the R programming language. Let us now consider a third synthetic dataset, denominated as DB7, this is a 2D dataset that contains clusters of different spatial distributions. The grouping trees for the abovementioned synthetic datasets were simulated according to the distance among clusters. Thus, we consider two or more clusters belonging to a large cluster if they are closer than the other clusters.

Considering the real datasets, in **IPEA** dataset[4], each tuple corresponds to one of the 5,564 Brazilian cities among within its Federative unit and Region, and is composed of 5 features with localization information (latitude and longitude) and continuous values related to the following development indexes: IDHM-Longevity, IDHM-Education,

[1]Available at: http://guilhermealves.eti.br/research/data/

[2]Documentation available at: https://cran.r-project.org/web/packages/streamMOA/streamMOA.pdf.

[3]A framework for data stream mining: http://moa.cms.waikato.ac.nz/

[4]Brazilian Institute of Applied Economics Research, IPEA: http://www.ipeadata.gov.br

Table 2. Parameters setting for each dataset.

Algorithm	Parameter	IPEA	KDD'99$_5$	FROGS	DB$_7$	SYN$_3$	SYN$_4$
BIRCH	B and L	500	10k	500	500	50	50
	T	0.07	0.01	0.25	0.01	0.0075	0.0075
DBSCAN	eps	0.075	0.02	0.5	0.075	0.075	0.075

and IDHM-Income. In the **FROGS** dataset, each tuple corresponds to an audio recording of *Anura*, an order of the Amphibian class, and is composed using information concerning specie, genus, and family. **KDD'99**$_5$[5] is a dataset generated by sampling 5% of data instances from the 10% subset of the original KDD Cup 99 dataset, maintaining the proportion of instances in each class. Moreover, this sampled dataset was processed with PCA in order to reduce the dimensionality, as it is the dataset with the highest number of dimensions. We consider two levels of labels found in **KDD'99**$_5$; (1) if the access is an intrusion or not, and (2) if it is an intrusion, what is the type of intrusion.

The main goals of the experiments were to assess the effectiveness and the efficiency of our proposed framework instance, Pointwise CABESS, in comparison with three baseline approaches, two semi-supervised and one unsupervised, considering the four questions presented in Sections 6.1, 6.2, 6.3, and 6.4.

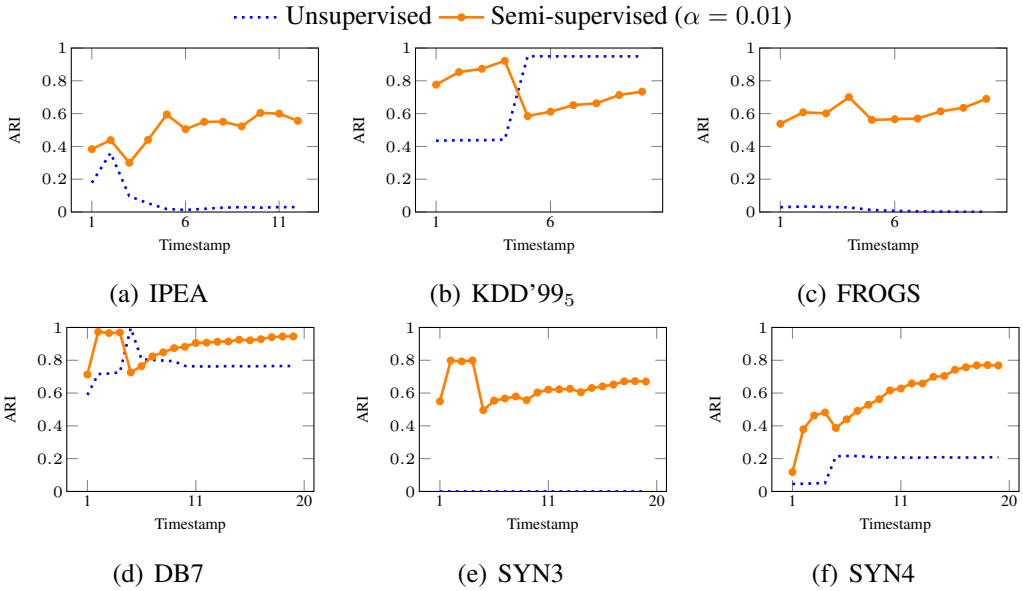
The **Unsupervised Approach** consists in periodically executing a clustering algorithm without any semi-supervision. In our experiments, we adopted DBScan, an efficient well-known unsupervised clustering algorithm. The semi-supervised approaches were run considering two strategies, those being static and window based. The **Static Approach** consists in periodically applying a semi-supervised clustering algorithm. In our experiments, we adopted SSDBScan providing the true labels as semi-supervision. Notice that this approach does not discard any label over time. The **Window-based Approach** is a variation of the previous approach, where instead of executing the clustering algorithm over all the semi-supervision set, we remove old semi-supervision information (labels).

All experiments were implemented within the same platform using the Java programming language. In order to evaluate the effectiveness of the approaches we compared the clustering results against the optimal partition set using the Adjusted Rand Index (ARI) [Hubert and Arabie 1985]. ARI is an external criteria indicated when researchers have a-priori information about the desirable partition set. In our experiments, the desirable partition set is obtained from the true labels in the datasets. For each timestamp t only one label is considered valid for a data instance according to the grouping tree.

Each experiment over Pointwise CABESS was performed based on the Prequential Protocol. Hence, the effectiveness of Pointwise CABESS is evaluated before the user provides current feedback. The other approaches are evaluated periodically without any delay between the new partition set and new feedback.

Considering that our datasets have no online arrival for instances neither for feedback, and our main reference simulates the online aspects of getting user binary feedback, we also simulated the temporal aspects of the data instances and the user interaction phase for obtaining the semi-supervision. Hence, the arrival of the data instances and the user feedback are given according to the uniform distribution. Furthermore, we insert an ex-

[5]UCI KDD archive: http://kdd.ics.uci.edu/



(a) IPEA (b) KDD'99_5 (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

Figure 4. Effectiveness assessment.

ternal cluster transition, according to the grouping tree, at $t = 5$ in order to evaluate the ability of the approaches to adapt to new clusters. In each dataset, one or more clusters are specialized into two or more new clusters and feedback, which are generated according to these new clusters.

Table 2 shows the parameters used in BIRCH and DBSCAN algorithms for each dataset. For the other algorithms, the parameters adopted are the SSDBSCAN in step 4, which used the same value of DBSCAN for $minPts = 2$, and in the final step, MONIC was run using $\tau = 0.7$ and $\tau_{split} = 0.1$. The experiments were run on an Intel Core i7 (3.4 GHz) with 12 GB of RAM, SATA3 HD of 1.31 TB (7,200 rpm) on Windows 7 x64.

6. Results and Discussion

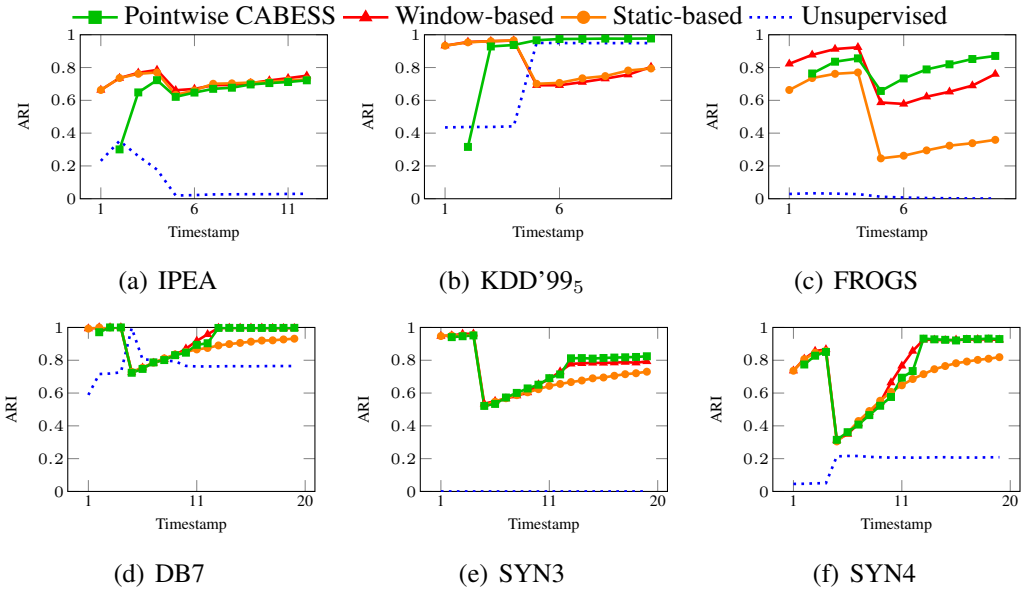
Sections 6.1, 6.2, 6.3 and 6.4 present the discussion of the results obtained throughout the experiments, which took into consideration four questions.

6.1. How accurately does semi-supervision aid on clustering effectiveness when there are external clustering transitions over time? (Q1)

In order to answer Q1, we chose to compare the results obtained running the Unsupervised Approach and the Static Approach on each dataset. This question is important for determining whether a clustering approach can benefit from semi-supervision information. Figure 4 shows ARI considering the semi-supervision rate α set to 1% for the Static Approach. One notes that semi-supervision performs better in most of the datasets. In KDD'99_5 semi-supervision improved the clustering quality before the cluster transition ($t = 5$). Therefore, the results achieved corroborate that low rates of semi-supervision contribute to the obtainment of a more effective clustering in a majority of the scenarios.

6.2. Are there major differences between clustering effectiveness when using semi-supervised clustering approaches based on feedback and labels? (Q2)

One of the main contributions of our approach is its ability of dealing with semi-supervision specified through feedback. The previous question focused on understanding whether online clustering could benefit from semi-supervision. Next, we evaluated



(a) IPEA (b) KDD'99_5 (c) FROGS

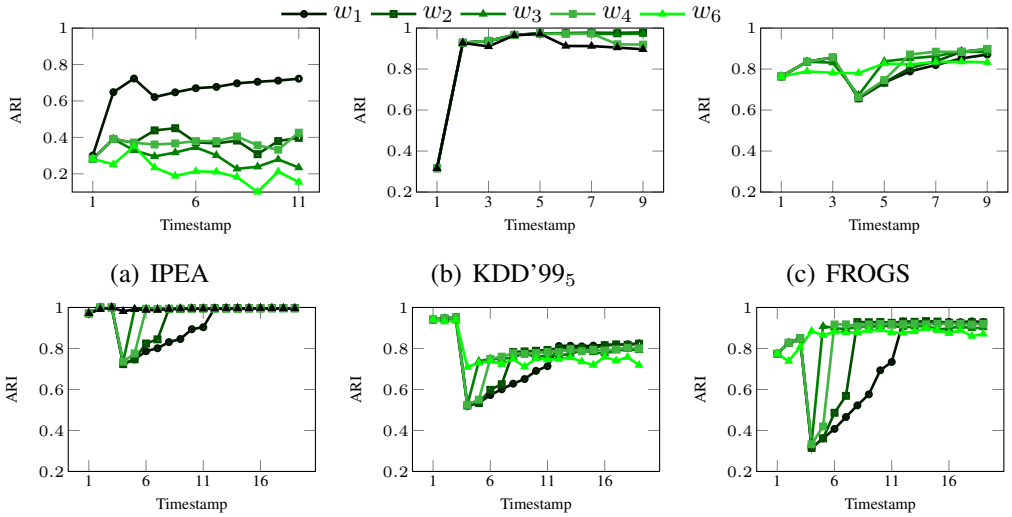
(d) DB7 (e) SYN3 (f) SYN4

Figure 5. Effectiveness comparison between Pointwise CABESS and the baseline approaches ($\alpha = 0.1$ and w_1).

the performance of traditional semi-supervised approaches against our approach. Figure 5 shows ARI for all the baseline approaches considered and the Pointwise CABESS. In this experiments, we adopted $\alpha = 0.1$ and the largest semi-supervision window size w_1. By analyzing the obtained results, one notes that Pointwise CABESS performed better than other approaches above all after the external transition when considering the datasets KDD'99_5 and FROGS. For the other datasets, the effectiveness obtained for the semi-supervised approaches were equivalent. Emphasis is given here to the fact that the semi-supervision information used by Pointwise CABESS is lower standard when compared to the semi-supervision used by the other approaches. Note that our framework receives feedback instead of labels. Thus, it needs to infer and maintain labels correct over time. The experimental results obtained showed that there are no major differences in the clustering quality by using using semi-supervision in the form of labels or inferring labels from feedback, as is the case in our approach.

6.3. How the feedback window size variation affects semi-supervision information and clustering effectiveness? (Q3)

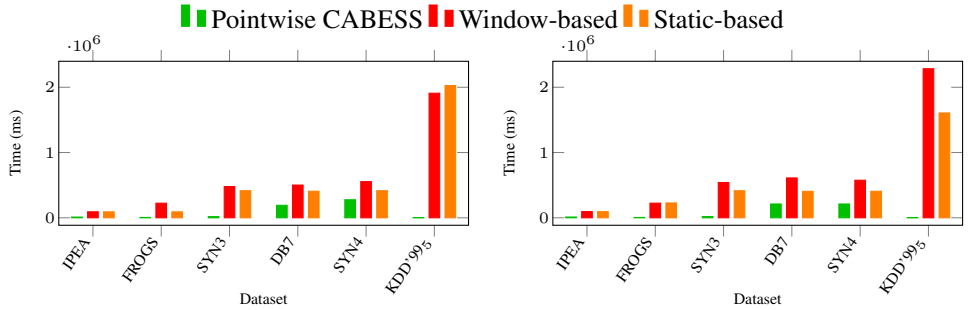
In data stream mining literature there is a common assumption that small windows mitigate the adaptation time of the clustering process [Gama 2010]. In order to answer Q3 we adopted six different semi-supervision window sizes to assess it. Figure 6 shows the Pointwise CABESS ARI for each window setting: $w_6 < w_4 < w_3 < w_2 < w_1$. Some curves are overlapped, e.g, w_6 and w_5 in DB7. Analyses of the obtained results shows that the small window w_6 showed quicker adaptation but in some cases, the quality was unstable. This occurs due to some change of instance over the cluster at each iteration. Note that when we have a large window size w_1, there is a notable slow adaptation but we can see stable results. The use of the detector of transitions is responsible for producing this behavior. Moreover, running Pointwise CABESS over small label windows is the same as the Window-based Approach, due to the fact that a small window size mitigates the benefic effects obtained when using the detector of transitions.



(a) IPEA (b) KDD'99_5 (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

Figure 6. Effectiveness assessment considering different window sizes.



(a) $\alpha = 0.1$ and w_1 (b) $\alpha = 0.01$ and w_6

Figure 7. Mean runtime to cluster different datasets using high (a) and low (b) semi-supervision rates.

6.4. How efficient is our approach compared to existing semi-supervised approaches? (Q4)

Another contribution of our approach is its ability to summarize data instances and semi-supervision information. Here, we quantified the efficiency of our approach and compared it against the efficiency of the existing semi-supervised algorithms. Figure 7 shows the run times for Pointwise CABESS and the other semi-supervised approaches. Through an analysis of these results, one observes that our framework instance performed faster than the other approaches. The main reason for this behavior is the summarizing algorithm used in its first step. Then, the semi-supervised clustering algorithm considers a smaller number of summarized instances compared to the number of instances considered by the other semi-supervised approaches.

7. Conclusion

The goal of the proposed framework presented herein, CABESS, is to assist online clustering in coping with external transitions. Our experiments and analyses were performed driven by four research questions on three real datasets and three synthetic datasets. The results showed that our approach presents a higher efficiency when compared to other semi-supervised approaches while keeping an equivalent effectiveness.

With CABESS, we can put forward new directions to extend traditional semi-supervised clustering techniques. Our work could be extended to explore other types of semi-supervision information such as instance-level constraints (must-link and cannot-link) that will allow for the employment of other types of semi-supervised clustering algorithms into CABESS. Another interesting direction for future work is to tackle other strategies for detecting transitions and to explore other types of external transitions.

References

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2003). A framework for clustering evolving data streams. In *VLDB*, pages 81–92. VLDB Endowment.

Barioni, M. C. N., Razente, H., Marcelino, A. M. R., Traina, A. J. M., and Traina, C. (2014). Open issues for partitioning clustering methods: An overview. *WIREs Data Min. and Knowl. Disc.*, 4(3):161–177.

Basu, S., Davidson, I., and Wagstaff, K. (2008). *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman and Hall/CRC.

Bilenko, M., Basu, S., and Mooney, R. J. (2004). Integrating constraints and metric learning in semi-supervised clustering. In *ACM ICML*, page 11, New York, NY, USA.

Castellano, G., Fanelli, A. M., and Torsello, M. A. (2013). Shape Annotation by Incremental Semi-supervised Fuzzy Clustering. In *WILF*, volume 8256 of *LNCS*, pages 193–200. Springer.

Colonna, J. G., Gama, J., and Nakamura, E. F. (2016). Recognizing Family, Genus, and Species of Anuran Using a Hierarchical Classification Approach. pages 198–212. Springer, Cham.

Dubey, A., Bhattacharya, I., and Godbole, S. (2010). A Cluster-Level Semi-supervision Model for Interactive Clustering. pages 409–424.

El Moussawi, A., Cheriat, A., Giacometti, A., Labroche, N., and Soulet, A. (2016). Clustering with Quantitative User Preferences on Attributes. In *IEEE ICTAI*, pages 383–387.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231. AAAI Press.

Gama, J. (2010). *Knowledge discovery from data streams*. Chapman & Hall/CRC.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice-Hall, USA.

Lai, H. P., Visani, M., Boucher, A., and Ogier, J.-M. (2014). A new interactive semi-supervised clustering model for large image database indexing. *Pattern Recognition Letters*, 37(1):94–106.

Lelis, L. and Sander, J. (2009). Semi-supervised Density-Based Clustering. In *IEEE ICDM*, pages 842–847.

Liu, E. Y., Zhang, Z., and Wang, W. (2011). Clustering with relative constraints. In *ACM SIGKDD*, page 947, New York, NY, USA.

Oliveira, M. D. and Gama, J. (2010). Bipartite graphs for monitoring clusters transitions. In *IDA*, pages 114–124. Springer.

Pereira, G. and Moreira, J. (2016). Monitoring clusters in the telecom industry. In *New Advances in Information Systems and Technologies*, pages 631–640. Springer.

Ruiz, C., Spiliopoulou, M., and Menasalvas, E. (2007). *C-DBSCAN: Density-Based Clustering with Constraints*, volume 4482 of *LNCS*. Springer.

Silva, W. J., Barioni, M. C. N., de Amo, S., and Razente, H. L. (2015). Semi-supervised clustering using multi-assistant-prototypes to represent each cluster. In *SAC*, pages 831–836, New York.

Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., and Schult, R. (2006). MONIC. In *ACM SIGKDD*, page 706, New York, NY, USA. ACM Press.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). BIRCH: An Efficient Data Clustering Method for very Large Databases. *ACM SIGMOD Record*, 25(2):103–114.

paper:171404

A Spline-based Cost Model for Metric Trees

Marcos V. N. Bedo[1,2], Agma J. M. Traina[2] and Caetano Traina Jr.[2]

[1]Fluminense Northwest Institute – Fluminense Federal University (UFF)
St. A. Pádua – RJ – Brazil

[2]Institute of Mathematics and Computer Science – University of São Paulo (USP)
São Carlos – SP – Brazil

{bedo,agma,caetano}@icmc.usp.br

Abstract. *Whenever two (or more) access methods are alternatives for the execution of a query, how to choose which one is the best for the task? Such a decision is made by the DBMS optimizer module, which models the query costs according to the distribution of the data space. Cost modeling of similarity searches, however, requires the representation of distances' rather than data distribution. In this paper, we propose the Stockpile model for cost estimation of similarity queries on metric trees by using pivot-based distance histograms that represent the local densities around the query elements. By combining the local densities to the probability of traversing the tree nodes, Stockpile provides a fair estimation of both disk accesses (I/O costs) and distance calculations (CPU costs). We compared Stockpile and two literature models regarding similarity queries in real-world data sources and our model was up to* 85% *more precise than the competitors.*

1. Introduction

Similarity searching is a foundational paradigm for many computer applications, such as content-based retrieval, classification, clustering, and data visualization [Zezula et al. 2006]. In practice, two of the most requested similarity operations are the range and k-NN searches. An example of range query is **(Q1)** "List the bottles in the wine cellar whose combination of fixed and volatile acidities differs at most 4 mg/L to this Italian wine", while a k-NN query example is **(Q2)** "Find the 3 closest cabs to this restaurant". Range and k-NN queries can be modeled upon a *metric space*, where the elements (bottles and cabs, in the examples) are represented as points and the (dis)similarity between each pair of points is evaluated by a distance function.

Formally, a metric space is a pair $\mathcal{M} = \langle \mathbb{S}, \delta \rangle$, where $\mathbb{S}$ is the domain of the points and δ is a metric that complies with the properties of symmetry, non-negativity, and triangular inequality. Accordingly, given a data source $S \subseteq \mathbb{S}$, a query element $s_q \in S$ and a threshold $\xi \in \mathbb{R}_+$, a range query Rq retrieves every element in S within the closed ball centered at s_q with radius ξ such that $Rq(S, s_q, \xi) = \{s_i \in S \mid \delta(s_i, s_q) \leq \xi\}$. On the other hand, a k-NN query returns a quantity $k \in \mathbb{N}$ of elements whose distance to the query element s_q are the smallest. In an equivalent way, a k-NN query can be seen as a variation of the range query, i.e., a range search with a set radius ξ such that $|Rq| = k$ [Tasan and Özsoyoglu 2004].

Several indexing schemes in the form of metric access methods have been proposed to speed up similarity searching [Lokoč 2010]. Particularly, *tree-based* methods

stand out as the most suitable strategies for the indexing of very large data sources as they organize the search space into a hierarchical and balanced fashion. Remarkable tree-based structures include the M-Tree [Ciaccia et al. 1997] and its variations, such as Slim-Tree [Traina Jr. et al. 2002] and PM-Tree [Skopal et al. 2004]. Although tree-based indexes focus on minimizing both distance calculations (through the clustering of the elements) and disk accesses (by using database paging principles), their performances differ depending on the query. Therefore, given a similarity query over a data source indexed by two or more access methods, a database query optimizer must decide which method will be employed to execute the search. Such a query optimizer' decision is made upon a *cost model* for metric trees.

Thus far, most of the research effort has addressed the cost of similarity queries regarding multidimensional spaces [Korn et al. 2001, Tao et al. 2004]. However, besides inapplicable to non-dimensional metric spaces, many of these models are also unable to handle cost estimation of k-NN queries. One of the challenging aspects of modeling k-NN searches is no limiting radius is known beforehand. Accordingly, existing k-NN models estimate a threshold as the query radius by using the pairwise distance distribution within S. Such estimates follow a *biased assumption*: query elements are more likely posed in high-density areas of the search space. For instance, the model for multidimensional spaces in [Aly et al. 2015] assumes the k-NN radii follow a uniform distribution regarding fixed intervals of k, while the models in [Ciaccia et al. 1998] and [Baioco et al. 2007] assume k-NN radii follow a binomial and an exponential distribution, respectively. The main drawback of such models is they disregard the 'locality' of each query, i.e. they rely on a global and pairwise distance distribution without properly considering the density around each query element.

In this study, we propose the *Stockpile cost model* for the estimation of similarity searching costs in tree-based methods. Stockpile distinguishes itself from previous models as it estimates the cost of range queries according to their locality. The overall idea of our approach is building a small set of pivot-based distance histograms in such a way the densities around query elements are represented. Additionally, by using pivot-based histograms to predict the k-NN radii, the local densities also are taken into account in the cost estimation of k-NN searches. Stockpile histograms are built as splines on pivot-based distance distributions according to a fixed number of buckets so that they easily fit into main memory. Aiming at evaluating the accuracy of Stockpile, we compared our approach to models in [Ciaccia et al. 1998] and [Baioco et al. 2007] regarding real-world data sources and results showed Stockpile outperforms both competitors in terms of accuracy. Accordingly, the main contributions of the paper are as follows:

- We introduce the Stockpile cost model, which estimates the cost of similarity queries by using the locality of the query element,
- We experimented with our approach over real-world data sources and results indicate Stockpile predictions are more accurate than those obtained by models in [Ciaccia et al. 1998] and [Baioco et al. 2007].

The remainder of the paper is organized as follows. Section 2 summarizes related work. Section 3 introduces Stockpile and its parameters. Sections 4, 5 and 6 show the results of evaluations performed, while Section 7 concludes the paper.

2. Background and Related Work

2.1. M-Tree and Slim-Tree

The M-Tree [Ciaccia et al. 1997] is a dynamic structure that hierarchically organizes the elements within a metric space into closed balls, which are stored as *nodes*. Basically, M-Tree uses two types of nodes ($\mathtt{N}$), namely directory ($\mathtt{N}_d$) and leaf ($\mathtt{N}_l$) nodes. Directory nodes store a set of balls, while leaf nodes store the indexed elements themselves. Accordingly, a leaf node $\mathtt{N}_l$ has the format $\mathtt{N}_l(s_j) = \{\langle s_i, d_{ij}\rangle\}$, where s_i is a data element such that $s_i \in S \subseteq \mathbb{S}$, and d_{ij} is the distance of s_i to the rooting element s_j (parent) of the leaf node. A directory node $\mathtt{N}_d$ has the format $\mathtt{N}_d(s_j) = \{\langle s_i, \xi_i, \mathtt{N}(s_i), d_{ij}\rangle\}$, where $s_i \in S$ is the rooting element to the subtree $\mathtt{N}(s_i)$, ξ_i is the covering radius of $\mathtt{N}(s_i)$, and d_{ij} is the distance of s_i to rooting element s_j. Precomputed distances d_{ij} are employed for the pruning of nodes/elements when a similarity query is executed [Traina Jr. et al. 2002, Skopal et al. 2004]. The top directory node is the root node: it has no parent and covers all indexed elements. Figure 1 shows two M-tree examples under the L_2 distance.

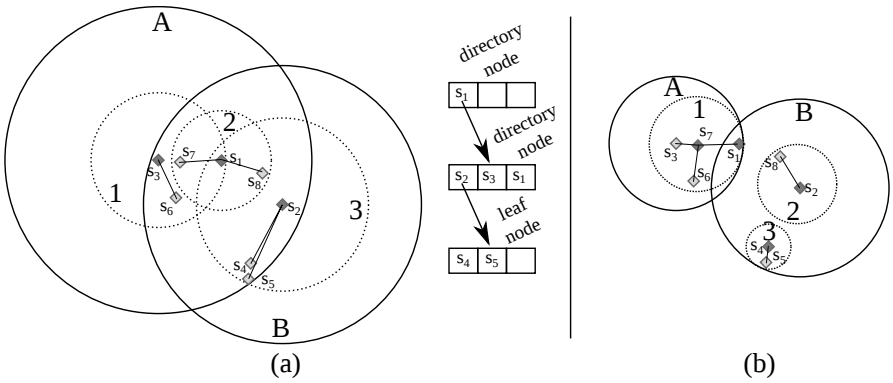


Figure 1. Two M-Trees indexing the same elements. (a) M-Tree with a large overlap, and (b) M-Tree with a smaller overlap.

Distinct partitioning strategies can be applied for the construction of a valid M-Tree such that the nodes may overlap, but all elements rooted by s_j are within the covering radius of $\mathtt{N}(s_j)$. Notice, however, the larger the "volume"[1] of the tree, the higher the probability of occurring intersections among the nodes. For instance, although both partitions of Figures 1(a) and (b) generate valid M-Trees, the tree in Figure 1(b) has a smaller volume in comparison to the tree in Figure 1(a). The Slim-Tree method extends the M-Tree by using improved partitioning algorithms that minimize the volume of the tree through the evaluation of the overlaps in terms of a single measure, called *fat-factor* [Traina Jr. et al. 2002]. Formally, the fat-factor of a given tree T regarding a data source S is defined by Equation 1.

$$fat(T) = \frac{I_C - |S| \cdot h}{|S|} \cdot \frac{1}{m - h} \tag{1}$$

where I_c denotes the sum of node accesses for the execution of point queries, i.e. $Rq(S, s_i, 0.0)$, for every $s_i \in S$, h is the height of the tree, and m is the total number

[1]We quoted "volume" as there is no universal notion of volume in metric spaces. In practice, the number of elements covered by the ball can be used as a suggestion for the volume.

of nodes. The fat-factor expresses how "good" is the tree in a $[0, 1]$ scale with regards to the overlapping of elements at the same level. For instance, the fat-factor of the tree in Figure 1(a) is 0.5, while the same measure for Figure 1(b) is ≈ 0.04. In this scenario, the tree in Figure 1(b) is more likely to avoid duplicate scans of the search space in comparison to the tree in Figure 1(a) when executing a query.

2.2. Costs models for range and k-NN queries

The relationship between elements is expressed by means of distances within metric trees. Therefore, estimating a query cost involves the representation of the *distance distribution* on S regarding a metric δ. Basically, two types of distance distributions can be gathered from a metric tree, namely the pairwise and the pivot-based distance distribution. A *pairwise distance distribution* $\mathcal{T}$ is a *single* distribution that captures the frequency of distances between every pair of elements in S. Therefore, a joint and normalized pairwise distance distribution $\mathcal{T}^{\mathcal{C}}$ can be seen as a probability function $F(x)$ so that $F(x) = \texttt{Prob}\{\delta(s_i, s_j) \leq x\},\ \forall\ s_i, s_j \in S$. The proposal in [Ciaccia et al. 1998] follows this rationale and assumes the cost of similarity queries can be estimated by a *biased* query model, which implies that the distances between the query element and points in S are supposed to follow $\mathcal{T}^{\mathcal{C}}$. In this case, given a range query $Rq(S, s_q, \xi)$, the probability of scanning a node rooted by s_i with radius ξ_i of a tree-based method T that indexes S is expressed as Equation 2.

$$\texttt{Prob}\{\delta(s_q, s_i) \leq \xi + \xi_i\} \approx F(\xi + \xi_i) \tag{2}$$

The number of scanned tree nodes is estimated by the sum of the probability of accessing each node as in Equation 3. Analogously, the number of comparisons between s_q and the element in T is the sum of the weighted probabilities of accessing each node, where the weight is the number of entries in that node as in Equation 4.

$$\texttt{nodes_scanned}_{Ciaccia}(T, s_q, \xi) \approx \sum_{i}^{m} F(\xi + \xi_i) \tag{3}$$

$$\texttt{distances_calculated}_{Ciaccia}(T, s_q, \xi) \approx \sum_{i}^{m} |\texttt{N}(s_i)| \cdot F(\xi + \xi_i) \tag{4}$$

The authors in [Ciaccia et al. 1998] also propose the use of $F(x)$ as part of a binomial probability function so that an estimated radius ξ for a k-NN query can be drawn from $F(x)$. However, their approach has two major drawbacks, namely *(i)* the pairwise distance distribution is expensive to obtain, and *(ii)* the cost of the estimations depends on the number of nodes.

Aiming at avoiding such drawbacks, the study in [Baioco et al. 2007] proposes a model that generalizes the use of the *fractal dimension* for the estimation of query costs. The proposal extends the previous approach of [Korn et al. 2001] for metric trees, where the joint pairwise distance distribution can be constructed following the box-counting algorithm. The authors argue the fractal dimension is a good approximation of the intrinsic dimension and can be calculated through the *Distance-Plot* [Korn et al. 2001, Navarro et al. 2017]. Such a graphic plots $\mathcal{T}^{\mathcal{C}}$ by using $log \times log$ axes

and enables the approximation of the joint frequencies by a linear function. Thus, the model assumes whenever $\mathcal{T}^{\mathcal{C}}$ is a joint exponential distribution, then the slope of the fitting line approximates the fractal dimension $\mathcal{D}$ of S. Under such conditions, the number of scanned regions and distance calculations of a range query $Rq(S, s_q, \xi)$ are given by Equations 5 and 6, respectively.

$$\texttt{nodes_scanned}_{Baioco}(T, s_q, \xi) \approx \frac{1}{R^{\mathcal{D}}} \sum_{i}^{h} |S|^{\frac{i}{h}} \left(\sqrt[\mathcal{D}]{|S|^{\frac{-i}{h}}} + \xi \right)^{\mathcal{D}} \tag{5}$$

$$\texttt{distances_calculated}_{Baioco}(T, s_q, \xi) \approx \frac{1}{R^{\mathcal{D}}} \sum_{i}^{h} |S|^{\frac{i+1}{h}} \left(\sqrt[\mathcal{D}]{|S|^{\frac{-i}{h}}} + \xi \right)^{\mathcal{D}} \tag{6}$$

where R is the covering radius of the entire tree. The authors also claim the distance between the query element and its k^{th} neighbor can be calculated by using $\mathcal{D}$, which enables the transformation of k-NN into range queries. The model in [Baioco et al. 2007] considers k-NN overestimated radii are preferable to underestimated ones, a subject deeply discussed in [Vieira et al. 2007]. Therefore, the model adds a controlled overestimation based on the characteristics of the tree, i.e. the fat-factor of the index, and scales Equations 5 and 6 by the constant $(1 + fat(T))$.

Other models also rely on a single and biased representation of the pairwise distance distribution. For instance, the study in [Tao et al. 2004] extends the binomial approach in [Ciaccia et al. 1998] for k-NN cost estimation in low dimensional spaces, while the proposal in [Lu et al. 2014] estimates the query costs by combining S to other domains. Recently, the study in [Aly et al. 2015] introduced the *Staircase* model for k-NN queries in multidimensional spaces. The authors argue the cost of such queries is *stable*, i.e., the cost of executing a k-NN query with larger k can be the same of executing a query with a smaller k as the same nodes of the index are accessed and provided the incremental k-NN searching procedure is employed [Hjaltason and Samet 2003]. The model assumes a uniform distribution for fixed intervals of k, which results in a compact representation of the indexed data. However, distance distributions hardly follow the uniform assumption [Korn et al. 2001], which harm the usability of the model in tree-based methods.

Notice the biases of the reviewed models are mainly related to the adoption of a single representation of the pairwise distance distribution. The study in [Tasan and Özsoyoglu 2004] comes up with a suggestion to avoiding such a bias in the task of predicting k-NN radii. Basically, the authors propose the gathering of distribution $\mathcal{T}$ in the form of a histogram $H(x),\ H(x) \approx \mathcal{T}$ so that no inference on the type of the distance distribution itself is required. Therefore, the radius ξ' of a k-NN query can be straightforwardly estimated by Equation 7.

$$\int_{0}^{\xi'} H(x)d(x) = k \tag{7}$$

Moreover, the authors show the *unique* representation of the pairwise distribution can be replaced by pivot-based distance distributions. Such an observation is similar to the "Homogeneity of Viewpoints" property in [Ciaccia et al. 1998], but in

[Tasan and Özsoyoglu 2004] the authors suggest keeping the pivot-based distributions instead of replacing them by the pairwise distribution. Formally, a pivot-based distribution $\mathcal{T}_p$ on S for a pivot $p \in \mathcal{P} \subseteq S$ follows the Definition 2.1.

Definition 2.1 (Pivot-based distance distribution – $\mathcal{T}_p$) *Given a data source S, a metric δ, and a pivot $p \in \mathcal{P}$, $\mathcal{T}_p$ captures the distance from each $s_i \in S$ to p. Distance value set $\mathcal{V}_p$ contains the distinct and sorted values of $\delta(s_i,\ p)$, i.e., $\mathcal{V}_p = \{v_p(j) : 1 \leq j < m_p,\ m_p \leq |S|\}$, where $v_p(m_p)$ is the largest distance between any s_i to p. Frequency $f_p(j)$ is the number of elements of S whose distance $\delta(s_i,\ p) = v_p(j)$. The pairs $\{\langle v_p(j), f_p(j)\rangle,\ \forall\ v_p(j) \in \mathcal{V}_p\}$ constitute $\mathcal{T}_p$, i.e. $\mathcal{T}_p = \{\langle v_p(1), f_p(1)\rangle, \ldots, \langle v_p(m_p), f_p(m_p)\rangle\}$. $\mathcal{T}_p^+$ is the extension of $\mathcal{T}_p$ to the entire domain of distances by setting 0 as the frequency for any $v_p \in \mathbb{R}_+ \setminus \mathcal{V}_p$.*

Equally spaced Equi-Width histograms can be employed for the approximation of $\mathcal{T}_p$ such that $H_p(x) \approx \mathcal{T}_p$. More sophisticated histograms enable the bounding of the approximation error as in V-Optimal [Ioannidis 2003], Curve-Fitting [König and Weikum 2002], or Compact-Distance Histograms [Bedo et al. 2015]. Particularly, a Compact-Distance Histogram (CDH) can be seen as a continuous piecewise linear function whose squared error to $\mathcal{T}_p$ is minimal so that optimal CDHs are obtained by using the squared error as the optimization target in the Bellman-Ford algorithm [Ioannidis 2003]. Therefore, given a k-NN query, a set of pivots $\mathcal{P} = S$, and a set of pivot-based distance histograms $H_p(x)$, Equation 7 can be rewritten as Equation 8.

$$k = \texttt{Prob}(p) \cdot \int_0^{\xi'} H_p(x)dx \tag{8}$$

where `Prob(p)` is a binary probability of p having the same distance distribution of the query element of the k-NN search. Although theoretically removing many of the biases from previous models, the estimations in [Tasan and Özsoyoglu 2004] still have two practical setbacks. First, both pairwise distance distribution and the set of pivot-based distributions are needed, which entails an expensive construction cost. Last but not least, it is not always possible to set $\mathcal{P} = S$ due to memory constraints.

In this paper, we follow the indications of both [Ciaccia et al. 1998] and [Tasan and Özsoyoglu 2004] for the creation of a cost model that takes into account the locality of query elements and that comply with memory constraints.

3. The Stockpile cost model

In this section, we propose a cost model for predicting the number of disk accesses and distance calculations for any range or k-NN query to be executed by a metric tree. The model itself relies on two previously computed data structures, namely *(i)* a set of histograms and *(ii)* meta-statistics, e.g. the fat-factor, about the metric tree. Such structures are typically kept in main memory as a *pile* of resources (and hence the name *Stockpile* for the model) to be evaluated on-the-fly according to user-posed queries, which requires the model structures to be parameterized in terms of space constraints. Histograms are especially suitable for this scenario, as they enable the use of the constraint *maximum number of buckets* ($\mathcal{B}$) for the representation of the distributions. Stockpile employs Compact-Distance Histograms (CDHs) for the approximation of pivot-based distance distribution

as splines. Accordingly, we denote a CDH by $f_p(x)$ such that $f_p(x) \approx \mathcal{T}_p$. Figure 2(a) shows an example of a $\mathcal{T}_p$, while Figure 2(b) presents three possible CDHs by using $\mathcal{B} = 4, 5$ and 6 buckets, respectively.

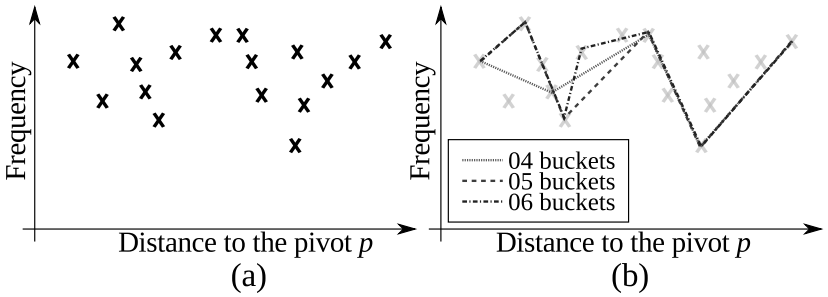


Figure 2. Summarization of a distance distribution $\mathcal{T}_p$ regarding a piecewise linear function $f_p(x)$. (a) Original pivot-based distance distribution, and (b) Compact-Distance Histogram for a maximum of $4, 5$ and 6 buckets.

Notice the *number of pivots* must be limited according to available memory so that constraints $|\mathcal{P}|$ and $\mathcal{B}$ are balanced somehow. Although an exhaustive search of trade-offs can be used for finding such a balance, we argue that a representative number of pivots is given by the intrinsic dimension, which is the rationale employed in several pivot-based problems [Bustos et al. 2003]. Thus, our model uses $|\mathcal{P}| = \mathcal{D}$ and obtains the value of the constraint $\mathcal{B}$ according to the remaining memory space.

The major strength of Stockpile is it takes into account the "locality" of the query element by assuming *pivots closer to query points are more likely to resemble the distance distributions of the query elements.* The premise is especially fair whenever the density of distances around the pivot is uniformly distributed according to $f_p(x)$. Therefore, we assume the probability of the query element having the same distribution of a given pivot is linear with regards to the distance between them. Formally, let s_q be a query element and $\mathcal{P}$ be a set of pivots, the probability $\texttt{Prob}(p)$ of s_q resembling the distribution $f_p(x)$ is proportional to $\delta(s_q, p)$ so that $\texttt{Prob}(p) = d(s_q, p)/C_2$, where $d(s_q, p) = C_1 - \delta(s_q, p)$. Both C_1 and C_2 are local constants that depend on the query element s_q. Such constants are calculated as $C_1 = \sum_{p \in \mathcal{P}} \delta(s_q, p)$ and $C_2 = \sum_{p \in \mathcal{P}} d(s_q, p)$. The joint probability of the query element resembling the pivots in $\mathcal{P}$ is given by Equation 9.

$$\sum_{p \in \mathcal{P}} \texttt{Prob}(p) = \frac{d(s_q, p_1)}{C_2} + \frac{d(s_q, p_2)}{C_2} + \cdots + \frac{d(s_q, p_{|\mathcal{P}|})}{C_2} = \frac{C_1(|\mathcal{P}| - 1)}{C_1(|\mathcal{P}| - 1)} = 1 \qquad (9)$$

Accordingly, Stockpile models the probability of traversing a node as the linear combination of the local information and the densities given by the CDHs.

3.1. Costs estimation of range queries

Suppose a range query $Rq(S, s_q, \xi)$ to be executed by a metric tree T. All leaf nodes in T that intercept the query ball defined by $\langle s_q, \xi \rangle$ must be evaluated because their elements are potentially inside the query ball. Root nodes to these leaf nodes must be evaluated as well. Therefore, the local probability of accessing a node $\texttt{N}(s_j)$ regarding a given pivot p is modeled upon the covering radius of the node (ξ_j) and the range query radius (ξ) as expressed by Equation 10.

$$\begin{aligned} \texttt{Prob}(\text{node is accessed}) &= \texttt{Prob}\{\delta(s_q, p) \leq \xi_j + \xi\} \\ &\approx \bar{F}_p(\xi_j + \xi) = \frac{\displaystyle\int_0^{\xi_j+\xi} f_p(x)dx}{\displaystyle\int_0^{v_p(m_p)} f_p(x)dx} \end{aligned} \tag{10}$$

Naturally, the local probability is 1 whenever $\xi_j + \xi > v_p(m_p)$. The overall probability of accessing a node of T is given by each pivot $p \in \mathcal{P}$ and the joining of Equations 9 and 10 into Equation 11. Similarly, Stockpile combines Equations 9 and 11 into Equation 12 for the estimation of distance calculations in range queries.

$$\texttt{nodes_scanned}(T, s_q, \xi) \approx \sum_{j}^{m} \sum_{p \in \mathcal{P}} \texttt{Prob}(p) \cdot \bar{F}_p(\xi + \xi_j) \tag{11}$$

$$\texttt{distances_calculated}(T, s_q, \xi) \approx \sum_{j}^{m} |\texttt{N}(s_j)| \cdot \left(\sum_{p \in \mathcal{P}} \texttt{Prob}(p) \cdot \bar{F}_p(\xi + \xi_j) \right) \tag{12}$$

The intuition in Equation 12 is the number of distance calculations is proportional to the probability of accessing each node, where $|\texttt{N}(s_j)|$ is either the number of entries (in the case of directory nodes) or the number of elements (in the case of leaf nodes).

3.2. Cost estimation of *k*-NN queries

The cost estimation of k-NN queries requires an additional step in comparison to range queries. Such an extra step is the "reduction" of the k-NN query to a range query by setting the query threshold ξ as the distance between the query element and its k^{th} neighbor. However, such a distance is unknown until the effective execution of the k-NN query and, consequently, Stockpile must estimate the query radius before calculating the query costs. Formally, given a k-$NN(S, s_q, k)$ query and a CDH related to pivot p, Stockpile estimates the distance between s_q and its k^{th} neighbor as the threshold ξ'_p according to Equation 13.

$$\frac{|S|}{\int_0^{v_p(m_p)} f_p(x)dx} \cdot \int_0^{\xi'_p} f_p(x)dx = k \tag{13}$$

where the term $(|S|)/(\int_0^{v_p(m_p)} f_p(x)dx)$ is the uniform distribution of the histogram approximation error among its buckets. Notice the solving of Equation 13 can be carried out by a numeric method, such as Newton-Raphson, as the primitive function for $f_p(x)$ is continuous and monotonically crescent. Accordingly, Stockpile combines the probability of selecting the pivot in Equation 9 to Equation 13 so that k-$NN(S, s_q, k)$ is reduced to a range query whose radius depends on $\mathcal{P}$. The number of T scanned nodes regarding a k-NN query is given by Equation 14, whereas the number of distance calculations is estimated as in Equation 15.

$$\texttt{nodes_scanned}(T, s_q, k) \approx \sum_{j}^{m} \sum_{p \in \mathcal{P}} \texttt{Prob}(p) \cdot \left(\frac{k}{|S|} + \frac{\displaystyle\int_{\xi'_p}^{\xi'_p+\xi_j} f_p(x)dx}{\displaystyle\int_0^{v_p(m_p)} f_p(x)dx} \right) \tag{14}$$

$$\texttt{dist_calc}(T, s_q, k) \approx \sum_{j}^{m} |\texttt{N}(s_j)| \cdot \left[\sum_{p \in \mathcal{P}} \texttt{Prob}(p) \cdot \left(\frac{k}{|S|} + \frac{\int_{\xi'_p}^{\xi'_p + \xi_j} f_p(x)dx}{\int_{0}^{v_p(m_p)} f_p(x)dx} \right) \right] \quad (15)$$

3.3. Optimistic *vs.* pessimistic estimates from Stockpile

The proposed estimates are particularly accurate for *optimal metric trees*, i.e. trees with such a partitioning that generate nodes with no overlap. Accordingly, we call *optimistic estimates* the Stockpile predictions of Equations 11, 12, 14 and 15. However, indexing of real data sources typically results in a non-negligible number of overlaps between the tree nodes. Therefore, a slight overestimation in the number of disk accesses and distance calculations may provide a better approximation for the query real costs [Vieira et al. 2007]. In this scenario, Stockpile uses the fat-factor of T for the calculation of the cost overestimation, which results in the scaling of Equations 11, 12, 14 and 15 by the $(1 + fat(T))$ constant. We call these Stockpile scaled predictions *pessimistic estimates*. Optimistic/pessimistic estimates can be easily set for each metric tree T in a query optimization environment of a database search engine.

4. Experiments

This section reports on a set of experiments performed over four real-world data sources with low to medium dimensionality, as detailed in Table 1. Data sources `CITIES`[2], `WINE`[3], `LETTER`[3], and `CANVAS`[4] were queried by using well-known metrics of the Minkowski family (L_p) and indexed by Slim-Trees. Stockpile model was set to use $\lceil \mathcal{D} \rceil$ pivots selected by the `K-MEDOIDS` strategy [Kaufman and Rousseeuw 1987], CDH's constrained by 128 buckets, and to provide *pessimistic estimations* for both range and k-NN queries. By using a fixed budget of memory, we evaluate our model against two baseline competitors, as follows:

1. The model in [Ciaccia et al. 1998], which was set to use a single Equi-Width histogram constrained by 256 buckets as the representation of the pairwise distance distribution, and
2. The model in [Baioco et al. 2007], which was set to use the fractal dimension drawn from the Distance-Plot graphs.

The compared methods were implemented under the same framework by using the Arboretum library[5], the g++ 4.9.2 compiler and the Ubuntu 14.04 OS running on an Intel Core i7 2.67 GHz, 6 GB of RAM and HDD SATA III 7200 RPM. Additionally, the accuracy of the evaluated models was normalized in terms of the absolute error to the real cost of the searches in all experiments. The range and k-NN costs were collected following the range search procedure in [Ciaccia et al. 1997] and the k-NN search algorithm in [Hjaltason and Samet 2003], respectively.

[2]Available at: `www.ibge.gov.br`
[3]Available at: `www.archive.ics.uci.edu/ml`
[4]Available at: `www.commons.wikimedia.org/wiki/Category:Paintings`
[5]Available at: `www.bitbucket.org/gbdi/arboretum`

Table 1. Data sources and parameters employed in the experiments.

Source	Dim.	δ	$\lceil \mathcal{D} \rceil$	Card.	Description
CITIES	2	L_2	2	$5,507$	Geographical coordinates of 5507 Brazilian cities.
WINE	11	L_1	6	$6,497$	UCI labeled data source of wine's description.
LETTER	16	L_∞	9	$20,000$	Statistical moments and edge counts from all letters of the A–Z alphabet.
CANVAS	16	L_2	6	$3,879$	MPEG-7 Color Layout features from Wikimedia paintings.

5. Comparison of range queries cost estimates

We employed the 10-folds cross validation approach (90% of data for indexing, 10% of data for querying, cycling) for the evaluation of Stockpile and range queries. For each data source, we took the maximum distance between a pair of elements (max) and define 6 thresholds (ξ) for the range queries. Basically, we set ξ as a percentage of max from 5% to 30% in steps of 5%. Figure 3 shows the comparison between evaluated models. Each point in Figure 3(a) represents the absolute difference between the predicted and real number of scanned nodes when executing a range query. Stockpile achieved up to 36%, 31%, 41% and 54% more accurate disk accesses' predictions in comparison to model in [Ciaccia et al. 1998] regarding data sources CITIES, WINE, LETTER and CANVAS, respectively. Moreover, our approach reached up to 85%, 78%, 62% and 81% better predictions than model in [Baioco et al. 2007] for the same data sources. Despite the behaviors vary for increasing values of ξ, Stockpile was particularly dominant for smaller ξ values, which are the most meaningful ones for the majority of applications. Similar results were obtained regarding the predicted number of distance calculations, as shown in Figure 3(b). Again, Stockpile was up to 51%, 49%, 11% and 23% more accurate than model in [Ciaccia et al. 1998] for data sources CITIES, WINE, LETTER and CANVAS, respectively. Our model also was 64%, 73%, 25% and 69% better than model in [Baioco et al. 2007] in the same scenario.

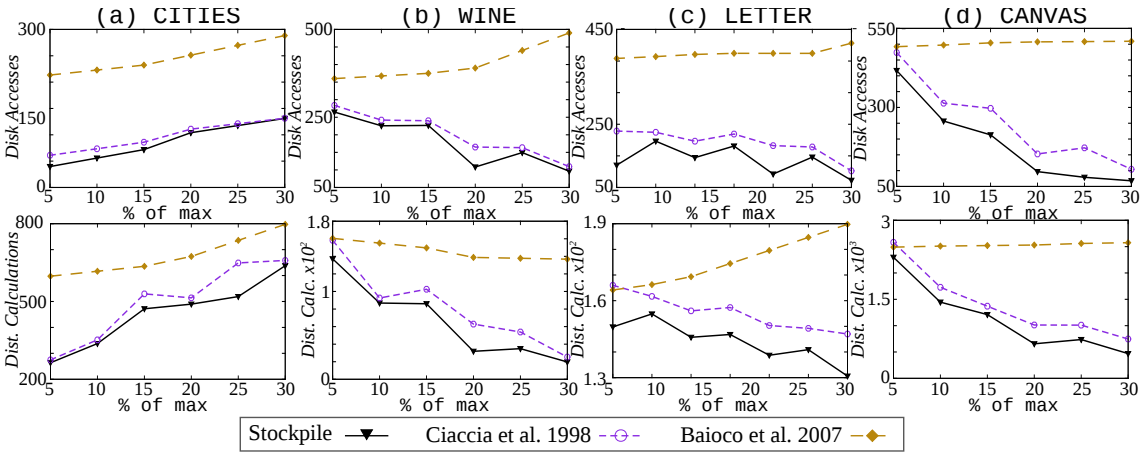


Figure 3. Comparison of models for range queries from 5% to 30% of max.

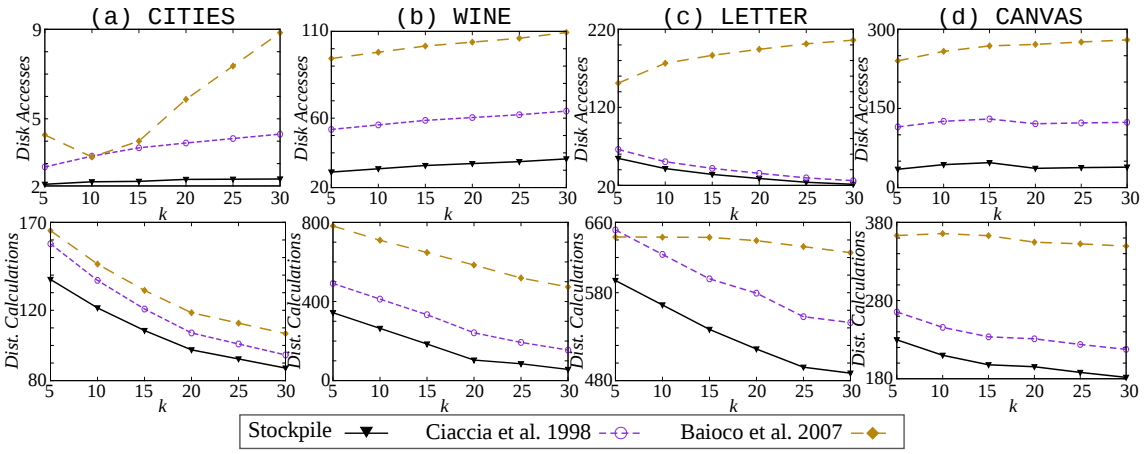


Figure 4. Comparison of models for k-NN queries from $k = 5$ to 30 in steps of 5.

6. Comparison of k-NN queries cost estimates

We also employed the 10-folds cross validation procedure for the evaluation of Stockpile accuracy in the task of predicting the cost of k-NN queries. In the experiment, we vary the parameter k from 5 to 50 in steps of 5 and evaluated our approach on terms of disk accesses and distance calculations. Figure 4(a) shows the Stockpile was up 46%, 43%, 19% and 54% more precise than model in [Ciaccia et al. 1998] for the estimation of disk accesses of k-NN queries on data sources `CITIES`, `WINE`, `LETTER` and `CANVAS`, respectively. Our model also was up to 73%, 69%, 85% and 83% better than model in [Baioco et al. 2007] in the same evaluation. Moreover, Figure 4(b) shows Stockpile was up to 13%, 30%, 13% and 12% more precise than the approach in [Ciaccia et al. 1998] in the estimation of distance calculations of k-NN queries on data sources `CITIES`, `WINE`, `LETTER` and `CANVAS`. Additionally, our approach was up to 20%, 80%, 26% and 45% better than model in [Baioco et al. 2007] in the same scenario. We highlight Stockpile was the most accurate in both range and k-NN queries. For instance, Stockpile not only outperformed its competitors for smaller radii (k-NN for $k \leq 30$), but it was also the best choice for larger values of radii (range for $\xi > 5\%$ of `max`). Accordingly, we point out the predictions drawn from pivot-based histograms are versatile enough to reach a good precision in comparison to estimates drawn from single pairwise distance distributions.

7. Conclusions

Cost modeling of similarity searches requires the proper handling of distance distributions. In this study, we proposed the Stockpile cost model that estimates the cost of similarity queries according to their locality by using pivot-based distance histograms. We compared Stockpile to strategies in [Ciaccia et al. 1998] and [Baioco et al. 2007] in the evaluation of similarity queries and the results showed our model was up to 54% and 85% more accurate than these two competitors, respectively. Future works include the evaluation of our model on other data sources and metric trees to verify the effects of the Stockpile parameters (e.g. number of pivots and optimistic/pessimistic estimates) regarding distinct ball-partitioning strategies.

Acknowledgments. The authors thank CNPq, CAPES, and FAPESP for their financial support.

References

Aly, A. M., Aref, W. G., and Ouzzani, M. (2015). Cost estimation of spatial k-nearest-neighbor operators. In *EDBT*, pages 457–468.

Baioco, G. B., Traina, A. J. M., and Traina Jr., C. (2007). MAMCost: Global and Local Estimates leading to Robust Cost Estimation. In *SSDBM*, pages 1:1–1:6.

Bedo, M. V. N., Kaster, D. S., Traina, A. J. M., and Traina Jr., C. (2015). CDH: a novel structure to boost k-nearest neighbor queries. In *SSDBM*, pages 36:1–36:6.

Bustos, B., Navarro, G., and Chávez, E. (2003). Pivot selection techniques for proximity searching in metric spaces. *PRL*, 24(14):2357–2366.

Ciaccia, P., Patella, M., and Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435.

Ciaccia, P., Patella, M., and Zezula, P. (1998). A Cost Model for Similarity Queries in Metric Spaces. In *PODS*, pages 59–68.

Hjaltason, G. R. and Samet, H. (2003). Index-driven similarity search in metric spaces. *TDS*, 28(1):517–580.

Ioannidis, Y. (2003). The history of histograms (abridged). In *VLDB*, pages 19–30.

Kaufman, L. and Rousseeuw, P. (1987). *Clustering by means of medoids*. North-Holland.

König, A. C. and Weikum, G. (2002). A framework for the physical design problem for data synopses. In *EDBT*, pages 627–645.

Korn, F., Pagel, B., and Faloutsos, C. (2001). On the 'Dimensionality Curse' and the 'Self-Similarity Blessing'. *TKDE*, 13(1):96–111.

Lokoč, J. (2010). *Tree-based indexing methods for similarity search in metric and non-metric spaces*. PhD thesis, Charles University, Ovocný trh 3-5, 116 36 Praha.

Lu, Y., Lu, J., Cong, G., Wu, W., and Shahabi, C. (2014). Efficient algorithms and cost models for reverse spatial-keyword kNN search. *TDS*, 39(2):13:1–13:46.

Navarro, G., Paredes, R., Reyes, N., and Bustos, C. (2017). An empirical evaluation of intrinsic dimension estimators. *IS*, 64:206–218.

Skopal, T., Pokorný, J., and Snásel, V. (2004). PM-tree: Pivoting Metric Tree for Similarity Search in Multimedia Databases. In *ADBIS*, pages 1 – 16.

Tao, Y., Zhang, J., P., D., and Mamoulis, N. (2004). An efficient model for optimization of kNN in low and medium dimensional spaces. *TKDE*, 16(10):1169–1184.

Tasan, M. and Özsoyoglu, Z. M. (2004). Improvements in distance-based indexing. In *SSDBM*, pages 161–170.

Traina Jr., C., Traina, A. J. M., Faloutsos, C., and Seeger, B. (2002). Fast indexing and visualization of metric data sets using slim-trees. *TKDE*, 14(2):244–260.

Vieira, M. R., Traina Jr., C., Traina, A. J. M., Arantes, A. S., and Faloutsos, C. (2007). Boosting kNN queries estimating suitable query radii. In *SSDBM*, pages 1:1 – 1:10.

Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2006). *Similarity Search - The Metric Space Approach*, volume 32. Kluwer.

paper:171434

Analyzing the Performance of Spatial Indices on Flash Memories using a Flash Simulator

Anderson Chaves Carniel[1], Tamires Brito da Silva[1], Kairo Luiz dos Santos Bonicenha[1], Ricardo Rodrigues Ciferri[2], Cristina Dutra de Aguiar Ciferri[1]

[1]Department of Computer Science – University of São Paulo
13.560-970 – São Carlos – SP – Brazil

{accarniel,tamiresbs,kairo.bonicenha}@usp.br, cdac@icmc.usp.br

[2]Department of Computer Science – Federal University of São Carlos
13.565-905 – São Carlos – SP – Brazil

ricardo@dc.ufscar.br

Abstract. *Spatial databases improve the spatial query processing by employing spatial indices. Due to the advantages of flash memories over magnetic disks like faster reads and writes, there is a special interest in managing spatial indices in these memories. However, many flash memories employ a Flash Translation Layer that does not provide open access to many important statistics, restricting the performance analysis of spatial indices. Flash simulators are promising tools to improve the performance analysis of spatial indices. In this paper, we analyze the performance of several distinct configurations of spatial indices by using a flash simulator and a real flash-based solid state drive. As a result, we provide correlations between these results to check the accuracy of a flash simulator in the spatial indexing context. In addition, we discuss the possibility of using a flash simulator as a first step for benchmarking spatial indices. That is, we check if the results provided by a flash simulator can be used to decrease the number of configurations to be evaluated in real flash memories, reducing the required time of an empirical analysis.*

1. Introduction

Advanced applications employ spatial database systems and Geographic Information Systems (GIS) for managing spatial information. For this purpose, spatial data types like *regions* are used [Güting 1994]. For instance, the area of a building is represented by a region. Commonly, applications issue *spatial queries* that return spatial objects. Typical queries involve *topological predicates*, e.g., "return all the buildings that *overlap* a given search area". To speed up the spatial query processing, spatial database systems and GIS make use of *spatial indices* [Gaede and Günther 1998], which discard spatial objects that certainly do not belong to the final result of a query. Examples of these indices are hierarchical structures like the R-tree and the R*-tree (see [Gaede and Günther 1998] for a survey). Since *Hard Disk Drivers* (HDDs) were the main storage devices used in previous decades, these indices assume that the spatial objects are stored in magnetic disks. We term them as *disk-based spatial indices*.

However, modern applications are increasingly requiring the use of newer storage devices like *flash memories* [Mittal and Vetter 2016, Brayner and Monteiro Filho 2016]. The reason is that, unlike HDDs, flash memories do not have mechanical parts and provide lower weight, size, power consumption, and read/write latency. An example of a storage device that employs flash memories is the flash-based Solid State Drive (SSD). These memories have intrinsic characteristics that introduce several system implications [Jung and Kandemir 2013]. For instance, the asymmetric cost between reads and writes, where a write requires more time and consumes more power than a read. Further, a write is only performed on empty pages of the flash memory; thus, an update is performed as an *erase-before-update* operation.

To improve the use of flash memories in current applications, the *Flash Translation Layer* (FTL) is employed [Chung et al. 2009]. It maps physical addresses of a flash memory into logical addresses and transforms reads and writes from application layers into a set of internal read, write, and erase operations. As a result, erase-before-updates can be avoided by applying an *out-of-place update* algorithm. Intuitively, this algorithm marks a page to be updated as invalid and writes its new content in another empty page. When a number of invalid pages are reached, a *garbage collector* erases the blocks with invalid pages, leading to erase-before-updates. Hence, a unique write performed on the application layer does not necessarily mean that only one write is performed on the flash memory. This also applies to reads since the data can be sliced into several physical pages and thus, a read from the application layer can lead to multiple reads on the flash memory.

To deal with the intrinsic characteristics of flash memories, *flash-aware spatial indices* have been proposed in the literature [Wu et al. 2003, Lv et al. 2011, Sarwat et al. 2013, Jin et al. 2015]. In general, these indices avoid random writes by using an *in-memory buffer* that stores modifications of the index. When this buffer is full, a *flushing operation* is performed by writing a set of modified nodes, called *flushing unit*. For instance, FAST [Sarwat et al. 2013] is a framework that transforms a disk-based spatial index (e.g., the R-tree) into a flash-aware spatial index (e.g., the FAST R-tree).

However, analyzing the impact of flash memories on spatial indexing, such as to measure the performance of a given spatial index, is a problematic task. The reason is that the FTL is physically integrated into the flash memory with license restrictions of the manufacturer. Hence, these experiments often measure the processing time of index operations since the FTL restricts the collection of the number of writes, reads, and erases actually performed on the storage device. Flash simulators are promising tools for evaluating the performance behavior of spatial indices on flash memories. They emulate the structure of flash memories in the main memory, implement FTL algorithms, and enable the collection of several statistics, such as the number of writes, reads, and erases. Some flash simulators have been proposed in the literature [Su et al. 2009, Kim et al. 2009, Dong et al. 2012]. Among them, FlashDBSim [Su et al. 2009] has been used in the (spatial) indexing context [Jin et al. 2015]. In addition, it is an open source, reusable, flexible and extensible simulator.

Despite the relevance of flash simulators, there are four important open questions that motivate this paper. They are:

1. Is a flash simulator capable of determining if a flash-aware spatial index provides better performance than a disk-based spatial index?

2. Does a spatial index that performs the best in a flash simulator also perform the best in a real flash memory, for example an SSD?
3. How can a flash simulator be used to evaluate the performance of spatial indices?
4. Is it possible to use a flash simulator as a filter of configurations of spatial indices to be evaluated in an SSD?

To answer these questions, in this paper we conduct an extensive performance evaluation of disk-based (the R-tree and the R*-tree) and flash-aware (the FAST R-tree and the FAST R*-tree) spatial indices by using two environments: Flash-DBSim and a real SSD. For all these compared indices, we varied several parameter values, resulting in different configurations of spatial indices.

We answer question 1 by analyzing the obtained results in Flash-DBSim. In this analysis, we aim to find out the performance gains of the flash-aware spatial indices over the disk-based spatial indices. Then, we correlate these performance gains with the obtained results in the real SSD. To answer question 2, we check if there is a configuration that shows the best results in both the environments. This helps us to measure the accuracy of the flash simulator. Based on the previous analyses, we answer question 3 by exploiting the applicability of flash simulators to evaluate the performance of spatial indices. By answering the questions 1 to 3, we lead to the answer of the most important question of this paper. To answer the last question, we analyze if only some configurations from a large set of configurations of spatial indices can be examined in a real SSD based on preliminary results obtained from experiments using a flash simulator.

The rest of this paper is organized as follows. Section 2 summarizes intrinsic characteristics of flash memories. Section 3 surveys related work. Section 4 details our performance evaluation. Section 5 answers our questions. Section 6 concludes the paper.

2. Flash Memories

Flash memories organize data in flash pages and flash blocks [Jung and Kandemir 2013]. A block consists of a fixed number of pages. Flash memories handle three types of operations: program (write), erase, and read. Read and write operations are performed at page level with asymmetric costs as a read requires much less time and consumes less power than a write; on the other hand, erase is performed at block level and is very expensive. In addition, flash memories do not provide an operation for updates. Thus, a three-step algorithm is performed to update the content of a page. First, the unchanged pages of its block are internally buffered. Second, the block is erased. Finally, the updated page and the buffered pages of the block are written back to the erased block. This operation is named *erase-before-update*. Moreover, flash memories have lower endurance capacity than HDDs [Mittal and Vetter 2016]. The endurance refers to the amount of write and erase operations that a block can receive before it is unavailable to support new operations.

The FTL component (see [Chung et al. 2009] for a survey) is employed to permit the use of flash memories in current computational environments. It allows a flash memory to be recognized by the operating system as a regular disk and provides only two operations for application layers: write and read. These operations are performed on logical page addresses mapped from physical page addresses of the flash memory [Mittal and Vetter 2016]. Each logical address of a page is marked either free, valid, or invalid. When a write is issued to the FTL, it writes the data into a free page and marks

this page as valid. If a write is performed on a valid page (i.e., an update), the FTL marks this page as invalid and writes the new data to a free page. This operation is known as an *out-of-place update* and avoids erase-before-updates. When a number of blocks with invalid pages are reached and storage space is required, the *garbage collector* is dispatched. It selects a set of blocks to be erased, stores the content of valid pages of these blocks in other pages, and erases the selected blocks. To improve the endurance of flash memories, a *wear-leveling* algorithm is responsible for controlling the number of erases that a block can receive.

3. Related Work

There are several approaches that conduct experimental evaluations of spatial indices on flash memories. We classify them according to the following characteristics: (i) the performance evaluation of spatial indices under different storage devices [Emrich et al. 2010, Carniel et al. 2016b], and (ii) the performance evaluation of new flash-aware spatial indices on flash memories [Wu et al. 2003, Lv et al. 2011, Sarwat et al. 2013, Jin et al. 2015].

The first group includes approaches that analyze the performance behavior of the spatial indexing on HDDs and SSDs. These approaches mainly compare the performance gains that SSDs provide over HDDs. For instance, [Carniel et al. 2016b] showed that an SSD provided reductions in the spatial query processing up to 96% over an HDD. However, flash simulators are not used in the experiments. This results in a limited analysis of an experiment since its focus is only on the elapsed time of a spatial index operation.

The second group consists of approaches that propose spatial indices specifically designed for SSDs. While straightforward adaptations of the R-tree are proposed [Wu et al. 2003, Lv et al. 2011], FAST [Sarwat et al. 2013] distinguishes itself by defining a generic framework to transform a disk-based spatial index into a flash-aware spatial index. In addition, the FOR-tree [Jin et al. 2015] is proposed to eliminate the split operations of the R-tree by allowing overflowed nodes. It uses a counter to decide when new primary nodes are created from an overflowed node, growing up the tree as needed. However, the counter for the root node is only incremented in search operations. Thus, instead of organizing nodes in a hierarchical structure, the construction of a FOR-tree index leads to an overflowed root node. Thus, it results in unacceptable processing times.

These approaches often conduct experiments measuring the performance of the proposed indices against disk-based spatial indices. However, the parameter values of these indices are not exploited. For instance, there is a lack of studies that measure the impact of the index page size (i.e., node size). In addition, the selectivity in spatial queries does not vary in workloads. Since the FTL is a black-box component of SSDs (see Section 2), Flash-DBSim was used in the experiments involving the FOR-tree. However, it is unclear if this simulator can be used as a filter for experiments beause there is a lack of performance comparisons between this simulator and a real SSD.

In this paper, we provide an extensive experimental evaluation for analyzing the applicability of flash memories to evaluate spatial indices by answering the four questions introduced in Section 1. Our experiments vary several parameter values and focus on the spatial query processing. As a result, we analyze the flash simulator as a tool for reducing the configurations to be evaluated in an empirical study.

4. Performance Evaluation

4.1. Configuration Setup

We used a real dataset containing 1,486,557 regions extracted from the OpenStreetMap[1]. This dataset represents the buildings of Brazil, such as hospitals, schools, universities, houses, stadiums, and so on.

We compared the configurations showed in Table 1. The disk-based spatial indices considered are: (i) the R-tree, with linear and quadratic splits, and (ii) the R*-tree, with the reinsertion policy (RP) of 30%. For these indices, we employed an in-memory buffer to cache the nodes in the highest levels of the index by using the least recently replacement (LRU) policy. The flash-aware spatial indices considered are the FAST-versions of the disk-based spatial indices: (i) the FAST R-tree, and (ii) the FAST R*-tree. For them, we applied the FAST* flushing policy (FP) because it reported the best results in [Sarwat et al. 2013]. The buffer size applied in all the configurations was equal to 512KB. In addition, we varied the size in bytes of an index node from 2KB to 16KB. Furthermore, we used the flushing unit size equal to 1 for the FAST-based spatial indices. We did not compare the FOR-tree because of the problems discussed in Section 3.

We executed two workloads: (i) index construction, and (ii) processing of intersection range queries (IRQ) [Gaede and Günther 1998]. The second workload includes the execution of three sets of 100 IRQs. These sets applied query windows corresponding respectively to 0.001%, 0.01%, and 0.1% of the area of the total extent of Brazil. Considering that the selectivity of a query is the ratio between the number of returned objects and the total objects, these sets of query windows form spatial queries with low, medium, and high selectivity, respectively.

These two workloads were executed as a sequence for each used configuration, i.e., we executed the construction of a spatial index and then the processing of the IRQs. To this end, we employed an extended version of FESTIval[2] [Carniel et al. 2016a], an open source PostgreSQL extension for benchmarking spatial indices. It was extended to integrate Flash-DBSim and to measure the performance of flash-aware spatial indices.

We used Flash-DBSim to simulate a flash memory of 512MB as depicted in Table 2. We employed this simulator because of its advantages (Section 1) and previous usage in the spatial indexing context. We collected the number of writes, reads, and erases required by each configuration to execute each workload (Section 4.2). We correlated these results by executing the same workloads on an SSD (Section 5). For this purpose, we used a local server equipped with an Intel® Core™ i7-4770 with a clock frequency of 3.40GHz, 32GB of main memory, and an SSD Kingston V300 with a capacity of 480GB[3]. In this environment, we performed the tests locally to avoid network latency and flushed the system cache after the execution of each configuration. Further, we executed each workload 10 times and calculated the elapsed time as follows. For the first workload, we collected the average elapsed time. For the second workload, we collected the average elapsed time to execute the IRQs of each kind of selectivity. We employed Ubuntu Server 14.04 64 bits, PostgreSQL 9.5, and PostGIS 2.2.0 in all the environments.

[1] `http://www.openstreetmap.org/`
[2] `http://gbd.dc.ufscar.br/festival/`
[3] `https://www.kingston.com/us/ssd/consumer/sv300s3`

Table 1. Configurations of the spatial indices used in the experiments.

Name	Spatial Index	Buffer Type	Parameters
Quadratic R-tree	R-tree	LRU	Split: Quadratic
Linear R-tree	R-tree	LRU	Split: Linear
R-tree*	R*-tree	LRU	RP: 30%
FAST Quadratic R-tree	FAST R-tree	FAST Buffer	Split: Quadratic; FP: FAST*
FAST Linear R-tree	FAST R-tree	FAST Buffer	Split: Linear; FP: FAST*
FAST R-tree*	FAST R*-tree	FAST Buffer	RP: 30%; FP: FAST*

Table 2. The emulated flash memory[4].

Latency			Size	
Read	Write	Erase	Block	Page
30µs	300µs	2,500µs	128KB (= 64 pages)	2KB

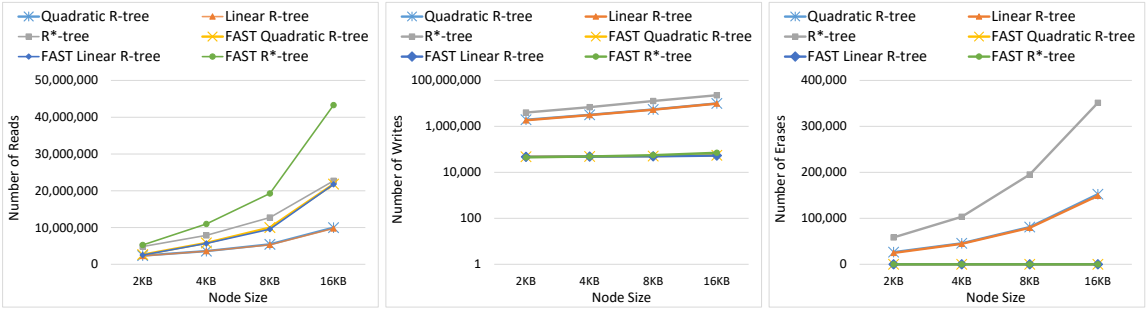
4.2. Execution on the Flash Simulator

4.2.1. Index Construction

Figure 1 depicts the results obtained for building the spatial indices in the flash simulator. Clearly, the FAST-based spatial indices required far fewer writes than disk-based spatial indices for all the page sizes. The reduction of the number of writes varied from 98% to 99%. This expressive result led to the non-use of erase operations. This is due to the buffer management of the FAST-based spatial indices, which reduced the number of writes compared to the traditional LRU buffer. On the other hand, the number of reads required by the FAST-based spatial indices was higher than the reads required by disk-based spatial indices. The reason is that the FAST buffer is only dedicated to storing the modifications of the index and is not used for reading purposes.

In general, the construction of the indices by using the page sizes equal to 2KB and 4KB required less operations than using other page sizes. The page size equal to 16KB provided the worst results since the index nodes have to be split into several pages of the flash simulator. In this case, each node of the index is stored on 8 pages of the flash simulator. In addition, because of the R*-tree reinsertion policy, the R*-tree generated the highest number of reads, writes, and erases.

[4]Values extracted from `https://www.digchip.com/datasheets/parts/datasheet/710/TC58NYG2S3ETA00.php`



(a) reads (b) writes (c) erases

Figure 1. Number of reads (a), writes (b), and erases (c) for building spatial indices in the flash simulator.

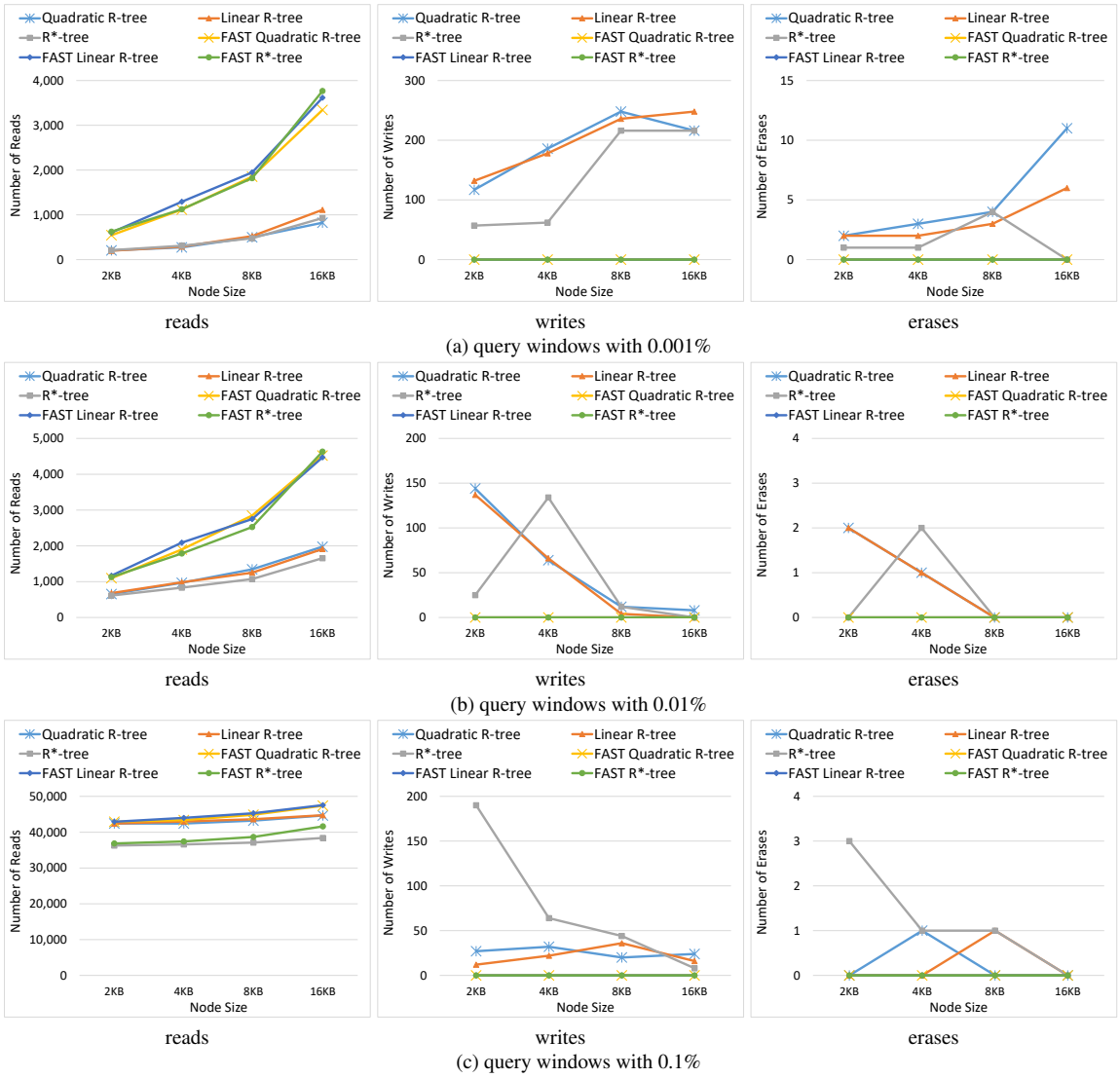


Figure 2. Results obtained for executing the IRQs in the flash simulator.

4.2.2. Spatial Query Processing

Figure 2 depicts the results obtained for processing of IRQs in the flash simulator. In most of the cases, the LRU buffer employed by the disk-based spatial indices required writes because of the LRU buffer that contained some modifications (from the index construction) to be applied. Hence, it also occasionally caused erases. The number of writes and erases decreased as the IRQs were executed because of the sequence of execution of workloads. On the other hand, the FAST-based spatial indices did not perform writes or erases because the employed buffer is not used for caching nodes without modifications. Thus, the elements stored in the buffer were not replaced.

With respect to query windows with 0.001% (Figure 2a), all the configurations showed the best results by using the page size equal to 2KB and increased the number of reads as the used node size also increased. The FAST-based spatial indices required much more reads than the corresponding disk-based spatial indices. This is a consequence of



Figure 3. Correlating the performance gains and losses of the FAST-based spatial indices over the corresponding disk-based spatial indices for building indices.

the buffer strategy employed by FAST, which only avoids a read from the storage device when the node to be retrieved is entirely stored in the buffer. The best performance results were obtained by using the Linear R-tree with the page size equal to 2KB.

With regard to query windows with 0.01% (Figure 2b), the number of writes and erases performed by the disk-based spatial indices decreased, compared to the query windows with 0.001%. Again, the FAST-based spatial indices required much more reads than the other configurations. In these IRQs, the best performance results were obtained by using the R*-tree with the page size equal to 2KB.

With respect to query windows with 0.1% (Figure 2c), the FAST-based spatial indices decreased the overhead of the number of reads compared to the execution of the previous query windows. In addition, the impact related to the size of nodes employed in the index was minimized, if compared to the execution of the query windows with 0.001%. Due to the high selectivity of the query windows with 0.1%, more objects have to be loaded in the main memory to process the topological predicates. Consequently, if a node has a greater capacity, the index will require fewer reads from the storage device. The best performance results were obtained by using the R*-tree using the page size equal to 2KB, which generated 4,754 nodes of which 1.07% of them were accessed.

5. The Performance Relation between the Flash Simulator and the Real SSD

To answer the questions in Section 1, we compare the results obtained in the flash simulator (Section 4.2) and in the SSD. Figures 3 and 4 depict the performance gains and losses of the FAST-based spatial indices over the corresponding disk-spatial indices (e.g., the FAST R*-tree and the R*-tree) for constructing indices and for processing spatial queries, respectively. A performance gain shows as a percentage value how much a configuration was better compared to another configuration. Conversely, a performance loss shows how much a configuration increased a compared value in relation to another configuration. Note that the percentage values of the number of reads, writes, and erases were calculated from the results of the flash simulator (Section 4.2), while the elapsed time percentage values were obtained from the results of the SSD.

In the following, we answer each question of Section 1 by firstly providing an analysis to serve as a foundation.

Analysis to Answer Question 1. For the index construction (Figure 3), in spite of the



Figure 4. Correlating the performance gains and losses of the FAST-based spatial indices over the disk-based spatial indices for processing the IRQs.

expressively positive results in the flash simulator, we did not obtain performance gains by using the FAST-based spatial indices in the SSD. In this case, the collected elapsed times only reported performance losses. The main reason was that these indices reduced the number of writes and erases, but at the same time increased the number of reads. In fact, the number of reads doubled for the node size equal to 16KB.

For the spatial query processing (Figure 4), the performance of the disk-based spatial indices was impacted by the writes performed in the SSD. Thus, although the FAST-based spatial indices have required more reads, it showed reductions in the processing time of IRQs in some cases. These reductions were more frequent to process the query windows with 0.1%, which showed the lower losses in the number of reads. In this case, the flash simulator was capable of determining that the FAST-based spatial indices would have a better performance compared to their corresponding disk-based indices.

Table 3. Best configurations obtained in the environments. In parentheses is showed the node size used in the configuration. The cells with '-' means that more than five configurations showed the best results.

Workload	Flash Simulator # of Reads	Flash Simulator # of Writes	Flash Simulator # of Erases	SSD Elapsed Time
Index Construction	Linear R-tree (2KB)	FAST R*-tree (2KB)	FAST-based indices	Linear R-tree (4KB)
Query Windows with 0.001%	Linear R-tree (2KB)	FAST-based indices	FAST-based indices	R*-tree (4KB)
Query Windows with 0.01%	R*-tree (2KB)	-	-	Linear R-tree (8KB)
Query Windows with 0.1%	R*-tree (2KB)	FAST-based indices	-	Linear R-tree (16KB)

Answer of Question 1. For the index construction, the answer is yes. Our experiments showed that when the number of reads in the flash simulator increased by more than 8%, a flash-aware spatial index tends to be inefficient in the SSD. For the spatial query processing, the answer is yes for the most of the cases. Our experiments showed that often when there is a low performance loss in the number of reads, the simulator is capable of determining that the spatial index will provide a good performance on an SSD.

Analysis to Answer Question 2. We use Table 3 to show the best configurations by comparing the number of reads, writes, erases obtained in the flash simulator and the elapsed time to process the workload in the SSD. For the index construction, our experiments showed that the reduction of writes and erases was not enough to provide the best performance results in the SSD. For the spatial query processing, we gathered different best configurations in the environments. But, we can note that the reduction of writes and erases associated to a small increase of required reads in the flash simulator was enough to guarantee the performance gains for the FAST-based indices in the SSD (Figure 2).

Answer of Question 2. In this case, the best configurations were not exactly the same in the environments. But, the flash simulator was able to indicate the following relation in the index construction. The configuration that generated the smallest number of reads showed also the best performance results in the SSD (e.g., the Linear R-tree, in spite of the difference in the used node size).

Analysis to Answer Questions 3 and 4. Our analysis is based on the spatial query processing since the spatial index should be constructed beforehand. For the IRQs with low selectivity (i.e., query windows with 0.001%), we did not need to evaluate the configurations using the node size equal to 8KB and 16KB. The main reason is that the use of smaller page sizes required the processing of fewer entries in the main memory, reducing the number of writes, reads, and erases. Consequently, it also reduced the processing time in the SSD. On the other hand, to efficiently process the IRQs with medium and high selectivity, the usage of small page sizes did not guarantee the better performance in the SSD. However, the flash simulator did not show this behavior.

Answers of Questions 3 and 4. The flash simulator showed to be applicable. Our experiments indicated that we could exclude several configurations to be evaluated in the SSD based on results obtained in the flash simulator.

In addition, we are able to avoid more configurations if an analysis has as goal to discover the best possible configurations to be employed on an SSD. Based on the number of reads obtained in the flash simulator, we can exclude configurations with the highest number of reads. For instance, we would avoid evaluating the FAST-based spatial indices

since they failed to reduce the number of reads compared to the disk-based spatial indices. Moreover, if the index construction is a critical step in the goal of the analysis, the flash simulator showed even more useful to exclude configurations to be evaluated in the SSD.

6. Conclusions and Future Work

This paper provides an extensive experimental evaluation in order to analyze the use of a flash simulator as a first step in the performance evaluation of spatial indices in flash memories. We considered the disk-based spatial indices the R-tree and the R*-tree because of their positive characteristics reported in the literature. In addition, we also considered their flash-aware versions, that is, the FAST R-tree and the FAST R*-tree.

As main conclusions, we can cite the following performance behaviors that lead us to answer the most important question of this paper. That is, the question that studies the applicability of a flash simulator as a filter of configurations to be evaluated in an SSD. Firstly, for building spatial indices, our experiments showed that the flash simulator is capable of avoiding configurations to be executed in the SSD based on the number of reads, writes, and erases. The indices that showed the greater number of operations in the simulator also showed poor performance in the SSD. Secondly, for spatial query processing, the number of reads obtained in the flash simulator can be used as a basis to exclude configurations with the highest number of reads. The reason is that these configurations showed the worst performance results in the SSD. We can conclude that a flash simulator is an interesting filter to determine configurations to be evaluated in an SSD. Consequently, the time required to perform empirical analysis can be decreased.

Future work will deal with the execution of other workloads mixing insertions, updates, and queries. In addition, our plan is to conduct the same evaluation in other flash simulators to compare the accuracy among them, as well as take into account other spatial indices, such as the Hilbert R-tree [Gaede and Günther 1998] and eFIND-based spatial indices [Carniel et al. 2017].

Acknowledgments. This work has been supported by CAPES, CNPq, and FAPESP. The first author has been supported by the grant #2015/26687-8, FAPESP. The second and third authors have been supported by the PUB/USP. The fourth author has been supported by the grant #311868/2015-0, CNPq.

References

Brayner, A. and Monteiro Filho, J. M. (2016). Hardware-aware database systems: A new era for database technology is coming - vision paper. In *Brazilian Symp. on Databases*, pages 187–192.

Carniel, A. C., Ciferri, R. R., and Ciferri, C. D. A. (2016a). Experimental evaluation of spatial indices with FESTIval. In *Brazilian Symp. on Databases - Demos*, pages 123–128.

Carniel, A. C., Ciferri, R. R., and Ciferri, C. D. A. (2016b). The performance relation of spatial indexing on hard disk drives and solid state drives. In *Brazilian Symp. on GeoInformatics*, pages 263–274.

Carniel, A. C., Ciferri, R. R., and Ciferri, C. D. A. (2017). A generic and efficient framework for spatial indexing on flash-based solid state drives. In *European Conference on Advances in Databases and Information Systems*.

Chung, T.-S., Park, D.-J., Park, S., Lee, D.-H., Lee, S.-W., and Song, H.-J. (2009). A survey of flash translation layer. *Journal of Systems Architecture: the EUROMICRO Journal*, 55(5-6):332–343.

Dong, X., Xu, C., Xie, Y., and Jouppi, N. (2012). NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):994–1007.

Emrich, T., Graf, F., Kriegel, H.-P., Schubert, M., and Thoma, M. (2010). On the impact of flash SSDs on spatial indexing. In *Int. Workshop on Data Management on New Hardware*, pages 3–8.

Gaede, V. and Günther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231.

Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399.

Jin, P., Xie, X., Wang, N., and Yue, L. (2015). Optimizing R-tree for flash memory. *Expert Systems with Applications*, 42(10):4676–4686.

Jung, M. and Kandemir, M. (2013). Revisiting widely held SSD expectations and rethinking system-level implications. In *ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems*, pages 203–216.

Kim, Y., Tauras, B., Gupta, A., and Urgaonkar, B. (2009). FlashSim: A simulator for NAND flash-based solid-state drives. In *Int. Conf. on Advances in System Simulation*, pages 125–131.

Lv, Y., Li, J., Cui, B., and Chen, X. (2011). Log-Compact R-tree: An efficient spatial index for SSD. In *Int. Conf. on Database Systems for Advanced Applications*, pages 202–213.

Mittal, S. and Vetter, J. S. (2016). A survey of software techniques for using non-volatile memories for storage and main memory systems. *IEEE Trans. on Parallel and Distributed Systems*, 27(5):1537–1550.

Sarwat, M., Mokbel, M. F., Zhou, X., and Nath, S. (2013). Generic and efficient framework for search trees on flash memory storage systems. *GeoInformatica*, 17(3):417–448.

Su, X., Jin, P., Xiang, X., Cui, K., and Yue, L. (2009). Flash-DBSim: A simulation tool for evaluating flash-based database algorithms. In *IEEE Int. Conf. on Computer Science and Information Technology*, pages 185–189.

Wu, C.-H., Chang, L.-P., and Kuo, T.-W. (2003). An efficient R-tree implementation over flash-memory storage systems. In *ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, pages 17–24.

paper:171397

Explorando arquiteturas multi-core para processamento eficiente de consultas em sistemas de gerência de Big Data

Frank W. R. da Silva[†,¥], Victor T. de Almeida[†,§], Vanessa Braganholo[†]
[†]Instituto de Computação, Universidade Federal Fluminense (UFF)
[¥]Universidade do Estado de Mato Grosso (UNEMAT)
[§]Petrobras S.A.
{frankwrs, valmeida, vanessa}@ic.uff.br

Abstract. *Big Data Management Systems usually manage each machine as one node in parallel query processing pipeline. In multi-core architectures, they leave several processor cores aside that could contribute to speed-up query processing. In this context, this paper explores the use of all available processor cores, assessing the query processing performance in several scenarios. In particular, we use the concept of worker nodes (which are allocated in cores without disk access) and data nodes (which are allocated in cores with disk access) in the same machine using the MyriaX engine as a base platform that supports this concept. We evaluate several cluster configurations varying the amount of data and worker nodes to process two types of queries (self-join and triangle) in a Twitter dataset. The results show that increasing the I/O parallelism in terms of data nodes is not always the most effective strategy. This reinforces the idea of using worker nodes in the query processing pipeline. In the best scenario, we achieved a speed-up of 2.92 by simply adding worker nodes in the available processing cores.*

Resumo. *Sistemas de Gerência de Big Data, em geral, gerenciam cada máquina como um nó dentro do pipeline de processamento paralelo de consultas, deixando de lado núcleos de processador que poderiam contribuir para acelerar o processamento das consultas. Neste contexto, este artigo explora o uso de todos os núcleos de processador disponíveis, avaliando o desempenho de consultas em diversos cenários. Para isso, usamos o conceito de worker nodes (alocados a núcleos que não possuem acesso a disco) e data nodes (alocados em núcleos com acesso a disco) em uma mesma máquina, tendo como plataforma base o mecanismo MyriaX, que suporta este conceito. Avaliamos diversas configurações variando a quantidade de worker nodes e data nodes para dois tipos de consultas (auto-junção e triângulos) em dados do Twitter. Os resultados mostram que aumentar o paralelismo de I/O em termos de data nodes nem sempre é a estratégia mais eficaz, o que reforça a ideia da utilização de worker nodes no pipeline de processamento de consultas. No melhor caso, obtivemos aceleração de 2,92x com a simples adição de worker nodes em núcleos de processamento disponíveis.*

1. Introdução

Sistemas de Gerência de Big Data têm sido usados para processamento de consultas analíticas [DeWitt and Gray, 1992]. Estes sistemas, em sua maioria, utilizam *clusters* para processamento massivamente paralelo de consultas. As arquiteturas utilizadas por eles variam. Alguns exigem que cada máquina tenha um disco acoplado [Abouzeid et al.,

2009; Alsubaiee et al., 2012; Bittorf et al., 2015; Das et al., 2013; Isard et al., 2007; Malewicz et al., 2010; Wang et al., 2017]. Outros utilizam de uma base de dados compartilhada que todos os nós acessam de forma concorrente [Dageville et al., 2016]. Há também sistemas que exploram a tecnologia *multi-core,* alocando vários nós em uma mesma máquina, mas com o recurso de armazenamento compartilhado ou fatiado entre os nós [Gupta et al., 2015; Warneke and Kao, 2009]. No entanto, nenhuma destas estratégias explora totalmente os recursos disponíveis de armazenamento e processamento de forma independente em uma mesma máquina.

Com a finalidade de melhorar a eficiência do processamento paralelo de consultas em sistemas de gerência de Big Data, este artigo explora todos os recursos de processamento (núcleos dos processadores) e armazenamento disponíveis por máquina. Isto é feito por meio do uso de *worker nodes*, que realizam processamento e não armazenam dados, e *data nodes*, que processam e armazenam dados, alocados em uma mesma máquina.

Para avaliar o impacto disso no desempenho de consultas, utilizamos o mecanismo de execução MyriaX, componente do sistema de gerência de Big Data Myria [Wang et al., 2017]. Esta escolha foi motivada pelo fato desse sistema permitir o uso de *worker nodes* e *data nodes* de forma independente. Deste modo, o MyriaX viabiliza a alocação de *data nodes* e *worker nodes* numa mesma máquina, apesar disto nunca ter sido explorado pelo Myria. Em nossa avaliação experimental, utilizamos duas consultas custosas em diversos cenários diferentes. Os resultados mostram que aumentar o paralelismo de I/O em termos de *data nodes* nem sempre é a estratégia mais eficaz, o que reforça a ideia da utilização de *worker nodes* no *pipeline* de processamento de consultas. No melhor caso, obtivemos aceleração de 2,92x pela simples adição de *worker nodes* em núcleos de processamento disponíveis. Além disso, mostramos também que a *cache* tem pouca influência no tempo de processamento das consultas.

As seções a seguir apresentam os trabalhos relacionados (Seção 2), uma visão geral do mecanismo MyriaX (Seção 3), o plano e execução dos experimentos (Seção 4), a avaliação dos resultados (Seção 5) e a conclusão (Seção 6).

2. Trabalhos Relacionados

Clusters de máquinas de propósito geral são utilizados por sistemas de gerência de Big Data, em sua maioria, para processamento massivamente paralelo de consultas. Alguns exigem que cada máquina tenha um disco acoplado, tais como ElasTras [Das et al., 2013], Impala [Bittorf et al., 2015], HadoopDB [Abouzeid et al., 2009], Apache Spark[1], Pregel [Malewicz et al., 2010], Dryad [Isard et al., 2007], Asterix [Alsubaiee et al., 2012], Presto[2] e Vertica[3]. Já o SnowFlake [Dageville et al., 2016] utiliza-se de uma base de dados compartilhada que todos os nós acessam de forma concorrente. Outros sistemas exploram a tecnologia *multi-core* alocando vários nós em uma mesma máquina, tais como Nephele [Warneke and Kao, 2009], Amazon Redshift [Gupta et al., 2015] e Greenplum[4]. Contudo, em tais sistemas o recurso de armazenamento é fatiado ou compartilhado entre os nós, o que implica em concorrência no acesso ao disco.

[1] https://spark.apache.org/
[2] http://prestodb.io/
[3] https://www.vertica.com/
[4] http://greenplum.org/

O serviço Myria fornece um *middleware* compatível com diversos mecanismos de execução de consulta, dentre eles MyriaX, Radish, SciDB, Spark e SPARQL [Wang et al., 2017]. O MyriaX se destaca por utilizar o conceito de *worker nodes* e *data nodes*. D*ata nodes* possuem acesso a disco, enquanto *worker nodes* apenas participam do pipeline de execução de consultas. É possível direcionar o processamento de consultas para *data nodes* e *worker nodes* específicos. Porém, atualmente, o MyriaX utiliza um único tipo de nó por máquina do cluster, independente de quantos núcleos a máquina tenha disponíveis.

De fato, a maioria destes sistemas trata cada máquina como sendo um único nó. Mesmo que cada máquina possua diversos processadores, cada um com diversos núcleos, e vários discos de dados, estes são normalmente gerenciados pelo sistema como um único recurso. No entanto, sabe-se que o uso de mais núcleos implica num melhor potencial de escalabilidade para operadores relacionais, tais como junções [Kim et al., 2009]. Neste contexto, em nossa pesquisa bibliográfica, não encontramos nenhuma estratégia que explore totalmente os recursos disponíveis de armazenamento e processamento de forma independente em uma mesma máquina. De fato, as estratégias existentes são incapazes de explorar núcleos ociosos de processadores e discos de dados adicionais para atuar no processamento de operadores da consulta.

3. Mecanismo de Execução de Consulta MyriaX

O MyriaX é um mecanismo de execução de consulta relacional *shared-nothing* e paralelo [Wang et al., 2017]. Sua arquitetura é composta por um *master node*, *worker nodes* e *data nodes* [Halperin et al., 2014]. O *master node* realiza o controle dos demais nós. A diferenciação entre *data node* e *worker node* acontece na atribuição de tarefas para cada nó. Nós que recebem operações de leitura e/ou escrita de dados atuam como *data nodes*. Por outro lado, nós que não recebem tarefas de leitura e/ou escrita de dados atuam como *worker nodes*. *Data nodes* podem também atuar como *worker nodes* após finalizar as operações de leitura e/ou escrita de dados. Desse modo, o MyriaX viabiliza a alocação de *data nodes* e *worker nodes* em uma mesma máquina, apesar disto nunca ter sido explorado pelo Myria A Figura 1 ilustra uma implantação deste mecanismo com nove máquinas, sendo quatro *data nodes* e quatro *worker nodes*, além do *master node*.

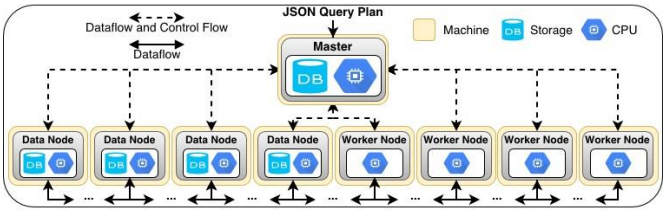


Figura 1. Arquitetura MyriaX com nove máquinas

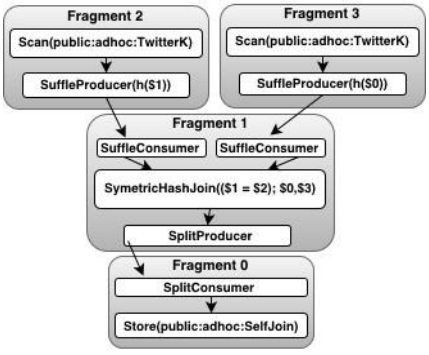


Figura 2. Plano de consulta de auto-junção

No MyriaX, cada máquina que participa do processamento de consultas tem apenas um tipo de nó (*worker node* ou *data node*). Contudo, as tecnologias atuais permitem processamento paralelo em processadores que contêm vários núcleos. Em

teoria, isto permite que uma mesma máquina tenha mais de um tipo de nó associado à um mesmo processador. Por exemplo, uma máquina que tenha um processador capaz de processar oito tarefas em paralelo (oito núcleos físicos) poderia ser utilizada com um *data node* e sete *worker nodes*. É importante notar que também é possível conseguir paralelismo de I/O, desde de que haja mais de uma controladora de disco por máquina.

A Figura 2 apresenta um plano de consulta para processamento paralelo de auto-junção (*self-join*) para o MyriaX [Mishra and Eich, 1992]. Neste plano, os Fragmentos 2 e 3 realizam a operação de leitura dos dados *Scan(public:adhoc:TwitterK)* (executados por *data nodes*), calculam o valor *hash* do atributo de junção com a operação *SuffleProducer(h($1))* e os encaminham para os nós da rede que irão efetuar a junção. A operação de *Scan()* faz leitura da tabela *public:adhoc:TwitterK*. O Fragmento 1 (executado por *worker nodes* e/ou *data nodes* livres) recebe os dados enviados pelos Fragmentos 2 e 3 através de uma porta específica, preenchendo um *buffer* e disparando a execução da junção, utilizando o algoritmo de *HashJoin* [Schneider and DeWitt, 1989] assim que este *buffer* fica cheio, de uma maneira não bloqueante. Os resultados obtidos são enviados para e salvos no Fragmento 0.

O MyriaX permite que se especifique quais nós cada fragmento irá utilizar para realizar suas operações. Aqueles que realizam operação de leitura de dados devem ser direcionados exclusivamente para *data nodes*. Aqueles que realizam processamento de operadores de álgebra relacional, por sua vez, podem ser direcionados para *worker nodes* e/ou *data nodes*. O plano de consulta é gerado e passado ao MyriaX através de algum otimizador de consultas, tal como o RACO [Wang et al., 2017]. A execução acontece em paralelo, com múltiplos fragmentos independentes ou encadeados (*pipelined*) entre nós, no caso comum do paralelismo de dados na avaliação da consulta. A partir da descrição do plano da consulta (em formato JSON), o MyriaX atribui cada fragmento ao seu conjunto de nós, tal como foram especificados, para processamento dos operadores. Cada fragmento inicia seu processamento a partir do momento que o anterior conclui parte de sua tarefa, em um *pipeline* de operações de álgebra relacional. Caso não haja fragmentos anteriores, como no caso de operações de leitura de dados, o *master node* inicia a execução do fragmento imediatamente.

No MyriaX, é possível associar vários nós ao mesmo processador, apesar disso não ser utilizado. Na próxima seção, apresentamos experimentos que exploram o uso de vários *worker nodes* na mesma máquina e comparamos o desempenho com o comportamento padrão do MyriaX.

4. Plano e Execução do Experimento

Como mencionado na introdução, máquinas utilizadas no processamento paralelo de consultas podem conter recursos subutilizados, como núcleos de processadores com e sem discos acoplados, que podem ser utilizados como *worker nodes* e *data nodes* adicionais no processamento de consultas. Com base nisto, submetemos o MyriaX a uma coleção de experimentos em *cluster* aplicando diversas configurações e alocações de *data nodes* e *worker nodes*. O *cluster* utilizado é composto de 42 máquinas, cada uma com 2 processadores Intel Xeon *quadcore* (8 núcles), 16 GB memória de RAM e 160 GB de disco. Todas as máquinas são interligadas com *switch Gigabit Ethernet*. O sistema operacional é o Red Hat Enterprise Linux Server versão 5.3.

Para validar a hipótese de que *"é possível diminuir o tempo de processamento de consultas através da adição de worker nodes em núcleos ociosos de processadores*", realizamos diversos experimentos. Nesses experimentos, utilizamos um algoritmo que, a partir de uma quantidade de *worker nodes* e *data nodes*, gera vários cenários de distribuição e alocação de *data nodes* e *worker nodes* em máquinas do *cluster*. O script realiza 5 rodadas de consultas para cada cenário gerado. Cada rodada de consultas é executada para todos os cenários gerados antes de iniciar a rodada seguinte, ao invés de todas as 5 rodadas de consultas serem executadas de forma sequencial para cada cenário. Com esta lógica, evita-se que um cenário seja totalmente prejudicado por alguma atividade de manutenção do sistema operacional que fuja ao nosso controle durante a execução das consultas. Além disso, para avaliar o impacto de uso de memória *cache*, cada rodada é constituída de 2 execuções sequenciais da consulta. Os tempos das 5 primeiras execuções de cada consulta são usados para avaliação nos cenários sem memória *cache*. Já os tempos das 5 segundas execuções são usados para avaliar o impacto de memória *cache*.

Tabela 1. Possíveis cenários para oito máquinas com oito núcleos cada

Cenário	# máquinas	# *data nodes*	# *worker nodes*	# nós de processamento
m:2_dn:2_wn:0	2	2	0	2
m:2_dn:2_wn:2	2	2	2	4
m:2_dn:2_wn:6	2	2	6	8
m:2_dn:2_wn:14	2	2	14	16
m:4_dn:2_wn:30	4	2	30	32
m:8_dn:2_wn:62	8	2	62	64
m:4_dn:2_wn:2	4	2	2	4
m:4_dn:4_wn:0	4	4	0	4
m:4_dn:4_wn:4	4	4	4	8
m:4_dn:4_wn:12	4	4	12	16
m:4_dn:4_wn:28	4	4	28	32
m:8_dn:4_wn:60	8	4	60	64
m:8_dn:2_wn:6	8	2	6	8
m:8_dn:4_wn:4	8	4	4	8
m:8_dn:8_wn:0	8	8	0	8
m:8_dn:8_wn:8	8	8	8	16
m:8_dn:8_wn:24	8	8	24	32
m:8_dn:8_wn:56	8	8	56	64

Cada cenário é composto por uma quantidade de máquinas (*m*), *data nodes* (*dn*) e *worker nodes* (*wn*). A quantidade de *worker nodes*, quando maior que zero, é dividida igualmente para cada máquina do cenário. A heurística para criar os cenários se baseou em valores de potência de 2 (iniciando em 2) para a quantidade de nós em cada implantação do serviço Myria em *cluster*, tendo como limite a quantidade máxima de núcleos de processadores disponibilizados (nesse experimento foi um processador com 8 núcleos). Em cada implantação o algoritmo determina a quantidade para cada tipo de nó. Para *data nodes*, também foram utilizados valores de potência de 2 (iniciando em 2), tendo como limite a quantidade de máquinas, pois cada uma conta com apenas uma controladora de disco. Os demais nós são alocados como *worker nodes*. Vale ressaltar que nos experimentos, *data nodes* também atuam como *worker nodes* no processamento da consulta. A Tabela 1 apresenta os cenários possíveis para implantações em 9 máquinas, destacando os cenários padrão do MyriaX. Nesta tabela, a coluna "# nós de processamento" contabiliza a quantidade de nós atuando no *pipeline* de processamento (*data nodes* + *worker nodes*). Em todos os cenários, uma das máquinas é reservada para atuar como *master node* e, portanto, apenas as demais 8 máquinas são consideradas pelos

cenários. Cabe ressaltar que os dados são particionados entre os *data nodes* usando a estratégia de *round-robin* [Mehta and DeWitt, 1997].

A Figura 1 ilustra o possível cenário *m:8_dn:4_wn:4*. Este cenário oferece um maior particionamento da tabela de dados, pois utiliza 4 *data nodes* e 4 *worker nodes*, armazenando uma menor quantidade de dados em cada *data node*, em relação ao cenário *m:2_dn:2_wn:6*, ilustrado na Figura 3, que conta com 2 *data nodes* e 6 *worker nodes*, além do *master node*. Este último cenário, por sua vez, sofre menor influência de tráfego de rede na transmissão de dados entre nós, pois contém 2 máquinas com 1 *data node* e 3 *worker nodes* cada, aumentando assim o paralelismo intra-máquina. Ambos os cenários, após concluídas as operações de leitura de dados, terão 8 nós (2 *dn* + 6 *wn*) disponíveis para o processamento paralelo da consulta.

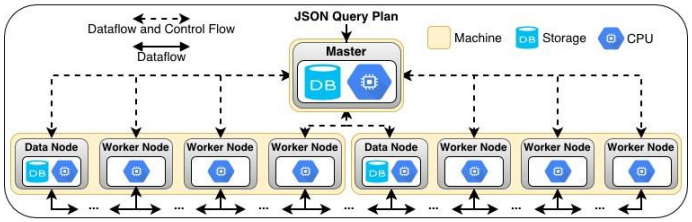


Figura 3. Arquitetura MyriaX com três máquinas

A base de dados utilizada para o experimento foi a do Twitter, que contém uma tabela com duas colunas (*follower* e *followee*) com valores de identificadores de usuários e representa relações entre usuários seguidores (*follower*) e seguidos (*followee*). Esta base contém 4.532.185 tuplas e foi escolhida por ser uma base de dados real com grande volume de dados. Processar certos tipos de consultas (utilizadas nesse experimento) nesse tipo de base é extremamente custoso.

Foram utilizadas duas consultas referentes à base de dados do Twitter. A primeira consulta (C1) realiza auto-junção entre as colunas citadas da base, com a finalidade de identificar relações entre usuários onde ambos seguem um ao outro. O plano desta consulta (Figura 2) foi apresentado e detalhado na Seção 2. A segunda consulta (C2) conta com uma operação de junção a mais e identifica triângulos entre usuários. Um triângulo se forma quando um usuário *A* segue um usuário *B*, que por sua vez segue um usuário *C*, que segue o usuário *A*. De fato, esta é uma consulta que ainda apresenta desafios para a comunidade científica quando os dados não cabem na memória [Hu et al., 2013]. Na base de dados utilizada no experimento, a primeira e segunda junções retornam 2.045.216.395 e 89.084.893 tuplas, respectivamente. Durante as submissões das consultas, o algoritmo captura o tempo que cada consulta leva para executar e calcula a média, eliminando as consultas com maior e menor tempo, para cada cenário. Os resultados obtidos são apresentados e avaliados na próxima seção.

5. Resultados e Avaliação

Nos resultados experimentais apresentados nesta seção, cenários padrão do MyriaX estão nomeados como *Baseline* e cenários que se baseiam na hipótese apresentada neste artigo são nomeados *Avaliação*. Além disso, também são apresentados e avaliados os resultados para experimentos com influência de memória *cache*. Todos os tempos discutidos nessa seção são medidos em segundos.

Avaliação do Impacto de Memória *Cache*. A Figura 4 apresenta os resultados que incluem tempo médio das consultas C1 e C2, com influência de memória *cache*. Os

cenários com maior influência de memória *cache* para a consulta C1 foram *m:4_dn:4_wn:28* e *m:8_dn:4_wn:60* com ganho de aproximadamente 3 segundos, e para a consulta C2 foi *m:4_dn:4_wn:28* com ganho de aproximadamente 15 segundos. Os demais cenários apresentam menor influência de memória *cache* no processamento da consulta. Os resultados mostram que a *cache* tem pouca influência no tempo de processamento das consultas. Por esse motivo, nos demais experimentos, usamos dados das execuções sem influência de memória *cache*.

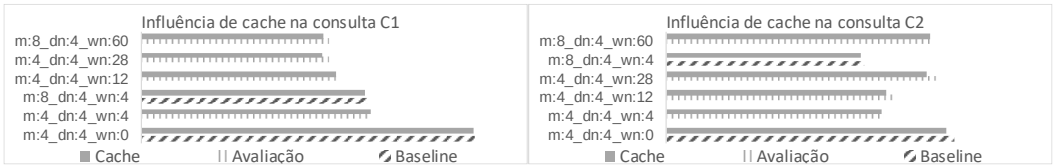


Figura 4. Tempo médio para C1 e C2 com influência de memória *cache*

Avaliação do impacto de *worker nodes* sem paralelismo de dados. Para avaliar o impacto de *worker nodes* sem paralelismo de dados, realizamos um experimento com apenas 1 máquina (Figura 5), ou seja, sem particionamento de dados e tráfego de rede (*overhead*), utilizando no máximo 7 *worker nodes*, totalizando 8 nós de processamento, explorando todos os núcleos de processamento disponibilizados. O cenário *Baseline m:1_dn:1_wn:0* apresentou o maior tempo médio para ambas as consultas. Isto porque o único *data node* desse cenário processou todos os operadores das consultas de forma sequencial. Os cenários *Avaliação* de ambas as consultas, por outro lado, utilizaram de paralelismo intra-máquina durante operações de junção devido ao uso de *worker nodes* e obtiveram menor tempo médio de processamento, com destaque para o cenário *m:1_dn:1_wn:7* que explora todo o recurso de núcleos de processamento. O tempo nesse cenário implicou aceleração de 3,06x para a consulta C1 e 3,44x para a consulta C2, em relação ao cenário *Baseline m:1_dn:1_wn:0*, mesmo sem paralelismo de I/O.

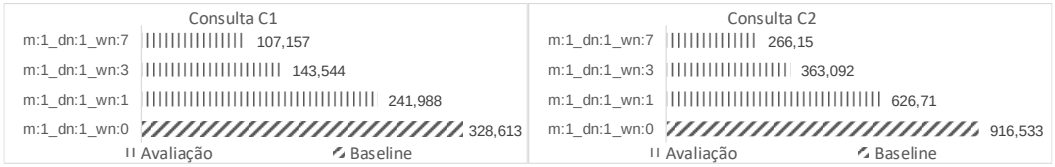


Figura 5. Tempo médio para C1 e C2 referente aos cenários de 1 máquina

Consulta C1 (auto-junção). A Figura 6 apresenta os resultados para a consulta C1 com uso de mais de um *data node*. Logo, há particionamento e paralelismo na leitura de dados. O cenário *Baseline m:2_dn:2_wn:0*, apresentou aceleração de aproximadamente 1,24x referente ao cenário *Baseline m:1_dn:1_wn:0* da Figura 5, que utiliza uma máquina.

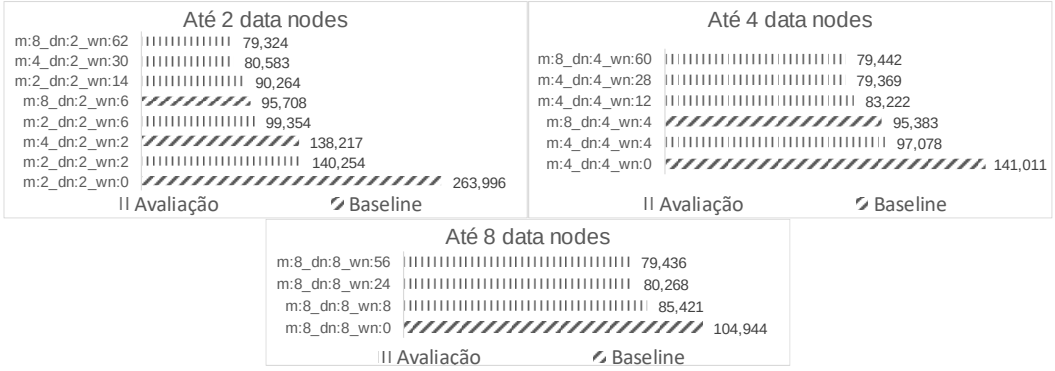


Figura 6. Tempo médio da consulta C1 com até 8 máquinas

Os resultados da Figura 6 com até 2 *data nodes* mostram que a consulta C1 teve melhor desempenho quando submetida em cenários *Avaliação*, com destaque para o cenário *m:8_dn:2_wn:62*, que explora totalmente os recursos disponibilizados utilizando todas as máquinas do cluster. Este cenário apresentou o melhor desempenho com aceleração de aproximadamente 3,33x em relação ao cenário *Baseline m:2_dn:2_wn:0*, que apresentou o maior tempo médio. O resultado deste cenário (*Baseline*) é influenciado pela ausência de *worker nodes* que levam os *data nodes* a executar os demais operadores da consulta de forma sequencial. Por outro lado, os cenários com *worker nodes* iniciam a execução do operador de junção antes da conclusão das operações de leitura de dados, pois *data nodes* encaminham partes dos resultados aos *worker nodes* durante esta operação em *pipeline*. É importante notar que, para cenários com até 2 máquinas, a simples adição de *worker nodes* implicou em aceleração de até 2,92x quando se explora todos os núcleos de processamento disponíveis. Os cenários *Avaliação m:8_dn:2_wn:62* e *m:4_dn:2_wn:30* apresentam ainda menor tempo médio, isto porque aumenta o paralelismo de I/O e os *data nodes* passam a processar menores quantidades de dados.

Nos cenários com até 4 *data nodes* da Figura 6, os cenários *Avaliação* apresentam melhor desempenho em relação aos cenários *Baseline*. Os cenários com melhor desempenho, *m:4_dn:4_wn:28* e *m:8_dn:4_wn:60*, apresentaram aceleração de aproximadamente 1,77x em relação ao cenário *Baseline m:4_dn:4_wn:0*, que apresentou o maior tempo. Tais cenários apresentaram resultado semelhante pela grande quantidade de *worker nodes*, que processam menores quantidades de dados. Os cenários *Baseline* apresentam resultado semelhante, onde aquele que não tem *worker nodes* (*m:4_dn:4_wn:0*) utilizou aproximadamente 3s a mais, justificado pela execução sequencial dos operadores da consulta. Os cenários *Avaliação* apresentam maior quantidade de *worker nodes* para processamento do operador de junção e também maior paralelismo intra-máquina, ou seja, utilizam até 7 *worker nodes* que não são influenciados pelo tráfego de rede quando recebem dados a partir do *data node* alocado na mesma máquina. Neste sentido, é importante notar que, para cenários com até 4 máquinas, a simples adição de *worker nodes* implicou em aceleração de até 1,77x quando se explora todo o recurso de núcleos de processamento disponíveis.

Nos cenários com até 8 *data nodes* da Figura 6, os cenários *Avaliação* também apresentam melhor desempenho em relação aos *Baseline*, porém de forma menos acentuada. O cenário *m:8_dn:8_wn:56*, que explora todo o recurso de processamento e armazenamento disponível nas 8 máquinas, apresenta aceleração de aproximadamente 1,32x em relação ao cenário *m:8_dn:8_wn:0* que apresentou o maior tempo médio. Por conter apenas *data nodes*, este cenário teve seu desempenho influenciado pela execução sequencial dos operadores da consulta. Os outros dois cenários *Baseline* (*m:8_dn:2_wn:6* e *m:8_dn:4_wn:4*) apresentam resultados semelhantes pois ambos contêm *worker nodes*, ou seja, realizam *pipeline* no processamento dos operadores da consulta, e também sofrem influência do tráfego de rede pelo motivo de cada nó estar alocado em uma máquina.

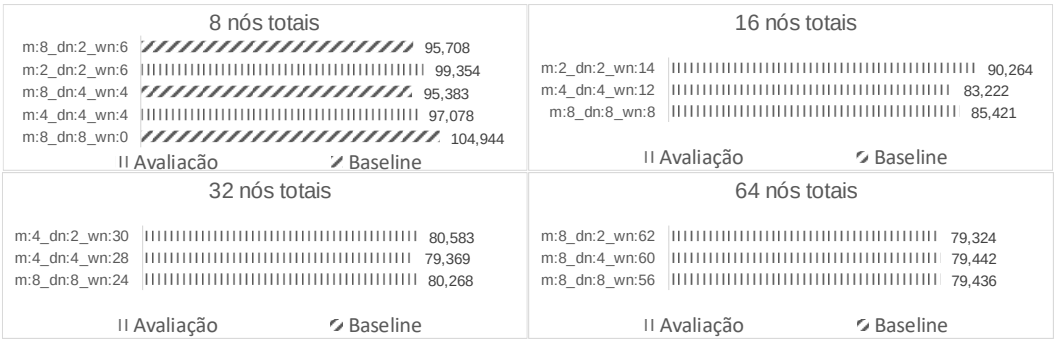


Figura 7. Tempo médio da consulta C1 em cenários com até 8 máquinas agrupados pela quantidade total de nós

A Figura 7 apresenta os resultados para a consulta C1, utilizando cenários com até 8 máquinas agrupados pela quantidade total de nós (8, 16, 32 e 64). Cenários com 8 nós totais apresentaram o maior tempo médio para execução da consulta C1. O cenário que não utiliza *worker nodes* apresentou o maior tempo médio dentre estes cenários, justificado pela execução sequencial dos operadores da consulta. Os demais cenários com quantidade totais de 16, 32 e 64 nós apresentam melhora gradativa no tempo médio do processamento da consulta C1, respectivamente, e apresentam resultados semelhantes dentro destas quantidades de nós totais. Isto porque estes cenários utilizam de maior quantidade de *worker nodes* que passam a processar pequenas quantidades de dados e aumentam o tráfego de rede. Desta forma, a Figura 7 evidencia que, para a consulta C1, é possível atingir um determinado desempenho com menor quantidade de máquinas dimensionando *workers nodes* e *data nodes* e explorando recursos de núcleos de processamento e discos de dados de uma mesma máquina. Por exemplo, o cenário *m:8_dn:4_wn:4* utiliza 8 máquinas e apresentou desempenho inferior aos cenários *m:2_dn:2_wn:14* e *m:4_dn:4_wn:28* que utilizam 2 e 4 máquinas, respectivamente.

Consulta C2 (triângulos). A Figura 8 apresenta o tempo médio de 5 rodadas da consulta C2. Em cenários com 64 nós totais, foi realizada apenas uma rodada da consulta, pois o MyriaX apresentou instabilidade no funcionamento.

Os cenários da Figura 8 com até dois *data nodes* mostram que cenários *Avaliação* apresentaram menor tempo médio em relação ao cenário *Baseline*. O cenário *Baseline m:2_dn:2_wn:0* apresentou desaceleração de aproximadamente 1,88x em relação ao cenário *Avaliação* de menor tempo médio (*m:8_dn:2_wn:62)*. Tal disparidade de tempos é influenciada pela execução sequencial dos operadores da consulta C2 realizada pelo cenário *Baseline*. Por outro lado, os cenários que utilizam *worker nodes* apresentam menor tempo médio. Cabe notar que o aumento de *worker nodes* em cenários *Avaliação* com 2 máquinas apresentou aceleração de até 1,7x em relação ao cenário *m:2_dn:2_wn:6*, 1,5x em relação ao cenário *m:2_dn:2_wn:14*. Isto ocorre porque, apesar de maior quantidade de *worker nodes* disponíveis para o processamento paralelo da consulta C2, há um maior tráfego de rede durante o fluxo dados para os *worker nodes* entre 2 máquinas. Este aumento no tráfego acontece pois a primeira auto-junção gera grande quantidade de possibilidades com dois vértices que são mantidas em memória enquanto a terceira operação de leitura de dados e a segunda junção são realizadas. Os cenários *Avaliação m:4_dn:2_wn:30* e *m:8_dn:2_wn:62*, por outro lado, apresentam aceleração de 1,77x e 1,88x, apesar de utilizarem mais *worker nodes*. Tal redução é justificada pelo aumento do paralelismo de I/O com a adição de máquinas no processamento da consulta.

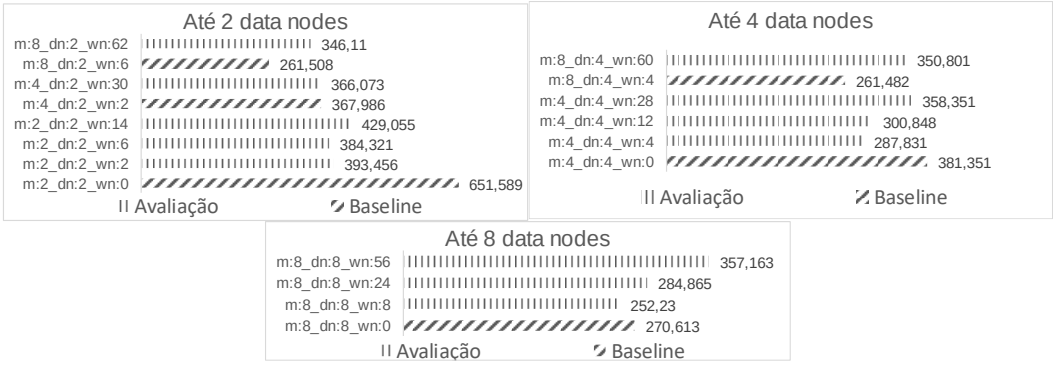


Figura 8. Tempo médio da consulta C2

Nos cenários com até 4 *data nodes* da Figura 8, os cenários *Baseline* apresentam diferentes tempos médios devido à existência ou não de *worker nodes*. O cenário *m:4_dn:4_wn:0*, apesar de utilizar mais máquinas, apresenta maior tempo médio pela execução sequencial dos operadores, justificada pela ausência de *worker nodes*. O cenário *m:4_dn:2_wn:2*, por sua vez, tem melhor desempenho por ter *pipeline* no processamento da consulta com 2 *worker nodes*. O cenário *Avaliação* com melhor tempo médio foi o *m:4_dn:4_wn:4*, com aceleração de 1,32x em relação ao melhor cenário *Baseline* (*m:4_dn:4_wn:0*). Da mesma forma que na avaliação anterior, o aumento de *worker nodes* em uma dada quantidade de máquinas causa desaceleração no processamento da consulta C2. Logo, os cenário *m:4_dn:4_wn:28* e *m:8_dn:4_wn:60*, que exploram todo o recurso disponível de 4 e 8 máquinas, apresentaram desaceleração de aproximadamente 0,8x em relação ao cenário *m:4_dn:4_wn:4*.

Nos cenários com até 8 *data nodes* da Figura 8, os cenários *Baseline* com uso de *worker nodes* também apresentam melhor desempenho em relação a cenários que contêm apenas *data nodes*. O cenário *Avaliação m:8_dn:8_wn:8* apresentou aceleração de até 1,07x e, portanto, o melhor desempenho para as 8 máquinas. É importante notar que o cenário *Avaliação m:8_dn:8_wn:56* apresentou maior tempo médio (desaceleração de 0,76x) no processamento da consulta C2 em relação ao cenário *Baseline* (*m:8_dn:8_wn:0*) para esta quantidade de máquinas. Isto porque, como dito anteriormente, sofre maior influência do tráfego de rede por explorar todos os núcleos de processamento com *worker nodes* em 8 máquinas. Tal desaceleração causada por gargalos de interferência e comunicação já era esperada, como discutido por StoneBraker [1986].

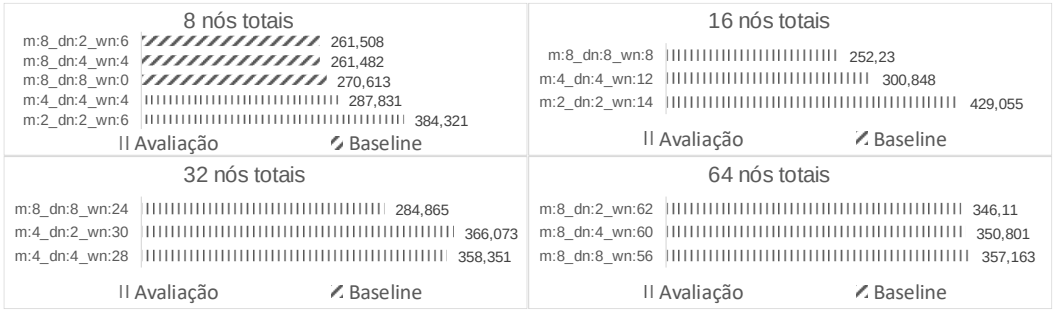


Figura 9. Tempo médio da consulta C2 referente aos cenários com até 8 máquinas agrupados pela quantidade total de nós

A Figura 9 apresenta os resultados para a consulta C2, utilizando cenários com até 8 máquinas agrupados pela quantidade total de nós (8, 16 e 32 e 64). Note que cenários

com 64 nós totais apresentam resultados de apenas uma rodada de consulta. O cenário *Avaliação m:8_dn:8_wn:8* apresentou ganho de aproximadamente 9s de tempo de processamento em relação ao cenário *m:8_dn:2_wn:6*, que apresentou menor tempo médio entre os *Baseline*. Cabe notar que para cenários de 8, 16 e 32 nós, que permitem variar quantidade de máquinas, o tempo médio de processamento da consulta C2 diminui com o aumento da quantidade de máquinas, ou seja, do paralelismo de I/O. Explorar totalmente o recurso de núcleos de processamento disponíveis para dada quantidade de máquinas causa um aumento no tempo de processamento devido ao gargalo formado pelo tráfego de rede [Stonebraker, 1986]. Neste sentido, o cenário *Baseline m:8_dn:2_wn:6* apresentou aceleração 1,45x sobre o cenário *Avaliação m:2_dn:2_wn:6*, sendo que ambos usam a mesma quantidade de *worker nodes* e *data nodes*. Isto porque o primeiro usa 8 máquinas (menor gargalo de rede), enquanto o segundo conta com 2 máquinas. O mesmo acontece para cenários com 16 e 32 nós totais. Cenários com 64 nós totais apresentam resultados semelhantes, pois usam a totalidade dos recursos disponíveis em 8 máquinas, variando apenas a quantidade de *data nodes*. A Figura 9 mostra que, para a consulta C2, o uso de todo o recurso disponível em dada quantidade máquinas piora o desempenho de processamento. Por outro lado, cenários *Avaliação* com quantidades iguais de *data nodes* e *worker nodes*, *m:8_dn:8_wn:8* e *m:4_dn:4_wn:4*, apresentaram os melhores desempenhos para a consulta C2 com aceleração de até 1,07x.

6. Conclusão

Sistemas de Gerência de Big Data têm sido usados para processamento massivo paralelo de consulta analíticas. Tais sistemas, em sua maioria, gerenciam cada máquina como um único tipo de nó dentro do *pipeline* de processamento paralelo da consulta. Com o advento da tecnologia *multi-core*, cada máquina passou a poder realizar inúmeras tarefas em paralelo. Neste contexto, este artigo avalia o uso de *worker nodes*, que participam do *pipeline* de processamento da consulta e não armazenam dados, e *data nodes*, que armazenam dados e também atuam como *worker nodes*, alocados em uma mesma máquina. O mecanismo de execução de consulta não-compartilhado MyriaX foi utilizado como plataforma base dos experimentos por permitir o uso desta abordagem. Visando confirmar a hipótese de que *é possível diminuir o tempo de processamento de consultas através da adição de worker nodes em núcleos ociosos de processadores,* experimentos foram realizados com dois tipos de consultas, uma de auto-junção (C1) e outra para identificação de triângulos (C2), utilizando uma base de dados do Twitter, sob diversas dimensões de *worker nodes* e *data nodes* em um *cluster*. Os resultados mostram que, para ambas consultas C1 e C2, sempre há pelo menos um cenário *Avaliação* (com adição de *worker nodes*) com melhor desempenho que o cenário *Baseline*. Assim, conseguimos neste trabalho confirmar a hipótese de pesquisa. No melhor caso, ao processar a consulta C1, explorando todos os recursos de núcleos de processamento por meio de *worker nodes*, conseguimos aceleração de até 2,92x. Entretanto, nem sempre a adição desenfreada de *worker nodes* apresentou melhores resultados; em quase todos os casos há uma deterioração a partir de um determinado ponto, de forma bem diferente para as consultas C1 e C2, devido às suas diferentes características. Isto mostra que a característica da consulta precisa ser levada em conta na escolha do melhor cenário de execução, e abre oportunidades para elaboração de heurísticas que possam ser usadas na geração do plano da consulta e adição de *worker nodes*. Além disso, os resultados mostram que a memória *cache* teve pouca influência no tempo de processamento das consultas. Em trabalhos futuros, pretendemos explorar esta questão, avaliando o comportamento de diferentes

cenários de *data nodes* e *worker nodes* com a aplicação de diversas consultas diferentes agrupadas em cargas de trabalho (*workloads*), o que nos permitirá também avaliar a influência de replicação de dados nos resultados obtidos.

Agradecimentos. Os autores agradecem ao CNPq e a UNEMAT pelo financiamento parcial desse trabalho.

Referências

Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Silberschatz, A. and Rasin, A. (2009). HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. *Proceedings of the VLDB Endowment (PVLDB)*, v. 2, n. 1, p. 922–933.

Alsubaiee, S., Behm, A., Grover, R., et al. (2012). ASTERIX: Scalable Warehouse-Style Web Data Integration. *International Workshop on Information Integration on the Web*, p. 1–4.

Bittorf, M., Bobrovytsky, T., Erickson, C. C. A. C. J., et al. (2015). Impala: A Modern, Open-Source SQL Engine for Hadoop. In *Conference on Innovative Data Systems Research (CIDR)*.

Dageville, B., Cruanes, T., Zukowski, M., et al. (2016). The Snowflake Elastic Data Warehouse. *International Conference on Management of Data (SIGMOD)*, p. 215–226.

Das, S., Agrawal, D. and El Abbadi, A. (2013). ElasTraS: An Elastic, Scalable, and Self-Managing Transactional Database for the Cloud. *Transactions on Database Systems (TODS)*, v. 38, n. 1, p. 5.

DeWitt, D. and Gray, J. (1992). Parallel Database Systems: The Future of High Performance Database Systems. *Communications of the ACM*, v. 35, n. 6, p. 85–98.

Gupta, A., Agarwal, D., Tan, D., et al. (2015). Amazon Redshift and the Case for Simpler Data Warehouses. In *International Conference on Management of Data (SIGMOD)*.

Halperin, D., Teixeira de Almeida, V., Choo, L. L., et al. (2014). Demonstration of the Myria Big Data Management Service. *International Conference on Management of Data (SIGMOD)*, p. 881–884.

Hu, X., Tao, Y. and Chung, C.-W. (2013). Massive Graph Triangulation. In *SIGMOD*.

Isard, M., Budiu, M., Yu, Y., Birrell, A. and Fetterly, D. (2007). Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. In *European Conference on Computer Systems (EuroSys)*.

Kim, C., Kaldewey, T., Lee, V. W., et al. (2009). Sort vs. Hash Revisited: Fast Join Implementation on Modern Multi-core CPUs. *Proceedings of the VLDB Endowment (PVLDB)*, v. 2, n. 2, p. 1378–1389.

Malewicz, G., Austern, M. H., Bik, A. J., et al. (2010). Pregel: A System for Large-scale Graph Processing. In *International Conference on Management of Data (SIGMOD)*.

Mehta, M. and DeWitt, D. J. (1997). Data Placement in Shared-nothing Parallel Database Systems. *The VLDB Journal*, v. 6, n. 1, p. 53–72.

Mishra, P. and Eich, M. H. (1992). Join Processing in Relational Databases. *ACM Computing Surveys (CSUR)*, v. 24, n. 1, p. 63–113.

Schneider, D. A. and DeWitt, D. J. (1989). A performance Evaluation of Four Parallel Join Algorithms in a Shared-nothing Multiprocessor Environment. *International Conference on Management of Data (SIGMOD)*, v. 18, p. 110–121.

Stonebraker, M. (1986). The Case for Shared Nothing. *IEEE Database Engineering*, v. 9, n. 1, p. 4–9.

Wang, J., Baker, T., Balazinska, M., et al. (2017). The Myria Big Data Management and Analytics System and Cloud Service. In *Conference on Innovative Data Systems Research (CIDR)*.

Warneke, D. and Kao, O. (2009). Nephele: Efficient Parallel Data Processing in the Cloud. In *Many-Task Computing on Grids and Supercomputers (MTAGS)*.

paper:171356

Extração de dados de conferências a partir da Web

Cássio Alan Garcia, Viviane P. Moreira

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

Abstract. *Choosing the most suitable conference to submit a paper is a task that depends on a number of factors including: (i) the topic of the paper needs to be among the topics of interest of the conference; (ii) submission deadlines need to be compatible with the necessary time for paper writing; and (iii) the quality or impact of the conference. These factors allied to the existence of thousands of conferences, make the search of the right event very time consuming, especially when researching in a new area. Intending to help researchers finding conferences, this paper presents a method developed to retrieve and extract data from conferences web sites. Our method combines the identification of conference URL and deadline extraction. The retrieved data is stored in a database to be searched with an online tool. The paper also reports on experiments that evaluate the quality of the extracted data, focusing on the deadlines.*

Resumo. *A escolha da conferência adequada para o envio de um artigo é uma tarefa que depende de vários fatores incluindo: (i) o tema do artigo deve estar entre os temas de interesse do evento; (ii) o prazo de submissão do evento deve ser compatível com tempo necessário para a escrita do artigo; e (iii) a qualidade da conferência. Esses fatores aliados à existência de milhares de conferências tornam a busca pelo evento adequado bastante demorada, em especial quando se está pesquisando em uma área nova. A fim de auxiliar os pesquisadores na busca de conferências, esse artigo apresenta um método desenvolvido para a coleta e extração de dados de sites de conferências. Este método combina a identificação de URLs de conferências da Tabela Qualis à identificação de deadlines. Os dados coletados populam uma base de dados que poderá ser consultada através de uma ferramenta online. O artigo também relata experimentos que avaliam a qualidade dos dados extraídos, enfatizando a extração dos deadlines.*

1. Introdução

O processo de escrita e submissão de artigos científicos é crucial na vida dos pesquisadores. A escolha do periódico ou conferência mais adequados para a divulgação da pesquisa realizada é uma tarefa bastante importante e que por vezes toma bastante tempo dos pesquisadores.

Existem milhares de conferências científicas que ocorrem anualmente. Quando se deseja submeter um artigo para uma conferência, vários aspectos precisam ser levados em consideração: (i) tema do trabalho deve estar entre os temas de interesse do evento para que ele possa ser considerado; (ii) é necessário saber se os prazos (*deadlines*) do evento são compatíveis com os do término da escrita do artigo (ou algum outro critério temporal como o prazo para a conclusão do curso, por exemplo); (iii) questões de valores

financeiros como o local de realização da conferência e o valor da taxa de inscrição que podem inviabilizar a participação dos autores; e (iv) a qualidade do evento também é importante para essa escolha – um trabalho em nível inicial pode ser enviado para um evento de menor impacto, enquanto que um trabalho de final de doutorado, por exemplo, pode ser enviado para um evento de maior qualificação.

No Brasil, a avaliação da produção intelectual de programas de pós-graduação é feita pela CAPES e baseia-se no sistema Qualis. O Qualis [Souza and Paula, 2002] é um instrumento avaliativo cujo resultado é uma tabela que atribui um grau a conferências e periódicos visando refletir sua qualidade. Os graus são chamados *estratos* e podem assumir os valores A1, A2, B1, B2, B3, B4, B5 e C, sendo A1 o mais elevado.

A tarefa de encontrar uma conferência que seja adequada ao tema, ao prazo e ao Qualis costuma consumir um tempo considerável dos pesquisadores, em especial quando se está começando em uma área de pesquisa nova da qual não se conhece os eventos. Existem alguns web sites que reúnem chamadas de submissão de artigos (*call for papers - CFPs*), tais como ConfSearch[1] e WikiCFP[2], contudo a maioria baseia-se na inclusão manual de informações e não contempla o cruzamento de dados com o Qualis. O objetivo deste artigo é preencher essas lacunas, permitindo que a comunidade possa consultar dados atualizados sobre CFPs.

Este artigo apresenta um método automático de coleta e extração de dados de conferências. Os dados extraídos são armazenados em um banco de dados de eventos que permite a busca de acordo com seu tópico, prazos de submissão e Qualis. O foco principal deste artigo é a etapa de extração das datas de interesse a partir do web site da conferência (todas as etapas são detalhadas na Seção 4).

Como cada conferência possui seu próprio website, com diferentes layouts, formatos de datas e rótulos para representar um dado *deadline*, o desafio está em conseguir contornar esta grande variação na forma de expor a informação a fim de extrair corretamente as datas de cada *deadline* do evento. Além de reconhecer estes diferentes padrões de datas e rótulos, é necessário vincular corretamente as datas encontradas a seus rótulos, de forma que a data do *deadline* de submissão de artigo não seja atribuída à *deadline* de submissão de resumo, por exemplo.

2. Trabalhos Relacionados

Nesta Seção abordaremos trabalhos relacionados à extração de datas de interesse de conferências.

2.1. Extração de datas de conferências

Em nossas pesquisas, verificamos a existência de trabalhos que centralizam informações referentes a conferências. Por exemplo, o WikiCFP[3] é um repositório que reúne milhares de CFPs para eventos nas áreas de ciência e tecnologia. O site relata receber cerca de 100 mil visitas mensais. Contudo, o site depende que os dados das CFPs sejam inseridos manualmente por um usuário (por exemplo, um dos organizadores do evento).

[1] `http://www.confsearch.org/confsearch/`
[2] `http://wikicfp.com/cfp/`
[3] `http://wikicfp.com/cfp/`

Por sua vez, o AllCall [Correia et al., 2010], mais automatizado, é um sistema que processa CFPs contidas em e-mails a fim de extrair tópicos e datas importantes. Para que o sistema funcione, o usuário precisa estar inscrito em listas de e-mails que recebam mensagens com CFPs.

Já o ConfSearch[4] baseia-se em dados do DBLP[5], um importante repositório de dados bibliográficos da Ciência da Computação. Dados de conferências que não estão disponíveis no DBLP, como os prazos de submissão, precisam ser cadastrados pelos usuários do site.

O presente trabalho difere do WikiCFP e do ConfSearch por fazer a coleta e a extração dos dados das conferências de forma automática. Em relação ao AllCall, nosso diferencial está em não depender do recebimento de emails com CFPs. Além disso, nenhum dos sistemas pesquisados leva em conta a classificação Qualis dos eventos – sendo esta classificação de grande importância no Brasil.

2.2. Extração de Informações com *Conditional Random Fields*

Conditional Random Fields (CRF) [Lafferty et al., 2001] é um modelo estocástico utilizado habitualmente para etiquetar e segmentar sequências de dados ou extrair informações de documentos textuais. Em Vieira et al. [2015] o CRF é utilizado para analisar comentários sobre de produtos e verificar marca e modelo a que se referem. É fornecida uma quantidade pequena de dados de treinamento e a partir destas chamadas "sementes" anotadas, o modelo CRF é criado, aplicado a sentenças não rotuladas, encontrando novos modelos de produtos, que são adicionados ao conjunto de sementes e re-gerado o modelo CRF, terminando quando não forem mais encontradas sementes.

Para fins de extração de conteúdo de uma página web, pode-se dar ênfase aos segmentos relevantes de uma página, distinguindo título, autor, conteúdo, dos comentários, anúncios, ou então a distinção entre imagem, descrição e preço de um produto, por exemplo. Para isso, o CRF foi aplicado como um *framework* de segmentação em Fu et al. [2010] e em Zhu et al. [2005]. Foram definidas *features* que determinam distâncias entre blocos, características do texto (quantidade de números no texto, pontuação...) e características de *layout* (tags HTML, como H1, H2, a, li...).

Em Gong and Liu [2012] o *framework* CRF é combinado com técnicas de *Tree Edit Distance* aplicadas sobre a estrutura DOM como uma alternativa para a realização da tarefa de segmentação de páginas e Extração de Informação. Uma vez criada a ferramenta de segmentação, são realizados experimentos para extração de blocos de notícias.

Ainda trabalhando com o conceito da informação estar disposta em duas dimensões, em Pinto et al. [2003], o *framework* CRF é aplicado à extração de informações a partir de tabelas em páginas Web. Com base nas *features* propostas, a tabela é detectada em meio à página em uma primeira etapa e, em seguida, são rotulados títulos, cabeçalhos, linhas contendo dados e suas colunas.

Uma das principais tarefas da Extração de Informação é a detecção de relações entre palavras para, a partir de dados não estruturados, extrair informações estruturadas.

[4] `http://www.confsearch.org/confsearch/`
[5] `http://dblp.uni-trier.de/`

A exemplo disso, o *TextRunner* [Etzioni et al., 2008] utiliza-se do CRF para aprender o padrão sintático das sentenças e então extrair tuplas do tipo Entidade-Relação-Entidade.

2.3. Extração de Informações com outras técnicas

Em se tratando de correlação de informações em páginas, o trabalho Nguyen et al. [2008] faz a extração de rótulos relacionados a campos de *webforms* no sistema chamado *LabelEx*. A informação contida nestes rótulos é importante para a recuperação de informação na Web oculta que requer o preenchimento de formulários recuperar dados de bancos de dados online. Para a extração dos rótulos, foram testados os classificadores *Naive Bayes*, *Decision Tree*, *Suport Vector Machines - SVM* e *Regressão Logística*. Inicialmente são gerados "mapeamentos candidatos" entre rótulos e campos de entrada próximos uns aos outros. O classificador *Naive Bayes* foi o que obteve melhores resultados sendo utilizado para uma tarefa de poda de algumas relações falsas. Em seguida, para a localização da melhor relação entre rótulo e campo de entrada, o algoritmo *Decision Tree* foi escolhido por obter os melhores resultados dentre os classificadores testados. Para fins de experimento, o *LabelEx* é aplicado sobre formulários de diferentes domínios, obtendo altos índices de *F-measure*.

Outra proposta para extração de dados independente de estrutura de sites é feita por Gogar et al. [2016] com a utilização de Redes Neurais Profundas para criação de um modelo combinando informações textuais e de *layout*. Nos experimentos, o modelo é aplicado à extração de informações de produtos no *e-commerce*, tais como nome do produto, imagem e preços, atingindo alta acurácia.

Quanto à extração de dados de *Calls-For-Papers* (*CFPs*) enviados por e-mail, Li et al. [2013] trabalha na extração de afiliações, ou seja, organizações às quais pesquisadores estão vinculados. O método proposto combina *Named Entity Recognition* (NER) e informações de *layout* para a realização da extração. O primeiro é utilizado para detectar nome do pesquisador, da organização e outros; informações de *layout* são utilizadas para a diferenciação de áreas com informações relevantes para a extração, das demais regiões. Este trabalho não faz extração das datas dos CFPs.

3. Definição do Problema e Visão Geral da Solução

O problema investigado neste artigo pode ser resumido como *"Dado um conjunto de conferências de interesse (representadas por seus nomes e siglas) encontre suas páginas web e respectivas datas de interesse"*. A Tabela Qualis (que contém a sigla, o nome e a classificação das conferências) torna-se então o ponto de partida para a resolução deste problema. A fim de atingir o objetivo proposto, definimos um processo composto pelas seguintes etapas apresentadas na Figura 1: (i) descoberta das URLs das conferências, (ii) download do conteúdo, (iii) extração das datas, e (iv) disponibilização da informação para consulta online.

A seguir, definiremos os termos utilizado no decorrer do artigo que são necessários ao entendimento da proposta:

- *datas de interesse*: são os valores das datas limites, ou *deadlines*, para entrega de resumo, do artigo, data de notificação de aceitação, envio da versão final e período da conferência;

- *rótulo*: trata-se do texto usualmente vinculado a uma data, indicando a qual das datas de interesse aquela data se refere. Um exemplo de rótulo referente ao *deadline* de submissão de resumo é *"abstract submission"*, ou *"abstracts deadline"*.

Para exemplificar, considere a Figura 2. A data de interesse "*January 17, 2017*" refere-se ao prazo de submissão do resumo, cujo rótulo é "*Abstracts for full research papers due*" enquanto que a data de submissão do artigo "*January 24, 2017*" está associada ao rótulo "*Full Research papers due*".

Na primeira etapa, a partir de uma lista de conferências de interesse (dada pela tabela Qualis), é necessário descobrir a URL do site da conferência. Isto é feito por meio de consultas a motores de busca na Web. As consultas são compostas pelo ano, sigla e o nome da conferência. Apenas as três primeiras URLs retornadas são mantidas na fase seguinte. Além disso, são excluídas URLs de sites como *facebook*, *wikepedia*, *twitter*, *linkedin*, *github*, etc, que têm baixa probabilidade de ser a página oficial de uma conferência.

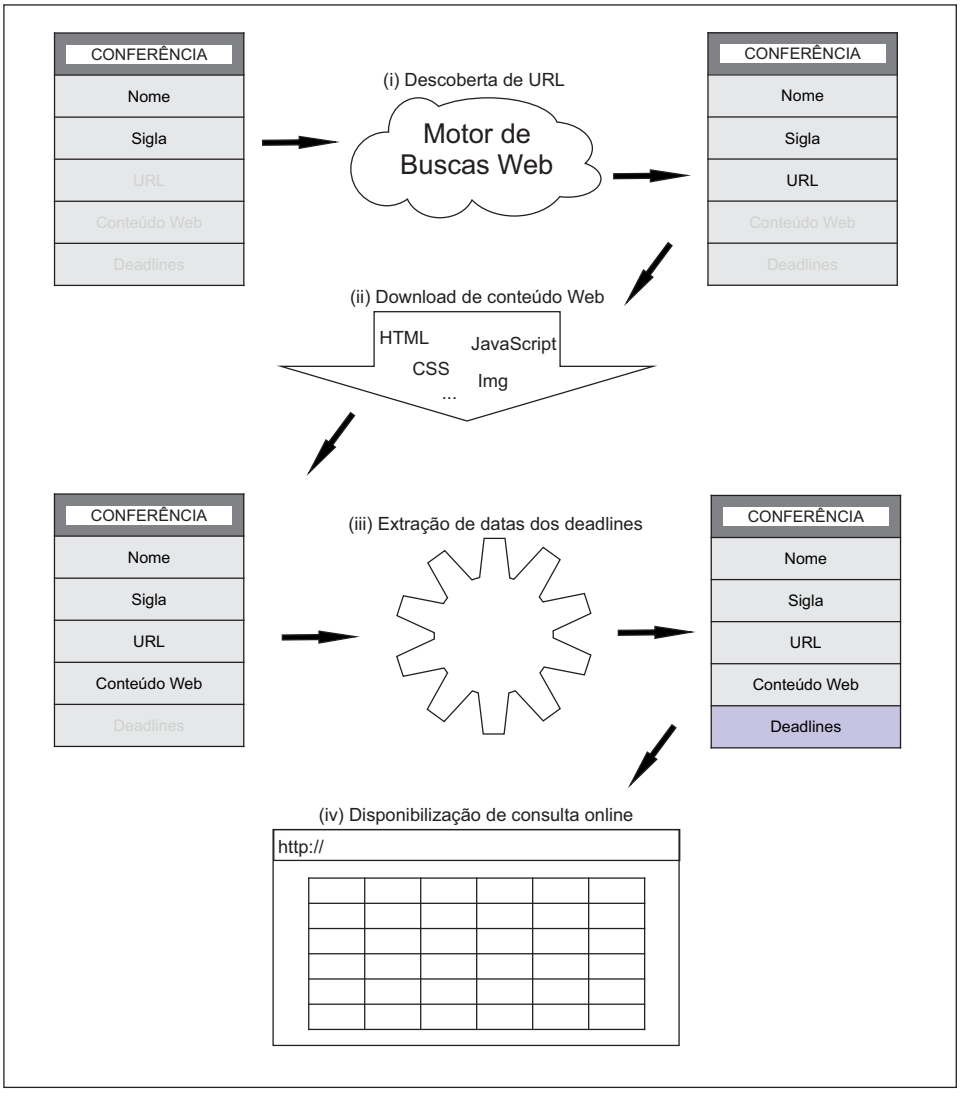


Figura 1. Etapas do processo de extração de datas



Figura 2. Exemplo de página de conferência com datas de interesse e rótulos. Extraída da página do EDBT 2017 http://edbticdt2017.unive.it/?important_dates

Na segunda etapa, é feito o *download* do conteúdo das URLs retornadas pela etapa anterior. São baixados recursivamente todos os documentos vinculados à URL informada, limitando-se a dois níveis. É baixado somente o conteúdo HTML, desta forma, arquivos com extensões mp3, mp4, pdf, ppt, pptx, doc, docx, etc. são ignorados.

Na terceira etapa, os arquivos são analisados, as datas são extraídas e armazenadas. Esta etapa é o foco principal deste trabalho e é detalhada na Seção 4.

A quarta etapa consiste na disponibilização de um ambiente web para a consulta das conferências e suas datas. Desta forma, as datas coletadas poderão ser facilmente e gratuitamente acessadas pela comunidade científica.

4. Extração de datas

Para a localização das datas em páginas HTML (Figura 3), utilizamos *Conditional Random Fields* (CRF) [Lafferty et al., 2001], um *framework* para geração de modelos probabilísticos para segmentação e rotulação de dados. A motivação para esta escolha foram os bons resultados dessa técnica aplicados a tarefas similares de Extração de Informações, conforme relatado na Seção 2.

Uma etapa de pré-processamento (i) é necessária (Figura 3), pois a ferramenta de CRF utilizada necessita que o arquivo de treinamento esteja em um formato específico: cada linha do arquivo é formada por N colunas. A primeira coluna representa o *token*, ou seja, o texto do HTML é extraído (removendo-se as *tags*) e cada linha deste arquivo de treinamento representa uma palavra do texto extraído; a N-ésima coluna de cada linha representa a Classe manualmente anotada; as colunas intermediárias representam as *features* implementadas e descritas a seguir.

Como entrada para o treinamento do CRF, foram anotadas as datas de interesse e os rótulos. Uma vez gerado o modelo, este é aplicado a todas as conferências. Este processamento gera um arquivo semelhante ao da Figura 4, porém com uma coluna a mais, contendo a predição do CRF. A partir das previsões, as datas de interesse são extraídas.

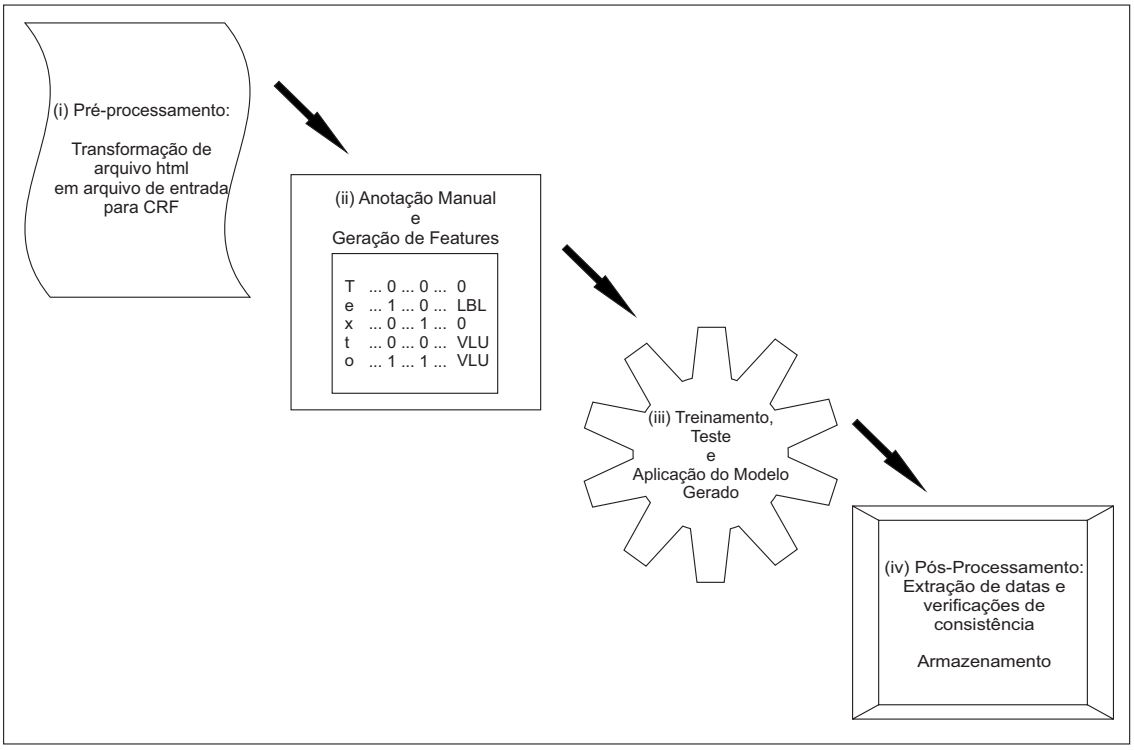


Figura 3. Extração de datas de interesse

No CONFTRACKER, o modelo CRF foi treinado para detectar dez possíveis classes de saída, referentes aos rótulos e valores das datas de interesse:

- deadline de resumo (`LBL_ABS` e `VLU_ABS`),
- deadline de artigo (`LBL_PPR` e `VLU_PPR`),
- notificação de aceitação (`LBL_ACC` e `VLU_ACC`),
- deadline para submissão da versão final (`LBL_CAM` e `VLU_CAM`) e
- período da conferência (`LBL_EVE` e `VLU_EVE`).

Previamente foram definidos *tokens* que podem referenciar os rótulos das datas de interesse (LBL_*). Por exemplo, os *tokens* $abstract$, $paper$, $submission$ são associados aos rótulos do deadline de submissão de resumo; *tokens* $camera$, $ready$, due, associados aos rótulos de confirmação de aceitação, etc.

Para a detecção das classes de rótulo e valores de datas de interesse, é necessário definir os atributos (ou *features*) a serem considerados pelo CRF. Foram criadas nove *features* para a detecção destas classes: seis delas para distinguir entre os rótulos das diferentes datas de interesse e as outras três para detectar datas. São elas:

- *token* avaliado pertence aos *tokens* previamente definidos para cada uma das cinco datas de interesse; *token* é uma sigla de evento (obtida pela tabela Qualis);
- *token* avaliado é potencialmente um dia, mês ou ano.

O *framework* CRF também necessita de um *template*, que permite a criação de uma janela de contexto, ou seja, ao classificar um *token* (ou aprender a fazê-lo) considerar também os x *tokens/features* anteriores e próximos. No arquivo de *templates*, exemplificado na Figura 5, cada linha representa um *template* (linhas iniciando com # são comentários) no formato prefixo, id e regra: "$U_i : \%x[row, col]$", onde *U* é o prefixo;

i é o id; *%x* é fixo; *row* é a linha, que pode ser definida com números negativos (indicando linhas prévias ao *token* sendo classificado), número zero (própria linha) e números positivos (próximas linhas); *col* é a coluna, indicando a *feature* a ser analisada.

A partir do arquivo de saída do CRF é executado o pós-processamento, que verifica as classes atribuídas pelo algoritmo, extraindo as datas e fazendo uma verificação de consistência. As datas de interesse de uma conferência obedecem a uma ordem *deadline de submissão de resumo* $<$ *deadline de submissão de artigo* $<$ *data de notificação de aceitação* $<$ *deadline de submissão de versão final* $<$ *período da conferência*. Logo, as datas extraídas precisam obedecer a esta regra.

```
...
dates 0 0 0 0 1 0 0 0 0 0
january 0 0 0 0 0 0 1 0 0 VLU_ABS
14 0 0 0 0 0 1 0 1 0 VLU_ABS
2016 0 0 0 0 0 0 0 0 1 0 VLU_ABS
abstracts 1 0 0 0 0 0 0 0 0 0 LBL_ABS
for 0 0 0 0 0 0 0 0 0 LBL_ABS
full 0 1 0 0 0 0 0 0 0 LBL_ABS
research 0 1 0 0 0 0 0 0 0 LBL_ABS
papers 0 1 0 0 0 0 0 0 0 LBL_ABS
due 1 0 0 1 0 0 0 0 0 LBL_ABS
january 0 0 0 0 0 0 1 0 0 VLU_PPR
21 0 0 0 0 0 1 0 1 0 VLU_PPR
2016 0 0 0 0 0 0 0 0 1 0 VLU_PPR
full 0 1 0 0 0 0 0 0 0 LBL_PPR
research 0 1 0 0 0 0 0 0 0 LBL_PPR
papers 0 1 0 0 0 0 0 0 0 LBL_PPR
...
```

Figura 4. Arquivo de treinamento

```
...
# token
U49:%x[0,0]
# abstract
U50:%x[0,1]
# papper
U51:%x[0,2]
# acceptance
U52:%x[0,3]
# camera ready
U53:%x[0,4]
# event period
U54:%x[0,5]
# day
U85:%x[0,6]
# month
U86:%x[0,7]
# year
U87:%x[0,8]
# conference initials
U88:%x[0,9]
...
```

Figura 5. Arquivo de *template*

5. Avaliação Experimental

Para a avaliação da qualidade da extração, foram realizados experimentos comparando as datas extraídas pelo CRF às datas esperadas. Além disso, os resultados do CONFTRACKER foram comparados aos de um *baseline*. Embora o método proposto possa ser aplicado a conferências de diferentes àreas de interesse, nossos experimentos foram realizados sobre conferências da Ciência da Computação.

5.1. Materiais e Métodos

Gold Standard. A partir das conferências da Tabela Qualis[6], foram escolhidas aleatoriamente oitenta, as quais foram utilizadas para a avaliação da extração. Sobre as conferências sorteadas, cada uma das datas disponíveis foi pesquisada manualmente para a criação do *gold standard* que contém as datas esperadas. Como o objetivo do experimento é avaliar apenas a qualidade da extração de datas, as URLs das conferências foram selecionadas manualmente.

Ferramentas. O sistema foi desenvolvido na linguagem C# da plataforma .Net, sobre o sistema operacional Windows 10, fazendo uso de algumas ferramentas para tarefas específicas, tais como:

- Wget[7]: para download do conteúdo Web;
- CRFSharp[8]: utilizada para aplicação do *framework* CRF;

Métricas de avaliação. As métricas utilizadas para a avaliação da qualidade da extração são Precisão (P), Revocação (R) e F-mesure (F). Definidas por:

$$P = \frac{|R_q|}{|S_q|}, \quad R = \frac{|R_q|}{|R|}, \quad F = \frac{2 \times P \times R}{P + R}$$

onde R_q é o número de datas corretas extraídas, $|S_q|$ é o número de datas extraídas pelo método sendo avaliado e $|R|$ é o número de datas do *gold standard*.

Baseline. Estes resultados são comparados com os de um *baseline* implementado por nós baseando-se em Mattes [2011] que propôs uma forma de automatizar o processo de extração de datas do ConfSerach explorando a posição dos rótulos e dos valores. A páginas são renderizadas em memória, juntamente com CSS e Javascript, para que as posições [x, y] de rótulos e valores de datas sejam descobertos. Esta técnica, além de avaliar o conteúdo na forma como ele estaria sendo exibido, tenta emular a percepção humana de uma página exibida. São criados pares de "nodos rótulo" e "nodos data", relacionando todos esses nodos encontrados na página e definidos valores de importância para a posição das datas em relação aos rótulos, conforme ilustrado na Figura 6. As datas recebem um *score* em função da sua posição em relação ao rótulo selecionado.

Procedimento. Uma vez extraídas as datas, é executado o módulo de avaliação que faz os cálculos de precisão, revocação e *f-measure*, medindo, assim, a qualidade da extração dos métodos CONFTRACKER e *baseline*. Além disso, avaliamos os resultados da combinação do CONFTRACKER com o *baseline*. Esta combinação foi obtida a partir dos resultados

[6] https://www.capes.gov.br/images/stories/download/avaliacao/Comunicado_004_2012_Ciencia_da_Computacao.pdf

[7] https://www.gnu.org/software/wget/

[8] https://github.com/zhongkaifu/CRFSharp

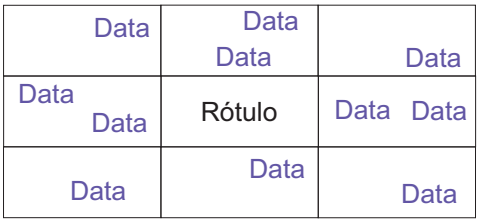


Figura 6. Posições de datas em relação a um rótulo

do CONFTRACKER, preenchendo-se as lacunas com as datas encontradas pelo *baseline*. Esta estratégia foi escolhida, pois conforme resultados apresentados e discutidos na Seção 5.2, o CONFTRACKER foi bastante superior ao *baseline*, tendo-se assim a extração do CONFTRACKER como resultado principal, sendo complementado pelo *baseline*.

5.2. Resultados e Discussão

A Tabela 1 mostra os resultados das execuções do *baseline*, do CONFTRACKER e da combinação entre eles, aplicadas sobre as oitenta conferências utilizadas. Considerando-se a média de todas as datas de interesse, O CONFTRACKER obteve a melhor precisão (0,804) e a melhor F-measure (0,703). A precisão atingida pelo CONFTRACKER foi 54,8% maior que a do *baseline*, tendo atingido 0,80 contrastando com o índice de 0,52 do *baseline*. A revocação resultante do CONFTRACKER teve um percentual de superioridade próximo ao da Precisão: 56,2%, tendo o CONFTRACKER atingido 0,63 enquanto o *baseline* obteve 0,52. Comparando-se os índices de *F-measure* da extração executada pelo *baseline* aos da extração do CONFTRACKER, nota-se uma superioridade significativa do CONFTRACKER. Isso é comprovado pelo Teste-T que resultou p-valor de $3{,}94 \times 10^{-8}$.

Comparando o CONFTRACKER com a combinação CONFTRACKER + *baseline*, observa-se uma F-measure bastante próxima. Conforme o esperado, a melhor revocação foi obtida pela combinação de métodos CONFTRACKER e *baseline* (0,694), pois desta forma mais datas de interesse foram localizadas. Contudo, algumas datas localizadas pelo *baseline* não estavam corretas, o que teve um impacto negativo sobre a precisão e, por consequência, sobre a F-measure. Assumindo que a precisão seja um índice mais importante que a revocação na resolução do problema definido neste trabalho, pois vem a ser preferível que uma data não tenha sido preenchida do que ter sido extraída erroneamente, a melhor alternativa das três abordagens aqui expostas é a utilização do método CONFTRACKER isoladamente.

Avaliando a qualidade da extração por tipo de data de interesse, percebe-se que as datas de início e fim do evento foram extraídas com maior taxa de acerto pelo CONFTRACKER. Acreditamos que isso ocorreu o por se tratar de um intervalo de datas, distinguindo das demais *deadlines*, que são datas simples. Esta distinção não impactou os resultados do *baseline*.

A *deadline* de submissão do artigo foi a que teve o pior resultado na extração do CONFTRACKER. Percebeu-se que isso ocorre devido aos termos dos rótulos relacionados a esta data de interesse serem muito genéricos e estarem presentes em muitas outras partes da página, como em títulos e em meio ao texto em geral. Isso dificultou com que o CRF aprendesse a extrair esta data específica. A mesma tendência se observa com a extração do *baseline*.

Tabela 1. Resultados dos experimentos

	Abstract	Paper	Notification	Camera Ready	Conf Start	Conf End	Médias
Posicional							
Precisão	0,296	0,351	0,440	0,448	0,775	0,804	0,519
Revocação	0,242	0,260	0,376	0,400	0,584	0,569	0,405
F-measure	0,266	0,299	0,406	0,422	0,666	0,666	0,454
CRF							
Precisão	**0,823**	**0,620**	**0,800**	**0,784**	**0,898**	**0,898**	**0,804**
Revocação	0,424	0,493	0,637	0,615	0,815	0,815	0,633
F-measure	0,560	**0,549**	**0,709**	**0,689**	0,854	0,854	**0,703**
CRF + Posicional							
Precisão	0,620	0,567	0,712	0,651	0,863	0,863	0,713
Revocação	**0,545**	**0,520**	**0,681**	**0,661**	**0,876**	**0,876**	**0,693**
F-measure	**0,580**	0,542	0,696	0,656	**0,870**	**0,870**	0,702

Limitações. Uma situação que prejudica os resultados aqui apresentados é a exibição das datas como imagens. Nestes casos não está sendo possível extrair as informações necessárias, pois no momento não fazemos uso de técnicas de *Optical Character Recognition* (OCR).

Apesar de serem previstos centenas de padrões de datas, ainda há algumas que não são contempladas, como casos em que consta o dia da semana em meio à data. Por exemplo, *04 (tue), october 2016*. Caso este padrão também fosse contemplado, a expressão regular poderia se tornar muito genérica, tendo maiores chances de aceitar padrões que não correspondessem a datas válidas. O mesmo vem a acontecer com a detecção de rótulos de interesse. Como as informações em páginas Web são expressas em linguagem natural, há ainda formas de expor o mesmo significado em rótulos diferentes, ainda não previstos nos padrões definidos no sistema.

Existe também a questão das datas serem atualizadas nos sites das conferências. Desta forma, faz-se necessário definir uma recorrência periódica para que as etapas (ii) e (iii) da Figura 1 sejam re-executadas. Uma estratégia seria executá-las a cada quinze dias para todas as conferências e, ao se aproximar dos *deadlines* de *abstract submission* e *full paper submission*, executar o processo diariamente. Assim, quando o usuário final consulta o ambiente web, a informação estará atualizada na base de dados.

6. Conclusão

Este trabalho propôs um método para a extração automática de dados de conferências. Nosso método, chamado de CONFTRACKER, emprega um processo em quatro etapas. Neste artigo, a ênfase foi na etapa de extração de datas que faz uso do *fremework* CRF.

Para realização de experimentos e avaliações, algumas conferências foram selecionadas randomicamente a partir da Tabela Qualis, sobre as quais foi gerado o *gold standard*. Para fins de comparação, implementamos um *baseline* baseado na distância entre os rótulos e as datas. Avaliamos também a combinação da técnica proposta com o *baseline*. Os resultados mostraram que a técnica proposta é significativamente melhor do que o *baseline*.

Como exposto, ambas as técnicas de extração, utilizadas para relacionar rótulos e datas dependem imensamente da capacidade de localizar as informações em linguagem

natural e não estruturada (os rótulos) utilizada nas páginas Web. Trabalhos futuros podem ser realizados no sentido de aperfeiçoar a geração das *features* utilizadas pelo CRF. Por exemplo, as que definem se um *token* está relacionado a datas de interesse. Hoje os possíveis rótulos são salvos em uma base de dados e ao executar a função geradora da *feature*, verifica-se a existência do *token* em questão na base de dados. Seria possível trabalhar a aplicação de aprendizagem de máquina para encontrar os *tokens* relacionados a cada data de interesse para que o valor gerado pela *feature* seja mais confiável.

Agradecimentos: Este trabalho foi parcialmente financiado pelo CNPq.

Referências

Fábio L Correia, Rui FS Amaro, Luís Sarmento, and Rosaldo JF Rossetti. Allcall: An automated call for paper information extractor. In *Information Systems and Technologies (CISTI), 2010 5th Iberian Conference on*, pages 1–4, 2010.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

Lei Fu, Yingju Xia, Yao Meng, and Hao Yu. Conditional random fields model for web content extraction. In *Computing in the Global Information Technology (ICCGI)*, pages 30–34, 2010.

Tomas Gogar, Ondrej Hubacek, and Jan Sedivy. *Deep Neural Networks for Web Page Information Extraction*, pages 154–163. 2016.

Yunfei Gong and Qiang Liu. Automatic web page segmentation and information extraction using conditional random fields. In *Computer Supported Cooperative Work in Design (CSCWD)*, pages 334–340, 2012.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the international conference on machine learning, ICML*, volume 1, pages 282–289, 2001.

Xinyu Li, Roya Rastan, John Shepherd, and Hye Young Paik. Automatic affiliation extraction from calls-for-papers. In *Proceedings of the Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 97–102, 2013. ISBN 978-1-4503-2411-3.

Jochen Mattes. Automated meta-data extraction for confsearch. Technical report, 2011.

Hoa Nguyen, Thanh Nguyen, and Juliana Freire. Learning to extract form labels. *Proceedings of the VLDB Endowment*, 1(1):684–694, 2008.

David Pinto, Andrew McCallum, Xing Wei, and W Bruce Croft. Table extraction using conditional random fields. In *Proceedings of the annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 235–242, 2003.

Elaine Pereira de Souza and Maria Carlota de Souza Paula. Qualis: a base de qualificação dos periódicos científicos utilizada na avaliação capes. *InfoCAPES Boletim Informativo*, 10(2), 2002.

Henry S Vieira, Altigran S da Silva, Marco Cristo, and Edleno S de Moura. A self-training crf method for recognizing product model mentions in web forums. In *European Conference on Information Retrieval*, pages 257–264, 2015.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2d conditional random fields for web information extraction. In *Proceedings of the International Conference on Machine Learning*, pages 1044–1051, 2005.

paper:171363

Identificação do grafo de genealogia acadêmica de pesquisadores: Uma abordagem baseada na Plataforma Lattes

Rafael J. P. Damaceno[1], **Luciano Rossi**[1], **Jesús P. Mena-Chalco**[1]

[1]Universidade Federal do ABC (UFABC)
Santo André - SP - Brasil

{rafael.damaceno,luciano.rossi,jesus.mena}@ufabc.edu.br

Abstract. *Different academic-scientific knowledge areas have made efforts to create databases related to researchers as well as the advisor-advisee relationships. However, most of these databases present problems such as redundancy, absence, and imprecision of information. In this paper, we present an algorithm for the automatic identification of academic genealogical graph. The contribution of our work relies on the precision of the hierarchical structure of academic advisor-advisee relationships resulting from the algorithm (graph), which facilitates genealogical analysis. As a case study, we prospected more than 272 thousand Ph.D. researchers registered in the Lattes Platform. Novel characteristics on the Brazilian academic genealogy are discussed in detail.*

Resumo. *Diferentes áreas do conhecimento acadêmico-científico têm realizado esforços para a criação de bases de dados de pesquisadores e seus relacionamentos de orientação. No entanto, grande parte destas bases apresentam problemas como redundância, ausência e imprecisão de informações. Neste artigo, apresentamos um algoritmo para a identificação automática de registros acadêmicos, considerando estes problemas. A contribuição deste trabalho recai na precisão da estrutura hierárquica de orientação acadêmica resultante do algoritmo (grafo), que facilita análises genealógicas. Como estudo de caso, prospectamos mais de 272 mil doutores registrados na Plataforma Lattes e apresentamos características inéditas sobre a genealogia acadêmica brasileira.*

1. Introdução

O acesso e a estrutura de dados que estão disponíveis nos mais diversos contextos nem sempre são suficientes para a extração de conhecimento. Assim, muitas vezes informações de interesse não podem ser obtidas a partir de consultas simples que se limitam à estrutura estabelecida, sendo necessário reorganizar os dados previamente para facilitar a recuperação de informação [Korfhage 1997].

Este trabalho está relacionado com métodos computacionais para criação de grafos que permitam o estudo de parentesco acadêmico denominado Genealogia Acadêmica (GA). A GA é uma ferramenta importante no estudo da propagação do conhecimento acadêmico-científico por meio dos relacionamentos de orientação [Sugimoto 2014]. Bancos de dados com informações sobre genealogia, especificamente aqueles dedicados a academia, comumente apresentam uma estrutura bem definida. Dessa forma, é possível obter informações (atributos) diretamente relacionadas aos acadêmicos (registros).

Diferentes áreas do conhecimento acadêmico-científico têm realizado esforços na estruturação de bancos de dados com informações de parentesco acadêmico. Os matemáticos possuem um banco de dados com informações genealógicas desta comunidade acadêmica que é disponibilizado pelo *Mathematics Genealogy Project – MGP*[1]. Outras áreas do conhecimento, como os neurocientistas e os cientistas da computação, dentre outros, possuem seus registros genealógicos mantidos pela plataforma *The Academic FamilyTree*[2]. Ambos os exemplos, apesar da indiscutível relevância, não fornecem a possibilidade de consulta e análise do grafo de genealogia de forma ampla.

No contexto nacional, a Plataforma Lattes apresenta-se como o maior e mais relevante repositório curricular de acadêmicos relacionados com a Ciência Brasileira [Lane 2010]. Para todos os acadêmicos registrados na Plataforma Lattes são encontradas informações correspondentes à produção científica, técnica e artística, sendo que a informação sobre orientação/supervisão está disponível para acadêmicos com atuação na formação de alunos. Como apontado por [Mena-Chalco and Cesar Junior 2013], essa informação registrada não mantém um padrão e, em muitos casos, é incompleta e sujeita a erros (e.g., registros com nomes abreviados tanto do orientador quanto do orientado, e sem associação do ID identificador). Assim, a obtenção de uma estrutura hierárquica correta que represente os parentescos acadêmicos torna-se um problema não-trivial.

Neste artigo apresentamos um algoritmo desenvolvido com o objetivo de extrair, de forma automática e com alto grau de corretude, grafos de genealogia acadêmica a partir de bancos de dados que possuam informações explicitas e/ou implícitas sobre relacionamentos de orientação acadêmica. Consideramos os currículos de pesquisadores em nível de Doutorado registrados da Plataforma Lattes como estudo de caso. Os principais desafios encontrados na estruturação do grafo de genealogia foram: (i) devido à falta de declaração explícita da orientação, atribuir um relacionamento por meio do casamento exato e inexato de nomes, (ii) desambiguação de nomes grafados de formas diferentes, e (iii) tratamento de inconsistências das informações registradas.

A aplicação do algoritmo desenvolvido possibilitou a estruturação do grafo de genealogia de todos os doutores registrados, de forma explícita ou implícita, na Plataforma Lattes. O grafo obtido é composto por mais de 334 mil vértices (doutores) e 300 mil arestas (relacionamentos de orientação), que foram quantificados em função das grandes áreas do conhecimento. Adicionalmente, foram aplicadas métricas genealógicas (descendência, fecundidade e índice genealógico) para caracterizar os indivíduos e os grupos em função de sua capacidade de orientação acadêmica. Os resultados apresentados evidenciam informações inéditas da Ciência Brasileira sob a perspectiva de formação de recursos humanos.

2. Trabalhos correlatos

O crescente interesse da comunidade científica, sobre trabalhos que considerem a GA, motiva o desenvolvimento de pesquisas sobre a extração de grafos de genealogia a partir de bancos de dados. O trabalho de [Dores et al. 2016] apresenta um método semi-automático que tem como objetivo específico a extração deste tipo de estrutura. Foi considerado como fonte de dados o *Networked Digital Library of Theses and Disser-*

[1] https://genealogy.math.ndsu.nodak.edu, último acesso em 27 de maio de 2017.
[2] https://academictree.org, último acesso em 27 de maio de 2017.

tations – NDLTD[3]. Do total de registros desta base (4 588 474), à época do artigo, os autores construíram o grafo considerando 638 812 (14%). Os autores apontam a ausência de informações e os erros de grafia no banco como as causas para este baixo percentual. Além da obtenção do grafo de genealogia, o referido trabalho apresenta uma caracterização por meio de métricas topológicas específicas.

Grande parte das pesquisas sobre GA se concentram na estruturação semi-automática do grafo e em sua caracterização. [Malmgren et al. 2010] é um dos pioneiros na análise deste tipo de estrutura. Este trabalho avalia o impacto da orientação acadêmica nos orientados, utilizando dados do *MGP* e caracterização por meio de métricas. No trabalho de [Gargiulo et al. 2016] o objetivo é a identificação da origem histórica dos matemáticos, também pela estruturação de grafos de genealogia. Este mesmo conjunto de dados foi utilizado por [Rossi et al. 2017] para caracterização de matemáticos através de uma nova métrica denominada índice genealógico. A plataforma *Web* denominada *The Academic Family Tree* é outra importante fonte de dados genealógicos que abrange diferentes áreas do conhecimento. Esta base de dados foi considerada no trabalho de [David and Hayden 2012] que estruturou o grafo de genealogia dos neurocientistas e promoveu a análise da estrutura com o uso de métricas.

No contexto nacional, a Plataforma Lattes foi utilizada como fonte de dados para identificar grafos de genealogia honoríficas de pesquisadores destacados ou de determinadas áreas do conhecimento [Elias et al. 2016]. É importante destacar que a coleta de dados nesses trabalhos foi realizada de forma semi-manual. Diferente dos trabalhos anteriormente descritos, este artigo apresenta como objetivo a proposição de um algoritmo que obtenha, de forma automática, grafos de genealogia acadêmica de fontes de dados curriculares, como a Plataforma Lattes. No nosso entendimento, trata-se de uma proposta inédita, tanto pela tratativa das inconsistências observadas, quanto pela precisão do grafo de genealogia acadêmica obtido.

3. Métodos

Neste trabalho definimos um algoritmo para identificação de grafo de genealogia acadêmica, obtido a partir de dados da Plataforma Lattes. Foram realizadas quatro etapas: (i) coleta, leitura e normalização dos dados, (ii) identificação de relações explícitas, (iii) identificação de relações implícitas, e (iv) redução de inconsistências e validação do grafo. A seguir, descrevemos estas etapas e, ao final, apresentamos o algoritmo que sumariza o processo de obtenção do grafo de genealogia.

3.1. Coleta, leitura e normalização dos dados

Os dados prospectados da Plataforma Lattes foram utilizados para a construção do grafo de genealogia acadêmica, onde os vértices representam os pesquisadores (com nível de Doutorado, Pós-Doutorado ou Livre-Docência) e as arestas direcionadas os relacionamentos de orientação entre eles. Os atributos disponíveis para cada registro são: identificador numérico exclusivo, nome completo, endereço, área de pesquisa, formação acadêmica, vínculo institucional, orientações em andamento e concluídas, publicações, dentre outros.

Após a coleta de todos os currículos disponíveis, os dados foram submetidos a um processo de normalização, que consiste em detectar e eliminar erros e/ou in-

[3]http://www.ndltd.org, último acesso em 27 de maio de 2017.

consistências nas informações, como definido nos trabalhos de [Fayyad et al. 1996] e [Rahm and Do 2000]. Todos os nomes de pesquisadores foram submetidos à eliminação de espaços em branco duplicados, acentos, pontuações e prefixos no início dos nomes completos (e.g., "Dr.", "Profa."), e à transformação dos caracteres para caixa-baixa. É importante frisar que nomes abreviados ou grafados de forma diferente podem acarretar em erros nas atribuições dos relacionamentos de orientação. O algoritmo apresentado neste trabalho permite lidar de modo eficiente com nomes grafados de forma similar.

3.2. Identificação de relações explícitas

As relações explícitas de orientação são aquelas onde o nome de um pesquisador e seu respectivo número identificador[4] estão presentes nos campos de "Formação Acadêmica" e "Orientações Concluídas" de cada currículo. No primeiro campo, há informações sobre o nível de formação do pesquisador, bem como o registro de seu orientador e coorientador, que são identificados por meio dos nomes completos e os números identificadores. Usamos estas informações para identificar a ascendência explícita de cada pesquisador.

O campo "Orientações Concluídas" contém os registros das orientações finalizadas pelo pesquisador. Podem ser consideradas as orientações e coorientações concluídas em nível de Iniciação Científica, Mestrado, Doutorado e Pós-doutorado. As informações disponíveis no registro de orientação são: nome completo do orientado, anos de início e conclusão da orientação, e número identificador do orientado. Extraímos os dados somente das orientações concluídas em nível de Doutorado.

As relações identificadas (por ascendência ou descendência) representam arestas no grafo de genealogia, e seus atributos disponíveis são: número identificador do vértice orientador, número identificador do vértice orientado, ano de início, ano de conclusão, nome completo do orientador e nome completo do orientado. Cada vértice representa um pesquisador com os seguintes atributos: número identificador, nome completo e nome da Grande Área do Conhecimento.

3.3. Identificação de relações implícitas

Para os registros nos quais o número identificador de um pesquisador está ausente (nos campos "Formação Acadêmica" e/ou "Orientações Concluídas"), utilizamos os nomes completos do orientador e orientado para fazer a identificação do relacionamento (i.e., deduplicação de instâncias[5]). Consideramos casamentos exatos, i.e., aqueles nos quais os nomes não apresentam diferenças em nenhum caractere e inexatos, ou seja, por até um caractere de distância de edição, conforme método proposto por [Levenshtein 1966].

Estes casamentos foram aplicados em dois escopos diferentes do grafo, primeiro de forma global, usando os nomes completos (como registrados na Plataforma Lattes) dos pesquisadores e por último, de forma local usando o nome e sobrenome (ignorando os termos intermediários) dos pesquisadores. Para os casamentos em escopo global, cada nome completo foi comparado com registros de um dicionário de nomes e identificadores numéricos, criado previamente com informações obtidas da Plataforma Lattes. Nos casamentos em escopo local, a comparação de nomes se deu entre irmãos acadêmicos.

[4]Um número identificar na Plataforma Lattes está composto de 16 algarismos e é um valor único associado a cada pesquisador registrado na Plataforma.

[5]O tratamento de ambiguidades é um problema importante que afeta a qualidade dos dados coletados [Kim et al. 2014, Ferreira et al. 2014].

Nos casos em que o casamento de nomes não fosse efetivo (tanto por casamento exato quanto por inexato) ou por haver homônimos nestes casamentos, novos vértices, representando pesquisadores não encontrados na Plataforma Lattes, foram construídos de modo artificial. Isto foi feito com o intuito de manter de forma fiel o número de orientações registradas por cada pesquisador. Nestes casos, os vértices novos não contém atributos como Grande Área e dados de formação acadêmica, bem como o próprio número identificador (sendo atribuído um valor artificial). Nas relações implícitas as orientações identificadas também representam arestas no grafo de genealogia.

3.4. Redução de inconsistências e validação do grafo

Esta etapa consiste na eliminação de vértices artificiais com nomes inconsistentes, na unificação de arestas duplicadas e na validação amostral do grafo. Primeiramente, submetemos os vértices artificiais a dois filtros: (a) nomes de tamanho menor que seis caracteres ou compostos por apenas um termo e (b) nomes presentes em uma lista de termos pré-definida (e.g., "a informar", "sem orientador", "não tinha na época"). Removemos os vértices artificiais e suas arestas, caso os nomes se enquadrassem nestes filtros.

Como o algoritmo coleta tanto as relações apontadas na ascendência quanto na descendência, ocorre de uma mesma relação ser apontada duas vezes, uma pelo orientador (em "Orientações Concluídas" do currículo do orientador) e outra pelo orientado (em "Formação Acadêmica" do currículo do orientado). Ainda, pode ocorrer de um pesquisador relacionar-se com outro mais de uma vez, i.e., ter mais de um Doutorado concluído com o mesmo orientador, gerando também arestas duplicadas. Nestes casos, as arestas foram unificadas considerando-se a mais antiga (ano de conclusão da relação), pois representam a mesma entidade no grafo de genealogia.

Por fim, fizemos uma validação amostral do grafo, junto aos currículos de 10 pesquisadores. Essa validação consistiu em verificar manualmente de qual currículo surgiu cada relação (ou de "Orientações Concluídas" do próprio currículo ou de "Formação Acadêmica" do currículo de outro pesquisador) e quantas são implícitas, explícitas ou transformadas, por meio de casamento, de implícitas para explícitas (ver Seção 4).

Algoritmo para identificação do grafo de genealogia acadêmica

O principal procedimento BUILD-ACADEMIC-GENEALOGICAL-GRAPH requer como parâmetro inicial uma lista S composta por números identificadores de currículos de pesquisadores, e chama procedimentos auxiliares para tratar cada currículo coletado. Como saída, produz uma lista de vértices (V) e arestas (E) que representam o grafo de genealogia acadêmica identificado. Um dos procedimentos auxiliares é GET-RELATIONSHIP, cuja função básica é extrair do currículo coletado os relacionamentos ($i_{relationships}$) de ascendência e descendência descritos nos campos relacionados à "Formação Acadêmica" e a "Orientações Concluídas", respectivamente.

Para cada relacionamento extraído, outro procedimento auxiliar (GET-IDENTIFICATION) busca o número identificador ($r_{identification}$) do orientado (descendência) ou do orientador (ascendência). Caso sejam relações explícitas, o número identificador do pesquisador está descrito no currículo, e portanto, a sua obtenção é trivial. Caso sejam relações implícitas, este número está ausente e o procedimento usa o nome completo do pesquisador para tentar fazer casamento junto a um dicionário (D) de nomes completos e identificadores numéricos, criado previamente.

```
BUILD-ACADEMIC-GENEALOGICAL-GRAPH(S)
1   V = E = {}
2   while |S| ≠ 0
3     for each i ∈ S
4       if i ∉ V
5         INSERT(V, i)
6         i_relationships ← GET-RELATIONSHIPS(i)
7         for each r ∈ i_relationships
8           r_identification ← GET-IDENTIFICATION(r)
9           if r_identification ∉ V
10            INSERT(V, r_identification)
11            if r_identification not artificial
12              INSERT(S, r_identification)
13          if r ∈ E
14            r_older ← GET-OLDER-EDGE(r, E)
15            INSERT(E, r_older)
16          else
17            INSERT(E, r)
18    REMOVE(S, i)
19  LOCAL-MATCHING(V, E)
20  FILTER-NODES(V, E)
21  return (V, E)

GET-IDENTIFICATION(r, D)
1   if r is implicit
2     r_identification ← PERFECT-MATCHING(r, D)
3     if r_identification = NIL
4       r_identification ← LEVENSHTEIN-MATCHING(r, D)
5     if r_identification = NIL
6       r_identification ← new artificial identifier
7   return r_identification
```

Este casamento pode ser de duas formas, ou PERFECT-MATCHING ou LEVENSHTEIN-MATCHING. Primeiramente, tenta-se fazer o casamento exato, e em caso de falha, procede-se ao inexato, usando, em ambos os casos, o nome completo do pesquisador. Caso não ocorra casamento, um vértice artificial é criado para o nome, evitando assim que o número de orientações realizadas seja menor do que a descrita nos currículos.

O procedimento GET-OLDER-EDGE certifica de manter em E a aresta mais antiga entre dois pesquisadores, comparando, para isso, o ano de conclusão dos registros. Durante a execução do procedimento principal, novos números identificadores são adicionados à lista S (somente os não artificiais) e outros são excluídos, conforme são tratados os relacionamentos. O laço principal do algoritmo só encerra após não haver mais números identificadores para serem verificados. Após, o procedimento LOCAL-MATCHING, tenta mesclar vértices artificiais com seus irmãos, comparando neste caso, o nome e sobrenome (os termos intermediários são ignorados) dos vértices em questão. Isso é feito com o intuito de reduzir o número de vértices artificiais duplicados e que representam, possivelmente, o mesmo pesquisador (nome abreviado). Por fim, o procedimento FILTER-NODES exclui vértices artificiais com nomes inconsistentes e as arestas a eles relacionadas.

A principal vantagem da utilização deste algoritmo é a capacidade de transformar relações implícitas de orientação em explícitas, por meio de deduplicação de nomes. Com isso, obtém-se um grafo mais completo, em que há mais relacionamentos informativos quando comparado à estrutura obtida sem essa transformação.

4. Resultados

O processo anteriormente descrito foi aplicado no conjunto de currículos disponíveis na Plataforma Lattes. Apresentamos, a seguir, os resultados obtidos, onde são detalhados os dados considerados, a validação manual e a caracterização do grafo conforme subconjuntos que representam as Grandes Áreas do Conhecimento. Adicionalmente, com objetivo estritamente ilustrativo, apresentamos uma análise genealógica do grafo obtido.

4.1. Conjunto de dados e validação das relações de orientação

Os dados de 272 165 currículos foram coletados da Plataforma Lattes em Maio de 2017[6]. A Tabela 1 apresenta as principais estatísticas que descrevem tanto os dados obtidos quanto o grafo resultante. A Tabela 2 descreve os resultados referentes ao processo de casamento de nomes nos escopos global e local, aplicados nos vértices cujos identificadores numéricos estavam ausentes.

Foi possível inferir, ou por casamento exato ou por inexato, registros da Plataforma Lattes para 57,3% do total de nomes sem número identificador (148 980), o que reduziu a quantidade de vértices artificiais e ampliou a qualidade e quantidade das informações da estrutura obtida.

É importante observar que mantivemos no grafo os vértices referentes a nomes que não foram identificados explicitamente. Apresentamos na Figura 1 a quantidade destes vértices por graus de entrada e saída. Nota-se que a maior parte (93,2%) deles possui grau de saída de no máximo 1. Estes casos foram gerados pelo campo "Formação Acadêmica" dos currículos, onde os pesquisadores indicaram seus orientadores, mas não os vincularam a um identificador numérico (ou pelo orientador não possuir currículo Lattes ou pelo nome grafado ser diferente do cadastrado na plataforma).

A Tabela 3 apresenta a análise amostral de descendência, dos currículos de 10 pesquisadores comparando-os com o grafo identificado. Para estes pesquisadores boa parte dos relacionamentos implícitos foram convertidos em explícitos. O número final de descendentes no grafo foi maior para os 10 pesquisadores, quando comparado ao número registrado nos currículos. Isto ocorreu porque estes pesquisadores não registraram em seus currículos alguns descendentes, que os registraram como orientadores.

[6]A extração de dados foi automática seguindo as informações de acesso oferecidas pela Plataforma Lattes: http://memoria.cnpq.br/web/portal-lattes/extracoes-de-dados

Tabela 1. Estatísticas referentes aos dados considerados e ao grafo resultante

Currículos disponíveis na Plataforma Lattes	5 102 445	100,00%
Currículos considerados neste trabalho (doutores)	272 165	5,33%
Vértices no grafo obtido	334 952	–
Arestas no grafo obtido	300 696	–
Componentes conexas no grafo obtido	37 444	–
Vértices na maior componente conexa	178 767	53,4%
Arestas na maior componente conexa	181 731	60,4%
Vértices isolados (doutores sem orientador nem orientados)	4 842	1,45%
Densidade do grafo	$2{,}68 \times 10^{-6}$	–
Grau médio do grafo	1,79	–
Grau máximo do grafo	130	–

Tabela 2. Estatísticas referentes ao processo de casamento de nomes

Nomes sem número identificador	148 980	100, 00%
Nomes identificados por casamento global exato	71 307	47, 86%
Nomes identificados por casamento global inexato	7 024	4, 71%
Nomes identificados por casamento local	7 033	4, 72%
Nomes inconsistentes excluídos	426	0, 29%
Nomes não identificados (vértices artificiais)	63 190	42, 41%

4.2. Caracterização do grafo em termos de Grande Área

O grafo obtido pode ser utilizado como insumo para estudo da multidisciplinaridade na orientação acadêmica. Para isso, associamos as grandes áreas de conhecimento para cada pesquisador. A Figura 2 mostra a proporção de pesquisadores e de orientações realizadas e recebidas, por Grande Área. Há ao todo nove grandes áreas, conforme descrito pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES[7]): Ciências Agrárias (AGR), Ciências Biológicas (BIO), Ciências da Saúde (SAU), Ciências Exatas e da Terra (EXA), Ciências Humanas (HUM), Ciências Sociais Aplicadas (SOC), Engenharias (ENG), Linguística, Letras e Artes (LIN) e Outros (OUT). Acrescentamos a Grande Área Indefinido (IND) para rotular os casos em que esta informação estava ausente nos currículos e nos vértices artificiais.

Excluindo-se a categoria Indefinido, nota-se que as grandes áreas com maior número absoluto de orientações realizadas são HUM com 37 685 orientações (17,98%) e SAU com 34 812 orientações (16,61%). As que possuem maior número de orientações recebidas são SAU com 39 005 orientações (18,61%) e HUM com 34 557 orientações (16,49%). As grandes áreas que possuem maior número de pesquisadores doutores são SAU com 43 831 pesquisadores (17,36%) e HUM com 42 417 pesquisadores (16,80%).

A Figura 3 mostra a influência recebida e exercida por Grande Área. As linhas vermelhas indicam a proporção de orientações recebidas, ou seja, indicam o quanto cada Grande Área influenciou a área em questão. De forma similar, as linhas verdes representam o quanto a Grande Área em questão influenciou as demais. Cabe destacar que,

[7]Tabela de áreas do conhecimento da CAPES, disponível em http://www.capes.gov.br/avaliacao/instrumentos-de-apoio/tabela-de-areas-do-conhecimento-avaliacao, último acesso em 27 Mai 2017.

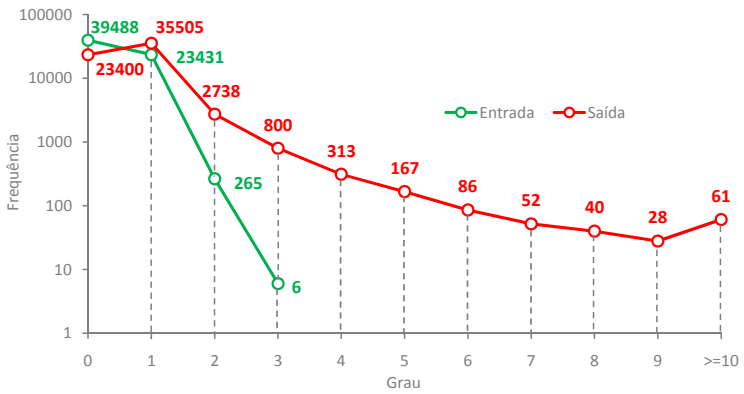


Figura 1. Quantidade de vértices artificiais quanto aos graus de entrada e saída

Tabela 3. Comparação de descendentes identificados nos currículos e no grafo

	Currículo Lattes					Grafo obtido				
	Total	Explícito		Implícito		Total	Explícito		Implícito	
F. A. P. Fialho	119	62	52,10%	57	47,90%	129	124	96,12%	5	3,88%
N. F. F. Ebecken	124	67	54,03%	57	45,97%	129	120	93,02%	9	6,98%
M. L. S. Braga	114	78	68,42%	36	31,58%	119	106	89,08%	13	10,92%
C. A. N. Cosenza	104	29	27,88%	75	72,12%	111	90	81,08%	21	18,92%
O. R. Gottilieb	66	1	1,52%	65	98,48%	98	35	35,71%	63	64,29%
P. de B. Carvalho	85	61	71,76%	24	28,24%	93	88	94,62%	5	5,38%
G. P. Witter	82	16	19,51%	66	80,49%	88	59	67,05%	29	32,95%
J. L. de Azevedo	79	36	45,57%	43	54,43%	85	65	76,47%	20	23,53%
E. L. da Silva	67	44	65,67%	23	34,33%	82	73	89,02%	9	10,98%
R. M. Filho	76	61	80,26%	15	19,74%	78	74	94,87%	4	5,13%

apesar dos maiores números de orientados estar presente entre as mesmas grandes áreas, há considerável número de relações entre grandes áreas distintas, evidenciando assim a multidisciplinaridade na orientação acadêmica. Exemplos desta multidisciplinaridade ocorrem entre BIO e SAU (4 603 orientações), e entre HUM e SOC (3 658 orientações).

4.3. Caracterização do grafo em termos genealógicos

Para ilustrar as possibilidades analíticas deste tipo de estrutura, consideramos um conjunto de métricas genealógicas desenvolvidas para este objetivo. Neste exemplo, utilizamos três métricas para a caracterização genealógica do grafo [Rossi and Mena-Chalco 2014]. A métrica *fecundidade* representa o total de descendentes, em todas as gerações, que um acadêmico de interesse possui, a métrica *descendência* captura o número de alunos que um acadêmico de interesse orientou no decorrer de sua carreira e a métrica *índice genealógico* [Rossi et al. 2017] é uma adaptação da métrica bibliométrica índice-h, e considera a quantidade (descendentes) e qualidade (abrangência das gerações) dos relacionamentos. O índice genealógico g de um acadêmico é o maior número g de descendentes que, por sua vez, possuem, no mínimo, g descendentes cada um.

As métricas foram calculadas para os vértices do grafo obtido pelo algoritmo proposto. Considerando o conjunto completo, os acadêmicos apresentam, em média, 9,6 descendentes em sua linhagem (fecundidade), 3,7 descendentes diretos (descendência)

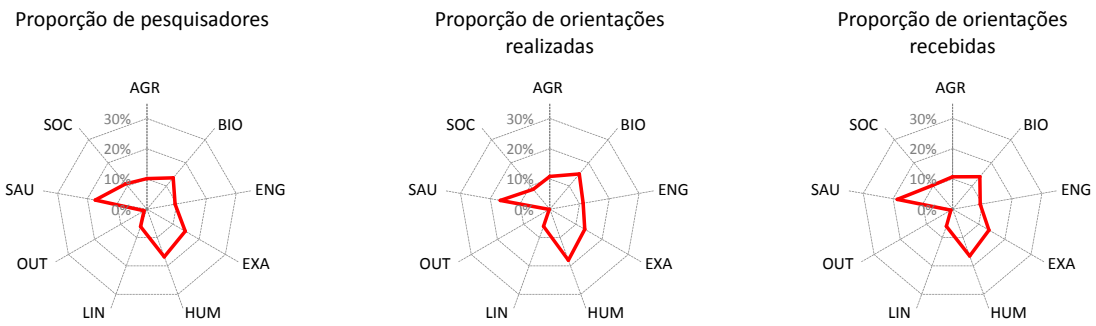


Figura 2. Proporções dos pesquisadores e orientações para cada Grande Área

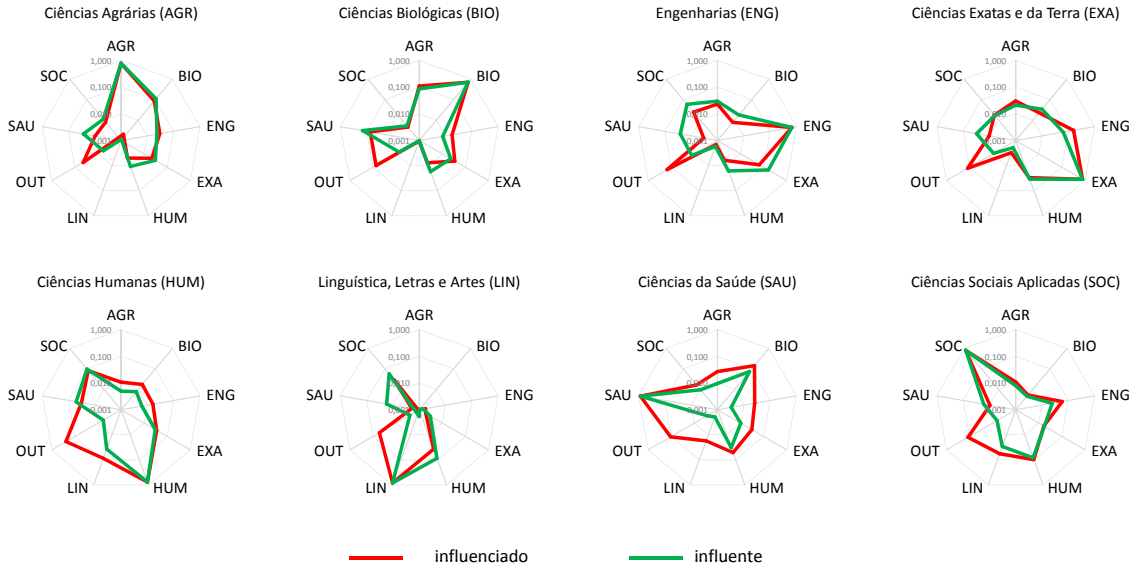


Figura 3. Interdisciplinaridade segundo as orientações observadas entre as Grandes Áreas. Os eixos estão em escala logarítmica.

e o índice genealógico médio é igual a 0,4. Estes resultados consideram somente os acadêmicos férteis, ou seja, não consideramos os vértices com descendência igual a zero; o conjunto resultante totaliza 80 479 (24,0%) acadêmicos. Na Tabela 4 apresentamos os resultados obtidos, estratificados por Grande Área.

Considerando os resultados individuais podemos destacar o acadêmico Joel Martins (BIO), que apresenta 1 747 acadêmicos em sua linhagem e o acadêmico Francisco Antonio Pereira Fialho (HUM), que orientou 129 alunos. Os professores Geraldina Porto Witter (HUM), Cidmar Teodoro Pais (LIN) e Carolina Martuscelli Bori (IND) são os acadêmicos com maior índice genealógico, cujo valor é igual a 12.

Finalmente, na Tabela 5 apresentamos os 10 acadêmicos mais relevantes, de acordo com os resultados das métricas, aplicadas ao grafo de genealogia. Para o índice genealógico, os seguintes acadêmicos também apresentam resultado igual a 10: Eduardo

Tabela 4. Métricas estratificadas por Grande Área (segundo a Plataforma Lattes)

	Fecundidade			Descendência			Índice genealógico			Acadêmicos
	média	mediana	máximo	média	mediana	máximo	média	mediana	máximo	considerados
AGR	11,81	6	388	7,07	5	63	0,44	0	8	14,37%
BIO	14,58	5	1747	6,59	4	85	0,53	0	11	16,57%
ENG	11,07	5	474	6,62	4	129	0,43	0	9	17,87%
EXA	9,69	4	865	5,47	3	98	0,42	0	11	17,12%
HUM	13,02	5	989	6,88	4	129	0,48	0	12	14,85%
IND	7,52	1	1536	1,30	1	48	0,35	0	12	49,98%
LIN	12,84	5	975	6,71	5	119	0,47	0	12	14,16%
OUT	4,09	2	54	2,47	1	31	0,18	0	4	8,58%
SAU	11,54	4	1003	5,98	4	70	0,48	0	10	15,75%
SOC	10,75	4	877	5,70	3	93	0,40	0	8	13,50%

Tabela 5. Acadêmicos com os 10 melhores resultados em cada métrica

Fecundidade			Descendência			Índice genealógico		
Acadêmico	Área	Valor	Acadêmico	Área	Valor	Acadêmico	Área	Valor
J. Martins	BIO	1747	F. A. P. Fialho	HUM	129	G. P. Witter	HUM	12
F. G. Brieger	IND	1536	N. F. F. Ebecken	ENG	129	C. T. Pais	LIN	12
A. Dreyfus	IND	1337	M. L. S. Braga	LIN	119	C. M. Bori	IND	12
C. Pavan	BIO	1336	C. A. N. Cosenza	ENG	111	O. R. Gottlieb	EXA	11
C. M. Bori	IND	1184	O. R. Gottlieb	EXA	98	J. L. de Azevedo	BIO	11
A. Coutinho	IND	1031	P. de B.Carvalho	SOC	93	D. Saviani	HUM	11
B. Pottier	IND	1020	G. P. Witter	HUM	88	M. de S. Chaui	HUM	11
J. P. Lima	IND	1004	J. L. de Azevedo	BIO	85	J. Martins	BIO	11
G. M. Böhm	SAU	1003	E. L. da Silva	EXA	82	E. M. Portella	LIN	11
L. Pereira	HUM	989	R. M. Filho	ENG	78	E. F. de A. Neves	EXA	10

Moacyr Krieger, Gabriel Cohn, Ivan Antônio Izquierdo e Massayoshi Yoshida[8]

5. Conclusão

A evolução do conhecimento científico é a base para o desenvolvimento sócio-econômico de uma comunidade. A ciência evolui por meio da interação entre acadêmicos e pela transferência de conhecimento que é realizada nestas interações. Bancos de dados inseridos neste contexto são importantes fontes para identificação do impacto de pesquisadores na comunidade acadêmica, bem como sobre a formação de comunidades e a evolução do conhecimento de modo abrangente. Considerando uma base de dados curriculares de pesquisadores, desenvolvemos um algoritmo que identifica um grafo de forma automática. As relações de ascendência e descendência descritas nestes currículos, por vezes não continham identificadores únicos, e o algoritmo, por meio de deduplicação, permitiu expandir o grafo para uma maior quantidade de pesquisadores e relações explícitas, melhorando a qualidade da estrutura obtida.

Analisamos este grafo sob a ótica da GA analítica [Sugimoto et al. 2011], considerando as grandes áreas do conhecimento, em função de seu desempenho individual e das interações observadas entre elas. Observamos que as grandes áreas apresentam um equilíbrio de desempenho, capturado pelas métricas, na formação de recursos humanos. Identificamos, de forma preliminar, as interações existentes entre as áreas, que reflete na multidisciplinaridade da jovem comunidade acadêmica no Brasil.

Diferentes são as frentes a serem consideradas como trabalhos futuros. Dentre elas destacamos: (i) a ampliação da base de dados para pesquisadores com nível de Mestrado, (ii) a utilização de técnicas de deduplicação mais sofisticadas, e (iii) a geração de redes mesoescala para estudar longitudinalmente a evolução, no tempo, das relações de orientação, conforme outras informações dos pesquisadores, como instituição e gênero.

Agradecimentos

Os autores agradecem à UFABC, ao CNPq e à CAPES pelo apoio financeiro ao projeto de pesquisa. Os autores também agradecem aos revisores anônimos, cujas sugestões foram muito relevantes para a melhora do artigo.

[8]A grafia dos nomes foi mantida da mesma forma disponível na Plataforma Lattes.

Referências

[David and Hayden 2012] David, S. V. and Hayden, B. Y. (2012). Neurotree: A collaborative, graphical database of the academic genealogy of neuroscience. *PloS one*, 7(10):e46608.

[Dores et al. 2016] Dores, W., Benevenuto, F., and Laender, A. H. F. (2016). Extracting academic genealogy trees from the networked digital library of theses and dissertations. In *Digital Libraries, 2016 IEEE/ACM Joint Conference on*, pages 163–166.

[Elias et al. 2016] Elias, M., Floeter-Winter, L. M., and Mena-Chalco, J. P. (2016). The dynamics of brazilian protozoology over the past century. *Memórias do Instituto Oswaldo Cruz*, 111(1):67–74.

[Fayyad et al. 1996] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37.

[Ferreira et al. 2014] Ferreira, A. A., Gonçalves, M. A., and Laender, A. H. F. (2014). Disambiguating author names using minimum bibliographic information. *World Digital Libraries*, 7(1):71.

[Gargiulo et al. 2016] Gargiulo, F., Caen, A., and Carletti, R. L. T. (2016). The classical origin of modern mathematics. *EPJ Data Science*, 5(1):26.

[Kim et al. 2014] Kim, J., Diesner, J., Aleyasen, A., Heejun, K., and Hwan-Min, K. (2014). Why name ambiguity resolution matters for scholarly big data research. In *2014 IEEE International Conference on Big Data*, pages 1–6.

[Korfhage 1997] Korfhage, R. R. (1997). *Information Storage and Retrieval*. Wiley, 1st edition.

[Lane 2010] Lane, J. (2010). Let's make science metrics more scientific. *Nature*, 464(7288):488–489.

[Levenshtein 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.

[Malmgren et al. 2010] Malmgren, R. D., Ottino, J. M., and Amaral, L. A. N. (2010). The role of mentorship in protégé performance. *Nature*, 465(7298):622–626.

[Mena-Chalco and Cesar Junior 2013] Mena-Chalco, J. P. and Cesar Junior, R. (2013). Prospecção de dados acadêmicos de currículos lattes através de scriptlattes. In *Bibliometria e Cientometria: reflexões teóricas e interfaces*, pages 109–128.

[Rahm and Do 2000] Rahm, E. and Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13.

[Rossi et al. 2017] Rossi, L., Freire, I. L., and Mena-Chalco, J. P. (2017). Genealogical index: A metric to analyze advisor-advisee relationships. *Journal of Informetrics*, 11(2):564–582.

[Rossi and Mena-Chalco 2014] Rossi, L. and Mena-Chalco, J. P. (2014). Caracterização de árvores de genealogia acadêmica por meio de métricas em grafos. In *Brazilian Workshop on Social Network Analysis and Mining (BraSNAM)*, pages 1–12, Brasília, DF, Brazil.

[Sugimoto 2014] Sugimoto, C. R. (2014). Academic genealogy. In Cronin, B. and Sugimoto, C. R., editors, *Beyond bibliometrics: Harnessing multidimensional indicators of scholarly impact*, pages 365–382. first edition.

[Sugimoto et al. 2011] Sugimoto, C. R., Ni, C., Russell, T. G., and Bychowski, B. (2011). Academic genealogy as an indicator of interdisciplinarity: An examination of dissertation networks in library and information science. *Journal of the American Society for Information Science and Technology*, 62(9):1808–1828.

paper:170799

Ligações Semânticas Utilizando Predicados SKOS

Ricardo Ávila[1], Salomão Santos[1], David Araújo[1], Vânia Vidal[1], José Macêdo[1]

[1]Universidade Federal do Ceará (UFC) – Fortaleza – Brasil

{ricardoavila,salomaosantos,araujodavid,vvidal,jose.macedo}@lia.ufc.br

Abstract. *Glossaries play a central role in creating better semantic relationship between data sources and facilitating understanding within a domain such as oil and gas Exploration and Production (E&P). This work proposes a type of semantic enrichment between glossaries, using SKOS vocabulary and SPARQL queries. The results were satisfactory, generating predicates skos:exactMatch, skos:broader, skos:narrower and skos:related, which suggests that automatic methods can be used to improve a quality of terminological mappings.*

Resumo. *Glossários desempenham um papel central para a melhoria das relações semânticas entre fontes de dados heterogêneas e para uma melhor compreensão das definições no domínio da Extração e Produção de Petróleo (E&P). Este trabalho propõe o enriquecimento semântico entre glossários utilizando o vocabulário SKOS e as consultas SPARQL. Os resultados foram satisfatórios, gerando predicados skos:exactMatch, skos:broader, skos:narrower e skos:related, o que sugere que os métodos automáticos podem ser usados para melhorar a qualidade dos mapeamentos terminológicos.*

1. Introdução

Serviços interativos como o *Linked Data Mashups* (LDM) são ofertados na *Web* combinando o conteúdo de diferentes fontes de dados em um novo serviço [Rahm et al. 2007]. Desenvolver essa conciliação por meio da construção de um Esquema Global, responsável por representar semanticamente duas ou mais fontes de dados que se deseja integrar é uma tarefa complexa que implica utilizar uma grande quantidade de código específico para acessar cada fonte de dados ou serviço, acarretando alto custo de desenvolvimento e manutenção.

A iniciativa *Linked Data* (LD) apresenta-se como uma solução atraente para o trato dessa problemática, provendo mecanismos para a publicação, a recuperação e a integração de dados distribuídos na *Web* de Dados [Bizer et al. 2007]. A publicação e a conexão de dados de forma estruturada na *Web* permitem a interligação de recursos entre diferentes fontes de dados, possibilitando, assim, a conexão dessas fontes em um espaço global único, possibilitando a inferência e um melhor uso dos dados [Lopes et al. 2016].

A combinação dessas técnicas com tecnologias semânticas modernas, com a utilização de abordagens de modelagem flexíveis e extensíveis, pode permitir a integração de dados anteriormente distintos, provendo facilidades de pesquisa e produzindo novos recursos e serviços de dados neutros em plataforma para a colaboração e a disseminação do conhecimento.

Existem diversas arquiteturas de aplicações de LDM que dependem amplamente de seu uso [Heath and Bizer 2011]. Quanto à integração de dados utilizando LDM, exis-

tem duas abordagens: a abordagem materializada, em que os dados são coletados, persistidos e consultados em uma base de dados centralizada, e a abordagem virtual que possibilita consultas em um conjunto de fontes de dados fixos. Nossa abordagem visa a efetuar a fusão dos glossários, criando uma base de conhecimento materializada no domínio do petróleo.

O *mashup*[1] de glossários no domínio do petróleo visa à integração de vários glossários de termos relacionados à Extração e à Produção de Petróleo (E&P). Essa integração permitirá uma melhor compreensão das relações entre as definições no domínio de produção e a extração de petróleo. As principais contribuições deste trabalho são: *(i)* definir heurísticas de geração de *links* semânticos e *(ii)* facilitar a integração semântica de ontologias no domínio do petróleo.

Para o aprofundamento desta pesquisa, utilizamos *Simple Knowledge Organization System* (SKOS) [Miles et al. 2005] e outros vocabulários para a construção da Ontologia de Domínio. SKOS permite o mapeamento das relações semânticas entre termos, utilizando métodos de raciocínio semânticos para sugerir ligações de uma terminologia para outra, por exemplo, SKOS possibilita a utilização de regras lógicas, como *subsumption* e *transitivity*, para inferir ligações não assertivas com base em ligações assertivas e relações hierárquicas entre termos dentro das terminologias.

Normalmente, se A possui *skos:exactMatch* B, e C *is-a* A, então se pode inferir que C possui *skos:broader* com B. Como SKOS necessita do esforço de especialistas e de severas atividades manuais, propomos o uso de consultas SPARQL [Prud'hommeaux and Seaborne 2008] baseado em regras simples, que identificará automaticamente os mapeamentos entre termos utilizando os predicados SKOS. Na linguagem de consulta SPARQL, o comando CONSTRUCT permite que, a partir de um resultado de pesquisa, seja construído um conjunto de triplas. Desse modo, os novos *links* semânticos serão materializados conforme as regras propostas, com a geração semiautomática de predicados *skos:exactMatch*, *skos:narrower*, *skos:broader* e *skos:related*.

Este artigo está organizado da seguinte forma: Na Seção 2, apresentamos o processo de geração de *Linked Data Mashup*. Na Seção 3, detalhamos o ambiente desenvolvido para a execução de *Linked Data Mashup*. Na Seção 4, denotamos a metodologia proposta. Na seção 5, exibimos os experimentos e os resultados obtidos. Na Seção 6, discutimos os trabalhos relacionados. Por fim, na Seção 7, são delineadas as conclusões e os trabalhos futuros.

2. Processo de Geração de LDM

O processo de criação de um LDM incide inicialmente na modelagem de uma ontologia de domínio e integração semântica [Tran et al. 2014]. A modelagem da ontologia de domínio e a integração semântica dispõe das seguintes etapas: *(i)* Elaboração do modelo conceitual da aplicação em uma Ontologia de Domínio (OD); *(ii)* Descrição de cada fonte de dados por meio de sua respectiva Ontologia Fonte (OF), descrevendo os dados que serão exportados; *(iii)* Mapeamento das correspondências entre a OD e as OFs e *(iv)* os *Links* no formato padrão *Resource Description Framework* (RDF)[2] descobertos entre as Fontes de Dados (FD) distintas determinam as ligações entre as OFs.

[1] Aplicação web que usa conteúdo de mais de uma fonte para criar um novo serviço completo.
[2] https://www.w3.org/RDF/

Tabela 1. Glossários de Termos no Domínio do Petróleo

Glossário	*Link*
Schlumberger Oilfield Glossary	http://www.glossary.oilfield.slb.com
Glossário da Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (ANP)	http://www.anp.gov.br/?id=582
Bureau of Safety and Environmental Enforcement (BSEE)	http://www.bsee.gov/Glossary-of-Terms
PetroWiki - SPE's E&P Glossary	http://petrowiki.org/Category:Glossary
Wikipédia	https://wikipedia.org

Para validar os conceitos propostos, selecionamos os glossários apresentados na Tabela 1 com base em *(i)* busca de repositórios de termos relacionados à Extração e à Produção de Petróleo (E&P), preocupando-nos em selecionar organizações que atuam nesse domínio e *(ii)* fontes de dados disponibilizados em *sites* de acesso ao público ou sem nenhuma restrição de direitos autorais para o uso dos termos para fins de pesquisa. Optamos por partir dessas fontes por considerarmos esses glossários produzidos criteriosamente, com orientação de profissionais que atuam e os criaram para o uso constante e a correta apresentação das informações.

Após a seleção dos glossários, utilizamos a OD e as OFs do *mashup* de glossários, representados nas Figuras 1 e 2, respectivamente. O passo a passo para a captura dos termos nas FDs, a transformação em RDF e o armazenamento são apresentados na Seção 3.

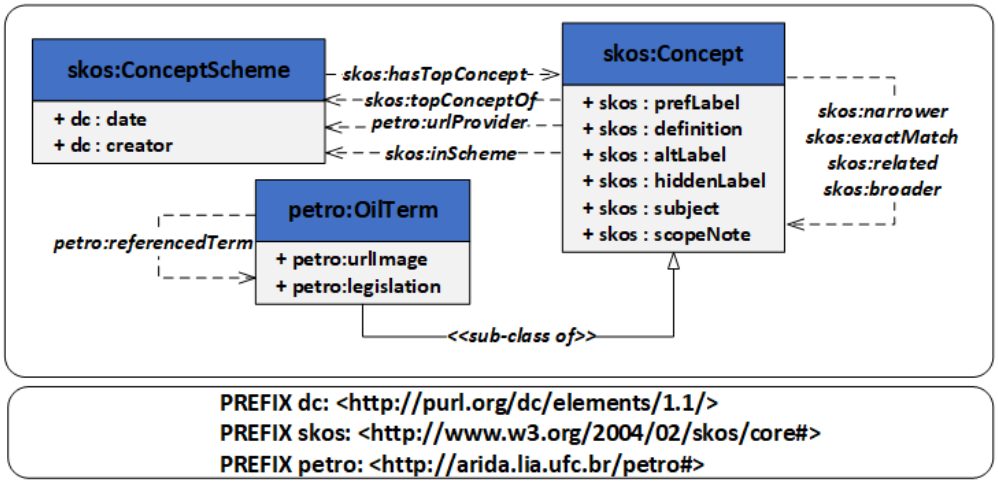


Figura 1. Ontologia de Domínio (OD) do *Mashup* de Glossários

3. Ambiente de Execução de LDM

A partir da seleção dos glossários e da implantação do *mashup*, necessitamos de alguns passos para a viabilização do ambiente. Destarte, utilizamos uma arquitetura visando a unificar as informações das diversas fontes de dados sobre os termos da área de Extração e Produção de Petróleo (E&P).

Os componentes dessa arquitetura possuem o objetivo de coletar os termos das diversas fontes de dados, assim como tratar as informações coletadas para adequá-las a

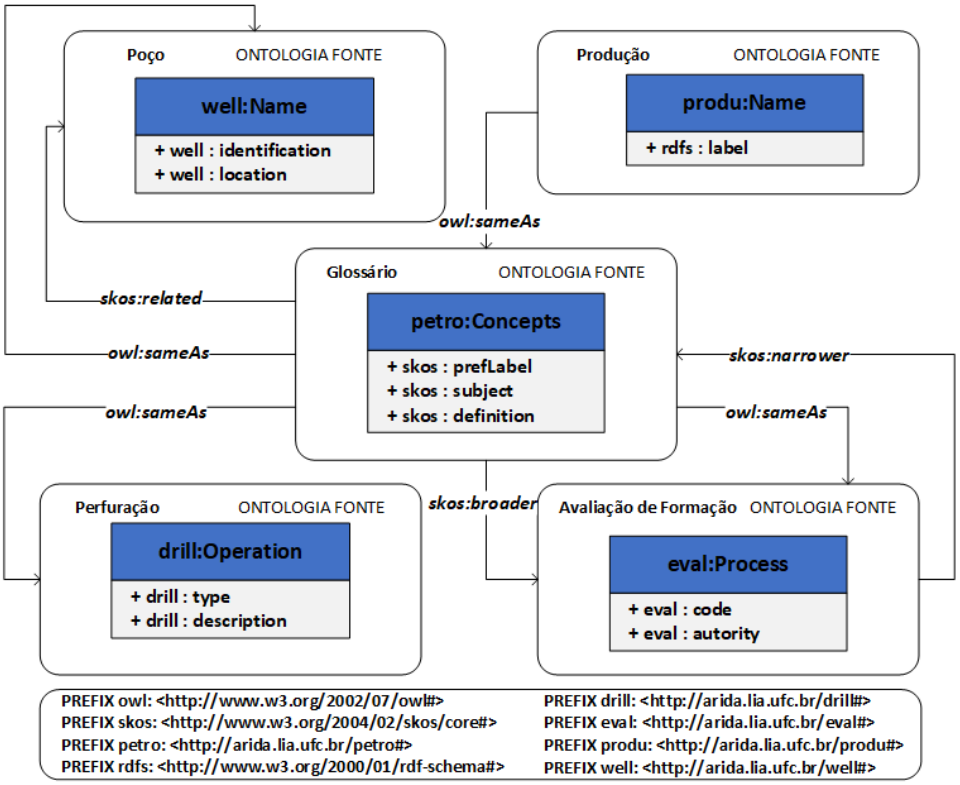


Figura 2. Ontologias Fontes (OFs) do *Mashup* de Glossários

uma mesma estrutura comum, a ontologia de domínio do *mashup* de glossários. As fontes de dados representam os diversos glossários apresentados na Tabela 1.

A partir de cada glossário, um *crawler* é desenvolvido para colher as informações disponíveis. Um *web crawler* é um programa ou *script* automatizado que navega em páginas da *Internet* de maneira metódica e automatizada, permitindo extrair tipos específicos de informação contidos na páginas. Cada *crawler* na arquitetura é responsável, então, por mapear a estrutura dos termos de uma página de glossário para a modelagem da base de documentos; após isso, a coleta é executada, e os termos extraídos são persistidos na base.

O componente base de documentos é responsável por agrupar ao máximo as definições dos diversos glossários para os mesmos termos. Esse processo permite uma inferência inicial entre as estruturas utilizadas pelos diferentes glossários ao definir um mesmo termo. Com isso, o processo de gerar triplas ficou mais rápido e completo. A modelagem definida na base para a persistência das definições foi concebida após a análise da estrutura da informação disponibilizada nas diferentes fontes de glossários, e cada uma delas contém termos relacionados ao domínio e à questão. Portanto, uma união das diferentes estruturas permite a maximização das informações sobre um mesmo termo. Essa base de documentos utiliza o MongoDB[3], e, como a estrutura de armazenamento é baseada em documentos JSON[4], isso facilita a heterogeneidade e a atualização de novos atributos identificados a partir de novos glossários. A modelagem proposta para a estrutura do armazenamento em MongoDB é apresentada na Figura 3.

[3]https://www.mongodb.com
[4]http://www.json.org

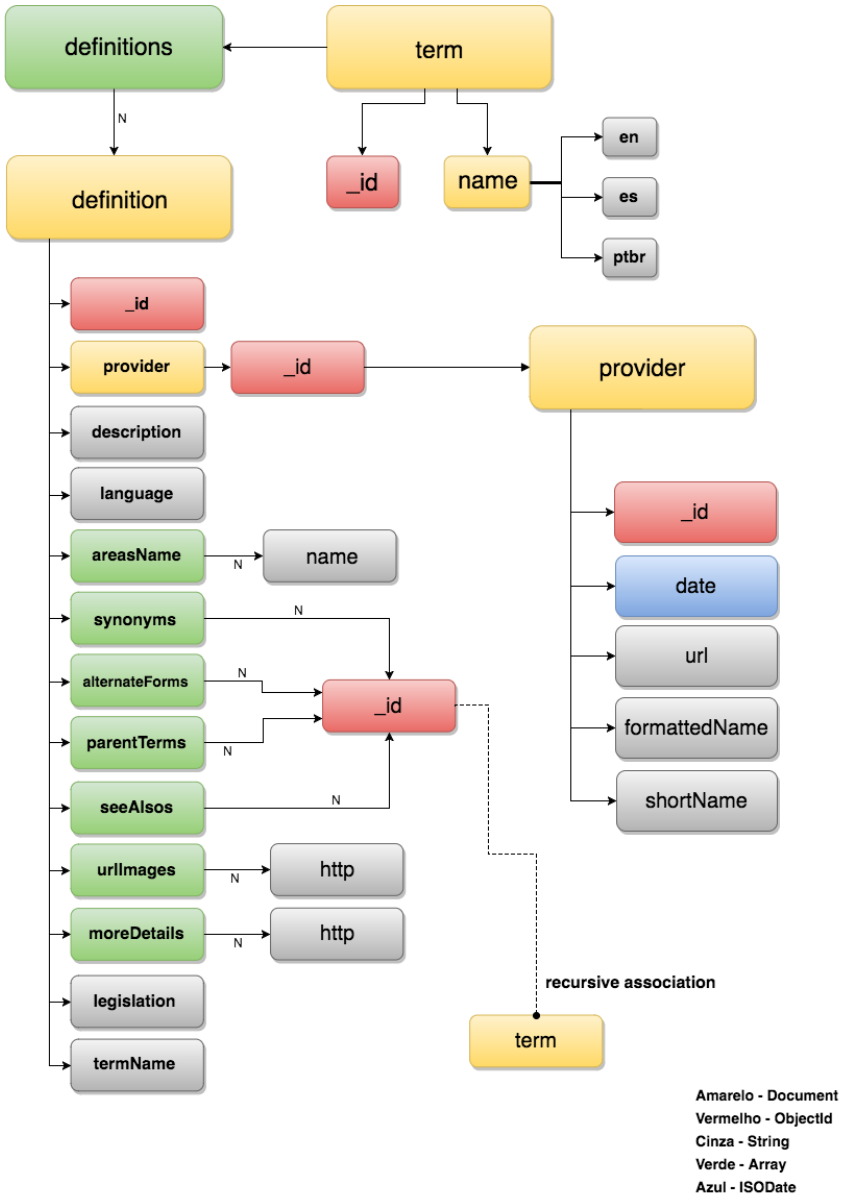


Figura 3. Modelo de Dados Armazenados em MongoDB

O conceito inicial por trás dessa modelagem é agrupar ao máximo as definições dos diversos glossários para os mesmos termos. Esse processo irá futuramente auxiliar o processo de enriquecimento das ontologias. A estrutura utilizada permite a realização de inferências entre definições de diferentes glossários para um mesmo termo. É importante destacar que essa modelagem foi concebida após a análise da estrutura de informação disponibilizadas nos cinco primeiros glossários utilizados, conforme apresentado na Tabela 1. Os diversos atributos apresentados nessa modelagem são a união das diferentes informações extraídas nesses glossários. Destarte, nem todos os termos possuirão todos os atributos preenchidos, dependendo da definição apresentada em cada glossário, sendo algumas mais completas do que outras.

A modelagem proposta facilita a geração das triplas, pois possibilita a estruturação das definições de diferentes glossários em diferentes idiomas para um mesmo termo. Esse agrupamento permite o entendimento dos dados de forma mais clara, visto que não se

encontram mais apenas como textos nas páginas Web dos glossários - que estruturam os dados de forma heterogênea. Com a identificação mais clara dos atributos comuns dos termos, temos um aumento do potencial de geração de *links* semânticos entre eles (i.e.: *skos:exactMatch*, *skos:related*, *skos:broader* e *skos:narrower*).

Os dois últimos componentes são responsáveis pelos processos de triplificação e amazenamento das triplas. O triplificador é um *script* que percorre todos os termos da base de documentos, transformando-os em instâncias triplificadas da ontologia de domínio do *mashup* de glossários. O triplificador foi implementado em *Python* e gera as triplas no padrão N-Triples[5]. Por último, as triplas são armazenadas em uma base RDF ou salvas em arquivos.

4. Metodologia Proposta

Para a geração semiautomática/automática dos predicados SKOS, desenvolvemos consultas SPARQL com o comando CONSTRUCT para validar e avaliar o desempenho do método proposto, utilizando como base o predicado *skos:prefLabel* dos glossários apresentados na Tabela 1. A Figura 4 apresenta o detalhamento visual das consultas desenvolvidas.

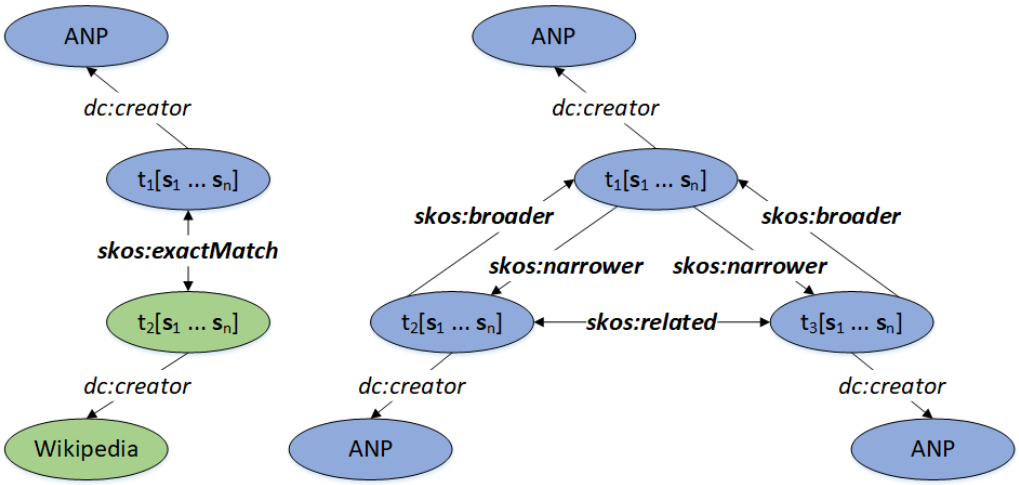


Figura 4. Descoberta de predicados SKOS

As regras das consultas para geração de predicados SKOS seguem os seguintes passos: *(i)* sendo **t**=termo e **s**=*string*, definimos que t_1[**s_1 ... s_n**] e t_2[**s_1 ... s_n**] possui ligação *skos:exactMatch* se t_1 for igual a t_2 e o predicado *dc:creator* de t_1 e t_2 forem diferentes; *(ii)* caso t_1[**s_1**] seja igual a t_2[**s_1** ... s_n] e *dc:creator* de t_1 e t_2 sejam iguais, definimos que t_1 é *skos:narrower* de t_2 e t_2 é *skos:broader* de t_1; *(iii)* finalmente, definimos que haverá ligação reflexiva *skos:related* entre t_2 e t_3 quando t_2[**s_1** ... s_n] e t_3[**s_1** ... s_n] possuírem ligações semânticas *skos:narrower* e *skos:broader* com t_1. A consulta SPARQL para a geração dos links semânticos *skos:narrower* e *skos:broader* é apresentada na Lista 1.

Para o alinhamento do predicado *skos:exactMatch* entre os glossários, utilizamos a ferramenta Silk [Volz et al. 2009]. Durante as experimentações preliminares, os resultados apontaram para o uso do algoritmo Levenshtein que obteve melhor resposta em relação a outros algoritmos como o Jaro-Winkler e o Jaccard, por exemplo. Definimos o

[5]https://www.w3.org/2001/sw/RDFCore/ntriples

limite para a métrica de alinhamento em 0.95, ou seja, dois termos são considerados equivalentes caso a similaridade do conjunto de caracteres que os compõe seja maior ou igual a 0.95. A adequação de diferentes métricas de distância de *strings* tem sido amplamente discutida na literatura [Cohen et al. 2003]. Feito o alinhamento do *skos:exactMatch*, seguimos com os experimentos para os demais predicados.

Esses tipos de relacionamentos são representados de acordo com as seguintes definições: **ET** (*Exact Terms*), definido entre dois termos t_i e t_j, desde que sejam consideradas a mesma palavra. **ET** é simétrico, isto é, t_i **ET** $t_j \Rightarrow t_j$ **ET** t_i. **BT** (*Broader Terms*) definido entre dois termos t_i e t_j, desde que t_i possua um significado mais geral do que t_j. **BT** não é simétrico. O oposto de **BT** é **NT** (*Narrower Terms*): t_i **NT** $t_j \Rightarrow t_j$ **BT** t_i. **RT** (*Related Terms*) definido entre dois termos t_i e t_j, que são geralmente utilizados em conjunto no mesmo contexto. **RT** é simétrico: t_i **RT** $t_j \Rightarrow t_j$ **RT** t_i. A descoberta desses relacionamentos terminológicos, a partir de esquemas de fonte, é uma atividade semiautomática proposta neste trabalho.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

CONSTRUCT

{
   ?s1 <http://www.w3.org/2004/02/skos/core#narrower> ?s2 .
   ?s2 <http://www.w3.org/2004/02/skos/core#broader> ?s1 .
}

WHERE {

  ?s1 skos:prefLabel  ?l1 .
  ?s2 skos:prefLabel  ?l2 .

  FILTER (lang(?l1) = 'en' &&
        !CONTAINS(?l1, "␣") &&
        lang(?l2) = 'en' &&
        !SAMETERM(?l1, ?l2))

  BIND(CONCAT("␣",STR(?l1)) AS ?label_1)

  FILTER(STRENDS(?l2,?label_1))

}
```

Lista 1. Consulta SPARQL para geração de *links skos:narrower* e *skos:broader*

O predicado *skos:related* permite a representação de *links* associativos (não hierárquicos), como a relação entre um tipo de evento e uma categoria de entidades que normalmente participam dele. Outro uso para o *skos:related* é a ligação entre duas categorias, em que uma é mais geral, e a outra é mais específica. O *skos:related* também

pode ser usado para representar *links* parte-inteiro que não são classificados como relacionamentos hierárquicos. A consulta SPARQL para a geração dos *links* semânticos *skos:related* é apresentada na Lista 2.

As regras utilizadas pela consulta SPARQL para gerar predicados SKOS seguem uma consistência formal, com um grau satisfatório de sucesso. Elas foram definidas com o uso de um método iterativo e a inclusão das seguintes etapas: (*i*) definição intuitiva de regras a partir da aplicação de uma amostra de 100 predicados *skos:prefLabel*; (*ii*) formalização das regras em formato algorítmico; (*iii*) teste da consulta SPARQL com CONSTRUCT em uma amostra maior; (*iv*) avaliação dos resultados; e, (*v*) refinamento das regras, se for o caso.

Foram necessárias cinco interações para obtermos a versão atual da consulta SPARQL. A consulta SPARQL com o comando CONSTRUCT gera automaticamente quatro propriedades de mapeamento no SKOS: *skos:exactMatch*, *skos:broader*, *skos:narrower* e *skos:related*.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

CONSTRUCT

{
   ?f1 <http://www.w3.org/2004/02/skos/core#related> ?f2 .
}

WHERE {

  ?p skos:narrower  ?f1 .
  ?p skos:narrower  ?f2 .
  ?p skos:prefLabel ?label1 .
  ?f1 skos:prefLabel ?label2 .
  ?f2 skos:prefLabel ?label3 .

  FILTER (!SAMETERM(?f1, ?f2) &&
        lang(?label1) = 'en' &&
        lang(?label2) = 'en' &&
        lang(?label3) = 'en')

}
```

Lista 2. Consulta SPARQL para geração de *links skos:related*

5. Resultados

Com a utilização da consulta SPARQL proposta, fizemos a coleta de 5373 termos do *Schlumberger Oilfield Glossary*, tendo como base para as comparações o predicado *skos:preflabel*. Os resultados obtidos e a proporção dos predicados SKOS gerados pela consulta estão descritos nas Tabelas 2 e 3.

Tabela 2. Número de predicados SKOS gerados pela consulta SPARQL

Predicado SKOS	exactMatch	related	broader	narrower	Total
Quantidade de *Links* gerados	2054	17487	1512	1512	22565
Amostra para avaliação	103	87	76	76	341
Validação pelo especialista	101	76	70	71	318
95% intervalo de confiança	$98,63^{\pm 4,37}\%$	$85,87^{\pm 1,13}\%$	$72,47^{\pm 3,53}\%$	$72,42^{\pm 3,58}\%$	82,35

Tabela 3. Exemplos de predicados SKOS gerados pela consulta SPARQL

Schlumberger Oilfield	PetroWiki - SPE's E&P	*Link* SKOS gerado	Avaliação
porosity	porosity	exactMatch	✔
	moldic porosity	narrower	✔
	diagenetic porosity	broader	✔
vugular porosity	vug	~~related~~	sameAs
	fracture porosity	related	✔
	wet clay porosity	broader	✔
wet clay porosity	porosity	broader	✔
	electrical double layer	~~related~~	petro:referencedTerm
	isolated porosity	related	✔
	clay	✘	broader
	smectite clay	✘	related

Os resultados foram avaliados com uma amostra de 341 exemplos de predicados SKOS gerados pela consulta SPARQL. Todos os *links* elaborados foram avaliados por um geólogo especialista no domínio da Extração e Produção de Petróleo (E&P) para comprovar a eficácia do modelo proposto. Vale ressaltar que a avaliação ocorreu com a utilização das ligações geradas entre os glossários *Schlumberger Oilfield* e *PetroWiki - SPE's E&P*.

O predicado *skos:exactMatch* obteve 98,63% de acerto, o que consideramos satisfatório, uma vez que utilizamos somente *stemming* e a remoção de *stopwords* como técnicas de pré-processamento textual para melhorar os resultados das comparações entre os termos [Avila and Soares 2012].

Já para os *links* gerados para o predicado *skos:related*, obtivemos 85,87% de acerto, contudo, de acordo com a Tabela 3, outros predicados deveriam ter sido gerados. O *link skos:related* somente será gerado após a confirmação da existência de ligações *skos:broader* e *skos:narrower* para o mesmo termo pai. Em alguns casos, ocorrerão exceções que precisam ser tratadas. No caso do termo **vugular porosity**, deveria ser gerado um *link owl:sameAs* ligando-o à **vug**. Faz-se necessário o refinamento da consulta SPARQL para tratar esses casos que não dependem da similaridade dos termos, mas sim do entendimento prévio do domínio em que estão sendo geradas as ligações semânticas.

Finalmente, para os predicados *skos:broader* e *skos:narrower*, obtivemos, respectivamente, 72,47% e 72,42%. O baixo desempenho desses resultados está intrinsecamente ligado à dependência que a consulta SPARQL tem com a comparação das *strings* que compõem cada termo. Dessa forma, os *links* somente serão gerados se ocorrer o *matching* entre os termos comparados.

Avaliando conjuntamente os predicados SKOS gerados, a consulta SPARQL ob-

teve 82,35% de ligações SKOS geradas corretamente. Levando-se em consideração que o predicado *skos:exactMatch* é a ligação semântica mais importante para o alinhamento terminológico, alcançando 98,63% de acerto nos experimentos, consideramos a consulta SPARQL eficaz para o mapeamento das correspondências entre a OD e as OFs. O desempenho da consulta SPARQL é menos convincente para as outras propriedades do SKOS, com resultados acima de 72%, o que consideramos eficaz.

Para trabalhos futuros, utilizaremos a proporção de cada tipo de ligação semântica gerada como base para o desenvolvimento de uma métrica de similaridade que apresente o nível de cobertura entre diferentes vocabulários controlados, dentro do domínio. Dessa forma, espera-se diminuir a dependência de um especialista na validação dos mapeamentos.

6. Trabalhos Relacionados

[Zapilko et al. 2012] detalha o uso de SKOS e o processo de adaptação de classes e propriedades para criar uma representação semanticamente completa. De acordo com os autores, essa nova modelagem permitiu a descoberta de ligações de equivalência e termos compostos com ambiguidades.

[van Ossenbruggen et al. 2011] apresenta as principais limitações das ferramentas atuais utilizadas para o alinhamento de instâncias/conceitos propondo uma abordagem alternativa em que utilizaram dois casos de uso para o mapeamento de ligações. Demonstra como a plataforma desenvolvida por eles obteve melhores resultados, em que, de acordo com os autores, os ganhos obtidos foram significativos. Entre as técnicas utilizadas destaca-se o uso de aprendizado de máquina.

Na pesquisa de [Ge and Chen 2010], foi desenvolvido um sistema baseado em ontologias para gerenciar dados de exploração de petróleo que abordam as questões de integração de dados e compartilhamento de informações. De acordo com os autores, a abordagem garante a validade dos dados de petróleo que irá apoiar o processo de descoberta de conhecimento de petróleo.

[Kazi and Kurian 2014] propõem uma metodologia de enriquecimento de ontologias, extraindo padrões de bases de conhecimento por meio de novas inferências derivadas do próprio domínio e utilizando algoritmos de aprendizado de máquina como redes neurais ou árvores de decisão. O conteúdo extraído passa por processos de mineração de dados e pela validação de um especialista. A ontologia construída auxilia na construção de um sistema especialista.

O trabalho de [d'Aquin et al. 2012] define um novo método de descoberta de conhecimento, combinando *(i)* as técnicas de mineração de dados para fazer emergir modelos implícitos de dados e *(ii)* o uso de padrões de engenharia de ontologias para capturar esses modelos de forma reutilizável. Os resultados atingidos apontam para a redução de tempo na preparação de dados e na interpretação de resultados, nas atividades de consulta de especialistas e na construção de ontologias e nas ligações semânticas.

[Miranda et al. 2016] apresentam estratégias alternativas para armazenar e acessar ontologias, com o objetivo de apoiar os processos de compartilhamento, o reuso, o planejamento, a avaliação, a personalização e a adaptação de conhecimentos em cenários relacionados à aprendizagem. Os resultados dos experimentos permitiram que os autores definissem uma estrutura capaz de suportar, do ponto de vista metodológico e tecnológico,

o uso de ontologias no contexto de um sistema educacional baseado na Web Semântica.

Os trabalhos aqui citados apresentaram diferentes maneiras de aplicar a ontologia para a descoberta de padrões e a correta categorização de termos e conceitos. Ainda não existe uma forma única para trabalhar com conceitos em contextos heterogêneos. Em cada caso, deve trabalhar-se no sentido de compreender o domínio que está sendo explorado e buscar os modelos mais apropriados para mapear/desenvolver sua respectiva definição, identificando os padrões que melhor se adaptam àquele contexto. Nosso trabalho apresentou resultados satisfatórios para a descoberta de ligações semânticas que podem auxiliar no enriquecimento de ontologias, favorecendo a descoberta de conhecimento.

7. Conclusão e Trabalhos Futuros

Demonstramos neste artigo uma consulta SPARQL com CONSTRUCT, conforme os resultados obtidos, eficaz para a geração semiautomática de predicados *skos:exactMatch*, sugerindo que a metodologia proposta pode ser utilizada para melhorar a qualidade dos mapeamentos terminológicos entre a OD e as OFs. A consulta SPARQL foi testada com a utilização de glossários de termos no domínio do petróleo, que pode, também em princípio, ser utilizada para outras terminologias e fontes de mapeamento.

O desempenho da metodologia proposta foi menos eficaz nos demais predicados *skos:broader*, *skos:narrower* e *skos:related*. Esses resultados ocorreram devido ao fato de as ligações serem geradas por meio de comparações de *strings*. Por exemplo, o termo **wet clay porosity** deve ter um *link skos:broader* com **clay**, porém a consulta SPARQL não mapeou essa ligação semântica. Outro fato importante é a impossibilidade de gerar *links* quando não houver alinhamento terminológico entre as *strings* comparadas. Dentre as possíveis soluções, podemos utilizar o sinônimo dos termos, aprendizado de máquina e/ou mapeamento das classes e subclasses do domínio.

De modo geral, a consulta SPARQL proposta depende principalmente da qualidade das terminologias comparadas. O uso de terminologias com princípios mais sistemáticos e relações hierárquicas mais transparentes pode facilitar os mapeamentos terminológicos e a geração dos *links* semânticos. As linguagens de representação de conhecimento formal, como a Linguagem de Ontologia da Web (OWL), podem ajudar nessa tarefa.

Os resultados apresentados foram para um único glossário, especificamente no domínio de Exploração e Produção de Petróleo (E&P). Daremos continuidade à pesquisa, aplicando o *framework* proposto para a integração de glossários em outros domínios. Entendemos que ainda necessitamos melhorar os resultados, principalmente em relação à geração dos *links* semânticos em casos de sinônimos, hiperônimos e hipônimos. Trataremos da resolução dessas entidades heterogêneas em trabalhos futuros.

Referências

Avila, R. and Soares, J. M. (2012). Concepção de ferramenta de apoio à correção de questões dissertativas com base na adaptação de algoritmos de comparação e busca textual combinados com técnicas de pré-processamento de textos. In *RENOTE Revista Novas Tecnologias na Educação*, volume 10.

Bizer, C., Cyganiak, R., and Gauß, T. (2007). The rdf book mashup: from web apis to a web of data. In *ESWC'07 Workshop on Scripting for the Semantic Web*, volume 1.

Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on*, pages 73–78, Acapulco, Mexico.

d'Aquin, M., Kronberger, G., and Suárez-Figueroa, M. C. (2012). Combining data mining and ontology engineering to enrich ontologies and linked data. In *KNOW@LOD*, volume 868 of *CEUR Workshop Proceedings*, pages 19–24. CEUR-WS.org.

Ge, J. and Chen, Z. (2010). Constructing ontology-based petroleum exploration database for knowledge discovery. *Trans Tech Publications, Switzerland*, 20-23:975–980.

Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition.

Kazi, A. and Kurian, D. (2014). An ontology based approach to data mining. In *International Journal of Engineering Development and Research (IJEDR)*, volume 2.

Lopes, G., Vidal, V., and Oliveira, M., editors (2016). *Construção de Linked Data Mashup para Integração de Dados da Saúde Pública*. SBC - XXXI Simpósio Brasileiro de Banco de Dados.

Miles, A., Matthews, B., Wilson, M., and Brickley, D. (2005). Skos core: Simple knowledge organisation for the web. In *Proceedings of the 2005 International Conference on Dublin Core and Metadata Applications: Vocabularies in Practice*, DCMI '05, pages 1:1–1:9.

Miranda, S., Orciuoli, F., and Sampson, D. G. (2016). A skos-based framework for subject ontologies to improve learning experiences. *Comput. Hum. Behav.*, 61(C):609–621.

Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL Query Language for RDF. W3C Recommendation.

Rahm, A. T. D. A. E., Thor, D., and Aumueller, E. (2007). Data integration support for mashups. In *Workshops at the Twenty-Second AAAI Conference on Artificial Intelligence*.

Tran, T. N., Truong, D. K., Hoang, H. H., and Le, T. M. (2014). *Linked Data Mashups: A Review on Technologies, Applications and Challenges*, pages 253–262. Springer International Publishing, Cham.

van Ossenbruggen, J., Hildebrand, M., and de Boer, V. (2011). Interactive vocabulary alignment. In Gradmann, S., Borri, F., Meghini, C., and Schuldt, H., editors, *TPDL*, volume 6966 of *Lecture Notes in Computer Science*, pages 296–307. Springer.

Volz, J., Bizer, C., Gaedke, M., and Kobilarov, G. (2009). Silk - a link discovery framework for the web of data. In Bizer, C., Heath, T., Berners-Lee, T., and Idehen, K., editors, *LDOW*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Zapilko, B., Schaible, J., Mayr, P., and Mathiak, B. (2012). Thesoz: A skos representation of the thesaurus for the social sciences. *CoRR*, abs/1209.5850.

paper:171359

ParallelNACluster: Uma estratégia paralela de clusterização para o casamento de múltiplos catálogos

Vinícius Pires de Moura Freire[1,3], **Fábio Porto**[2],
José A. F. de Macêdo [1]

[1]Departamento de Computação – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brasil

[2]Extreme Data Laboratory (DEXL Lab)
Laboratorio Nacional de Computacao Cientifica (LNCC)
Petrópolis – RJ – Brasil .

[3]Instituto Federal de Educação, Ciência e Tecnologia do Maranhão
Campus Coelho Neto – Maranhão – Brasil

vinicius.freire@ifma.edu.br, jose.macedo@lia.ufc.br, fabio@lncc.br

Abstract. *The astronomical catalogs cross-matching aims to identify common celestial objects present in different astronomical surveys. Traditional approaches in astronomy do not provide solutions to the problem of matching in the context of large data volume. In this paper, we have improved the NACluster algorithm by creating the ParallelNACluster strategy, a parallel version of NACluster that takes advantage of input data partitioning, and accepts large volumes of data even using a small hardware set. In addition, we propose the SCIBoundary, a new strategy for matching neighboring stars placed in different data partitions. The strategy leads to equivalent solutions in both NACluster and ParallelNACluster.*

Resumo. *O casamento de catálogos de astronomia tem o objetivo identificar objetos celestes em comum presentes em diferentes levantamentos astronômicos. As abordagens tradicionais em astronomia não apresentam soluções para o problema de casamento no contexto de grande volume de dados. Neste artigo, melhoramos o algoritmo NACluster ao apresentar a estratégia ParallelNACluster, uma versão paralela do NACluster que se aproveita do particionamento dos dados de entrada, e aceita grandes volumes de dados mesmo utilizando um conjunto de hardware de pequeno porte. Além disso, propomos o SCIBoundary, uma nova estratégia para tratamento do casamento de objetos espacialmente separados em partições de dados vizinhas. O SCIBoundary permite que obtenhamos resultados equivalentes entre o NACluster e o ParallelNACluster.*

1. Introdução

Levantamentos astronômicos usam instrumentos poderosos para varrer o céu e identificar objetos de interesse dentro da região pesquisada. Tais varreduras do céu dão origem ao mapeamento do céu, conhecido como catálogo. Os catálogos cobrem uma determinada faixa do céu e muitos deles possuem interseções em suas coberturas. Dessa forma, é interessante obter uma visão integrada deles, identificando os objetos em comum. Para isso, é necessário realizar o casamento de dados entre os catálogos, em outras

palavras, o *cross-matching* de catálogos. Em banco de dados, esse problema é conhecido como resolução de entidade. Em [Freire et al. 2014] apresentamos o NACluster, como solução para realizar o casamente de múltiplos catálogos. Essa solução, assim como as abordagens tradicionais funcionam de forma centralizada e são ineficientes para o casamento de grandes volumes de dados.

Usar grandes volumes de dados em algoritmos de casamento que funcionam de forma centralizada incorre em um tempo de execução extremamente alto ou incapacidade de executar por falta de memória. Em particular, para realizar o casamento entre bilhões de objetos através do NACluster de forma centralizada é necessário ter uma máquina com uma grande quantidade de memória RAM. Dessa forma, existe uma limitação física na maioria das máquinas usadas pela comunidade que impedem a realização de casamentos de grandes catálogos através do NACluster. Essas características nos motivam a criar uma versão distribuída do NACluster.

Este trabalho tem duas principais contribuições. A primeira contribuição é a proposta do algoritmo *ParallelNACluster*, uma solução distribuída para a resolução de entidades de vários catálogos espaciais através de técnicas de clusterização. O algoritmo tem o objetivo de reproduzir a clusterização do NACluster em um ambiente de larga escala com eficiência. De modo que, inicialmente, produza um particionamento dos dados de entrada tal que as fronteiras entre as partições sejam conhecidas e que haja um balanceamento das partições, ou seja, cada uma delas tenha um número semelhante de objetos e/ou variação de densidades semelhantes, para que o NACluster seja executado em cada partição com tempo relativamente próximo. O algoritmo paralelo mostrou-se eficiente e produziu clusters com qualidade semelhante ao centralizado, a diferença mínima apresentada foi em torno de 10^{-4} na precisão do resultado.

A segunda contribuição é a criação do *SCIBoundary*, uma estratégia para o tratamento das fronteiras geradas pelo particionamento e que objetiva encontrar os objetos influenciados por elas. Como o NACluster precisa da vizinhança para realizar o casamento, segundo seu algoritmo, e os dados particionados fazem com que a vizinhança da fronteira de duas partições vizinhas seja separada, é preciso conhecer as fronteiras e os objetos próximos em partições vizinhas para a que execução da clusterização seja feita corretamente sobre eles. Nos experimentos realizados constatamos a viabilidade do *SCIBoundary* e que a qualidade dos clusters produzidos por ele é semelhante aos do NACluster e ParallelNACluster, com uma diferença em torno de 10^{-2} para baixo no valor da precisão, justificada pelo tamanho da amostra ser bem menor que nos demais.

Este artigo está organizado da seguinte forma. Na seção 2 apresentamos o algoritmo centralizado NACluster, essencial para o entendimento da sua versão paralela proposta. A seção 3 apresenta as principais contribuições deste trabalho, o ParallelNACluster e o SCIBoundary. A seção 4 mostra os experimentos utilizados para validar as estratégias. Na seção 5 apresentamos os trabalhos relacionados. Finalmente, na seção 6 são feitas as considerações finais sobre este estudo sugerindo possíveis temas para trabalhos futuros.

2. Casamento de Múltiplos Catálogos

NACluster [Freire et al. 2014] é um acrônimo para **N-way CAtalog Clustering**, um algoritmo de clusterização para realizar casamentos entre múltiplos catálogos espaciais. Um conjunto S de *n* catálogos é fornecido como entrada e a saída é uma clusterização

composta por k clusters, onde todos os objetos associados ou mapeados a um cluster devam vir originalmente de catálogos diferentes e cada cluster deve representar um objeto espacial na região observada. Assim, não se sabe de antemão quantos clusters serão produzidos ao final da execução do algoritmo. Estas restrições nos abstém de adotar um algoritmo de clusterização tradicional, como *k-means*.

O NACluster é uma adaptação do *K-means*, onde os agrupamentos representam as entidades para as quais os objetos de catálogos devem ser mapeados. Assim, no NACluster, para cada objeto dos catálogos envolvidos, procura-se o cluster cuja distância entre seu centroide e o objeto seja menor que um valor dado, chamado de $epsilon$ (ϵ).

A ideia do algoritmo é comparar cada objeto com centroides de clusters distantes até ϵ, definido pelo usuário, usando a distância Euclidiana $d(o_i, C_a)$ de um objeto o_i para um centroide C_a. Quando existe um centroide C_a, tal que $d(o_i, C_a) < \epsilon$, então este centroide é classificado como potencial candidato a ser mapeado ao objeto o_i. Dentre os potenciais candidatos, aquele com menor distância ao objeto o_i é mapeado a ele. Esse mapeamento, no entanto, só pode ser aplicado quando não existe um outro objeto o_j no cluster de C_a que faça parte do mesmo catálogo de o_i.

No caso de um objeto o_j do mesmo catálogo de o_i já existir no cluster de C_a, é preciso tratar a colisão de mapeamento. Se $d(o_j, C_a) > d(o_i, C_a)$ então é necessário remover o objeto o_j do cluster de C_a, inserir o_i neste cluster, e procurar um outro cluster para o_j. Se $d(o_j, C_a) < d(o_i, C_a)$ então o algoritmo busca dentre os demais centroides candidatos a serem mapeados ao objeto o_i um novo cluster para realizar o mapeamento. Caso nenhum cluster seja encontrado na distância ϵ, um novo cluster com centroide C_b é criado para o ponto o_i, no qual sua posição é o centroide. O algoritmo termina quando todos os objetos de todos os catálogos forem avaliados.

A implementação do NACluster foi feita em Java está disponível à comunidade[1]. Para indexar os centroides usamos a *PH-tree*[Zaschke et al. 2014], reduzindo, assim, o espaço de busca. Os experimentos realizados são apresentados em [Freire et al. 2014].

3. Casamento Distribuído de Catálogos

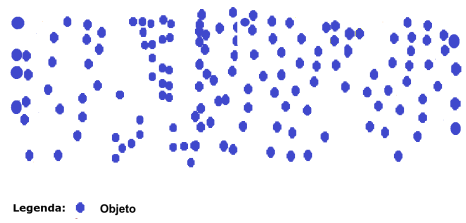
Conforme discutido anteriormente, este trabalho apresenta uma estratégia para a execução do casamento de catálogos astronômicos de forma distribuída e paralela. Neste sentido, considerando-se as técnicas atuais de processamento de grandes volumes de dados como o arcabouço Spark, existe a necessidade de particionar o conjunto de dados de entrada pelos nós de um cluster de computadores. Dessa forma, o conjunto de dados deve ser divididos em partes menores, chamadas partições. O particionamento distribui objetos pelos nós de processamento induzindo acesso distribuído quando referências a objetos em nós distintos são realizadas. A necessidade de particionamento vai de encontro com a noção de proximidade utilizada pelo método de clusterização. É importante que o aspecto físico de alocação de dados em partições não prejudique a noção lógica de proximidade entre os objetos, necessária ao correto processo de clusterização por vizinhança. De modo a tratar esse problema, introduz-se o conceito de fronteira que determina critérios lógicos de pertinência de objetos em partições. Considerando-se objetos espaciais identificados por sua posição no espaço 2D, as fronteiras formam um conjunto de posições em uma das dimensões que delimitam as partições dos dados.

[1]https://github.com/vinipires/NACluster

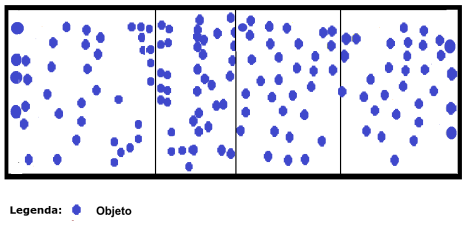
Nesta seção, estamos interessados em técnicas que permitam a paralelização do algoritmo NACluster. Suas características criam alguns desafios a serem superados na paralelização: 1 - os objetos vizinhos distantes em até ϵ devem ser executados juntos, portanto devem estar associados à mesma partição lógica; 2 - a alocação dos objetos em diferentes nós de forma desbalanceada pode fazer com que algumas máquinas executem o NACluster por mais tempo. Ou seja, enquanto as máquinas com mais objetos continuariam a clusterização, as demais terminariam o casamento e ficariam ociosas até o processo terminar por completo; 3 - o particionamento de um conjunto de dados gera fronteiras, f_i e f_{i+1}, entre partições, e essas fronteiras precisam ser conhecidas e tratadas.

3.1. Arquitetura Geral

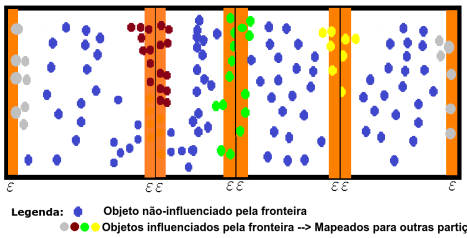
Neste artigo, propomos um modelo MapReduce para ser executado no framework de processamento distribuído Apache Spark[2]. Damos o nome de *ParallelNACluster* à solução proposta nos próximos parágrafos. Resumidamente, ela é dividida em quatro etapas principais ilustradas na Figura 1.



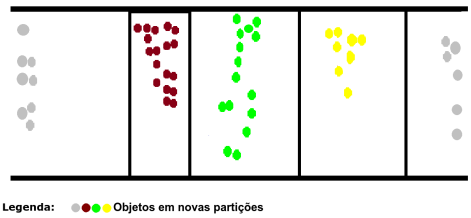
(a) Dados Originais



(b) Particionamento



(c) **Map** - NACluster aplicado em cada partição e os objetos influenciados pela fronteira são mapeados para uma nova partição e enviados ao Reduce



(d) Reduce - NACluster em cada nova partição

Figura 1. Etapas do Particionamento ao Reduce

Particionamento é fase em que os dados são particionados de acordo com os critérios apresentados anteriormente nos desafios da paralelização. O particionamento produz partições balanceadas e as fronteiras são conhecidas. A Figura 1(b) ilustra o particionamento a partir dos dados da Figura 1(a). **Map** é etapa em que executamos independentemente o NACluster em cada partição. Em seguida, identificamos os clusters que contêm objetos influenciados por cada fronteira, com o objetivo de mapear esses objetos para uma nova partição. A Figura 1(c) ilustra os não influenciados pela fronteira (os azuis) e influenciados pela fronteira (os demais). Mais detalhes serão apresentados na

[2]http://spark.apache.org/

seção 3.3. Os clusters não influenciados pela fronteira formados em cada partição, depois da execução do NACluster, são guardados para serem utilizados na última etapa (União). **Reduce** é o passo em que efetuamos o NACluster nas novas partições geradas, as quais contêm os objetos influenciados por cada fronteira. Esses objetos precisam ser executados juntos para obedecer os critérios de proximidade exigidos na clusterização do NACluster. A Figura 1(d) ilustra as novas partições e seus objetos. **União** é a fase em que unimos os clusters produzidos nas etapas Map e Reduce e geramos o resultado final.

3.2. Particionamento

Desejamos realizar um particionamento do *dataset* de maneira que os objetos vizinhos sejam associados à mesma partição lógica. Além disso, as partições precisam ser balanceadas, ou seja, cada partição deve ter um número semelhante de objetos.

A proposta do particionamento para este trabalho vem de [Gaspar and Porto 2014], onde é proposto o FRANCE (FRAgmeNtador de Catálogos Espaciais), um algoritmo iterativo para particionar dados em histogramas *equi-depth*. Formalizamos o particionamento através da Definição 1.

Definição 1. *(Particionamento) Dado um objeto* $o_{ji} < x, y, a_1, a_2, ..., a_t >$ *tal que* a_z, $1 \leq z \leq t$, *é um atributo de* o_{ji}, *e* $x \in X$ *e* $y \in Y$, *tal que* X *e* Y *são as dimensões espaciais; um particionamento* P *é uma lista de valores* $< v_1, v_2, ..., v_g >$ *tal que* $v_h \in x$ *e* $1 \leq h \leq g$. *Desta forma, uma partição* P_r, *tal que* $1 \leq r \leq g - 2$ *apresenta objetos vizinhos em uma região do espaço delimitada por* P_r *e* P_{r+1}.

O FRANCE calcula iterativamente os pontos de fragmentação segundo uma das dimensões espaciais, RA, se preocupando em manter as partições com uma quantidade equivalente de elementos (dado um δ), mesmo em catálogos com densidade não uniforme (como o Espaço).

As partições são geradas iterativamente até que a quantidade de elementos seja próxima em δ do balanceamento perfeito. Consideramos perfeito, balanceamentos em que as partições apresentam n/g elementos, onde n é o total de elementos e g a quantidade de partições.

O algoritmo tem como entrada o mesmo tipo de *dataset* usado no NACluster e gera como saída os valores da coordenada x onde ocorreram as divisões do espaço. A partir desses valores, podemos definir formalmente uma fronteira através da Definição 2.

Definição 2. *Uma fronteira* f_i *definida segundo um valor de particionamento* v_i *pertencente a* P, *contém um conjunto de objetos* O_{f_i} *cujos valores da coordenada* x *de particionamento estão entre* $v_i - \epsilon \leq x \leq v_i + \epsilon$.

Para maiores detalhes, a implementação do FRANCE que fizemos em java está disponível à comunidade na internet[3].

3.3. Tratamento de Fronteiras - Map (SCIBoundary), Reduce e União

Uma vez que as fronteiras tenham sido definidas, podem-se gerar as partições de dados a partir das associação de cada objeto do catálogo a um intervalo f_i e f_{i+1}. As partições assim criadas podem ser distribuídas pelos nós do cluster para processamento.

[3]http://github.com/vinipires/FRANCE

Assim, o NACluster pode ser executado em cada partição de forma isolada. A estratégia de processar isoladamente funciona para o NACluster, pois sua operação consiste em comparar cada objeto com possíveis centroides à uma distância ϵ. Portanto, é um processamento de vizinhança, e enquanto os vizinhos estiverem na mesma partição, o algoritmo tem um comportamento semelhante ao centralizado. A diferença aparece quando o vizinho está em outra partição.

O problema da fronteira pode ser enunciado da seguinte forma: dadas duas partições vizinhas P_i e P_j, uma fronteira f_k, $f_k \subset P_i$ e $f_k \subset P_j$, e dois objetos $o_i \in Pi$, e $o_j \in P_j$, se distância$(o_i, o_j) < \epsilon$ então o NACluster deve considerar f_k tal que $f_k \supset \{o_i, o_j\}$. Desta forma, para tratar o problema da fronteira, criamos a partir de P_i e P_j a fronteira f_k para processamento.

Como o particionamento do FRANCE é feito em apenas uma dimensão, existe somente uma fronteira entre duas partições vizinhas. Como a região onde pode aparecer ambiguidade na nossa definição de casamento está à uma distância ϵ do ponto referencial, e considerando que esse referencial é a fronteira, todos os elementos pertencentes à região distante ϵ dela precisam ser tratados nas duas partições.

Vamos expor a intuição de como podemos visualizar o tratamento das fronteiras. Essa visualização é mostrada através das componentes conexas de um grafo. Elas revelam existência de influência dos objetos da fronteira na formação dos clusters próximos à eles. As componentes conexas são quaisquer subgrafos conexos maximais de G, mais precisamente, são os maiores conjuntos de nós, tal que todos os nós conseguem alcançar os demais. Dado um grafo GC, formado a partir do resultado da execução do NACluster em cada partição, podemos observar todas as componentes conexas que têm origem na fronteira e detectar todos os clusters que tem influência da fronteira na sua formação. Caso processemos, juntos, todos os clusters que formam as componentes conexas que tem origem na fronteira de duas partições vizinhas em uma segunda etapa resolveremos o problema da fronteira.

Os nomes das partições que representam os objetos fora da fronteira são formados por Px, onde x corresponde ao número da partição. Os nomes das fronteiras são formados pelo nome da partição da esquerda unido com o nome daquela que fica à direita. Por exemplo, a fronteira que fica entre as partições $P1$ e $P2$ recebe o nome de $p1p2$.

O passo a passo do SCIBoundary é apresentado na Figura 2. Podemos examiná-la juntamente com os passos descritos acima. As elipses azuis representam os objetos, os retângulos vermelhos equivalem aos clusters (centroides) da fronteira e influenciados por ela, e os retângulos pretos correspondem aos clusters (centroides) fora da fronteira ou não influenciados por ela. Podemos observar que no momento "a" da Figura 2, possuímos os clusters (em preto) formados pelo NACluster, classificados inicialmente como fora da fronteira, e um objeto (em azul) pertencente à ela. Todos eles presentes na partição *P1*. No momento "b", o objeto procura por centroides de clusters em um raio ϵ e encontra dois deles. Estes dois clusters passam a ter a cor vermelha no instante "c", ou seja, passam a ser considerados como clusters influenciados pela fronteira e são mapeados para partição *p1p2*, e em "d" a componente conexa começa a se formar. No momento "e", os objetos do cluster encontrado são recuperados e inicia-se uma busca por outros clusters (centroides) fora da fronteira em "f". No instante "g", o novo cluster encontrado transforma-se em

cluster influenciado pela fronteira, tornando-se vermelho. De "h" até "n", os passos são semelhantes aos ocorridos anteriormente. Ao final da execução do SCIBoundary em "n", todos os clusters encontrados a partir da fronteira ficaram vermelhos e foram mapeados para a partição de mesmo nome da fronteira *p1p2*, mostrando que eles são influenciados por ela.

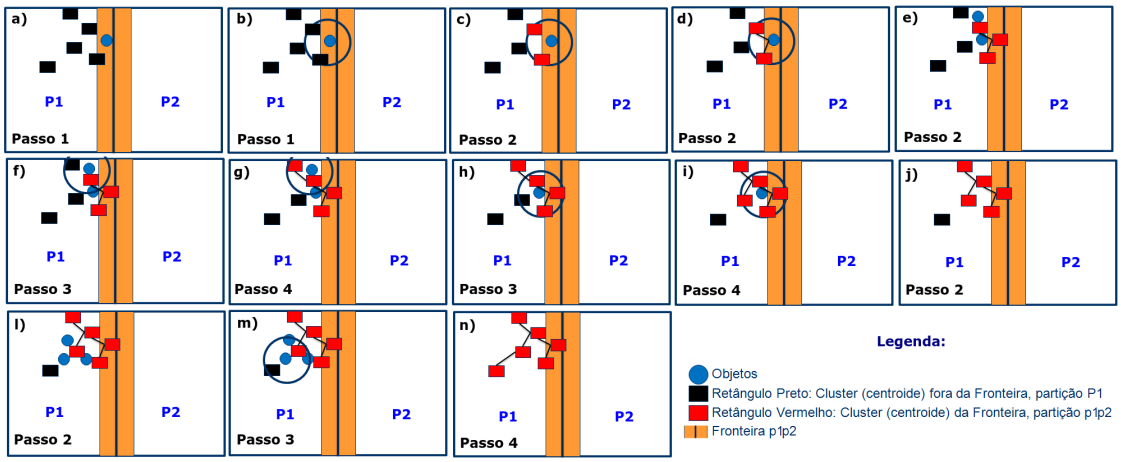


Figura 2. Passo a passo do SCIBoundary

No fim da execução do SCIBoundary em partições vizinhas teremos dois conjuntos de dados: o conjunto **A** contendo os clusters que não são afetados pela fronteira, e o conjunto **B** contendo os clusters afetados pela fronteira. Os clusters do conjunto **A** devem ser guardados para serem usados no resultado final e os clusters do conjunto **B** devem ser enviados para a etapa Reduce.

Um exemplo ilustrativo da classificação feita pelo SCIBoundary em todo o *dataset* é apresentado na Figura 3. Neste exemplo, o *dataset* possui 4 partições inicialmente. Ao final do SCIBoundary, o *dataset* passa a ter 8 partições, as 4 já existentes anteriormente ($P1$, $P2$,$P3$ e $P4$) e as 4 novas partições correspondentes às fronteiras ($p1p2$, $p2p3$, $p3p4$ e $p4p1$). A cor das elipses próximas às fronteiras representa a partição a qual os clusters pertencem.

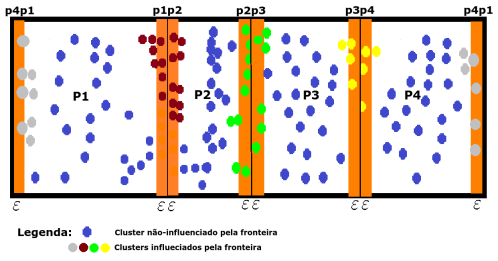


Figura 3. Exemplo ilustrativo dos clusters após a execução do SCIBoundary.

As novas partições criadas pelo SCIBoundary contendo os clusters influenciados pela fronteira são processadas na etapa Reduce. A Figura 4 ilustra as partições formadas. Cada elipse representa um cluster influenciado pela fronteira e sua cor representa sua partição composta por clusters que devem ser reexecutados, desta vez juntos na mesma

partição. O Reduce recebe como entrada o par (chave,valor), onde a chave é o *id* da nova partição e o valor é a lista dos clusters dessa partição. Para cada partição, são recuperados os objetos pertencentes à todos os seus clusters e é executado o NACluster sobre esses objetos, produzindo, assim, um novo casamento. Uma observação acerca desta etapa é que apesar dos objetos das fronteiras de *RA* 0.0 e 360.0 executarem juntos no Reduce, eles não estão sendo adaptados para corrigir seus clusters. A implementação foi feita dessa forma para facilitar o tratamento dessa fronteira em trabalhos futuros.

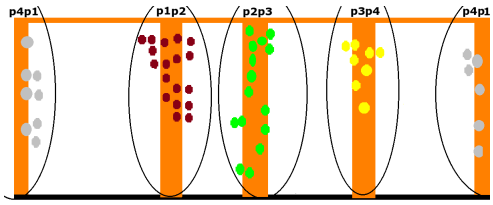


Figura 4. Partições formadas após a execução do SCIBoundary.

O resultado final do ParallelNACluster será a união dos clusters não influenciados pela fronteira, armazenados previamente na etapa *Map*, com os novos clusters produzidos pela etapa Reduce. A conclusão dessa etapa garante que o resultado do ParallelNACluster seja semelhante ao resultado do NACluster Centralizado aplicado ao mesmo *dataset*. A implementação do algoritmo paralelo está disponível à comunidade[4].

4. Experimentos

O conjunto de catálogos S dado como entrada no NACluster aparece implementado em um aquivo no formato texto. Cada linha contém uma tupla com a forma *(idObjeto, ra, dec, idCatalogo)*, cujos elementos são autoexplicativos. Quaisquer catálogos de dados espaciais que sejam de mesma natureza podem ser casados pelo NACluster. Utilizamos os catálogos de astronomia em todos os nossos experimentos.

Conforme visto na Seção 2, durante a execução do NACluster, para cada objeto dos catálogos envolvidos, procura-se o cluster cuja distância entre seu centroide e o objeto seja menor que um valor dado, chamado de ϵ. Além disso, cada cluster deve representar um objeto espacial na região observada. Para constatar que cada agrupamento formado pelo NACluster é um objeto espacial, precisa-se de um conjunto de catálogos S que possua um *Golden Standard* do seu casamento, ou seja, um conjunto de catálogos no qual saibamos previamente os casamentos corretos. No entanto, não encontramos nenhum conjunto de catálogos com essa característica, dificultando, assim, medirmos a qualidade do algoritmo. Portanto, geramos catálogos sintéticos e usamos os catálogos originais como *Golden Standard* com o propósito de avaliar a qualidade do casamento. Para criar um objeto celeste sintético, tomamos a posição do objeto real e adicionamos aleatoriamente um erro e, dentre os presentes na distribuição das distâncias de no máximo 1 arcseg entre os objetos de dois grandes catálogos, *2MASS* e *UCAC3*. Logo após, geramos aleatoriamente um ângulo θ (de 0 a 360 graus) ao qual esta nova posição estará deslocada em relação à original. Consideramos que objetos de catálogos diferentes distantes em raio de 1 arcseg (segundo de arco) são potenciais candidatosa ao casamento.

[4]https://github.com/vinipires/ParallelNACluster

#Catálogos	Tamanho	#Objetos	Precisão	Abrangência	Medida-F
3	31 GB	946 Milhões	0,99947	0,99947	0,99947
4	39 GB	1,1 Bilhão	0,99847	0,99849	0,99848
5	47 GB	1,4 Bilhão	0,99847	0,99849	0,99848
6	54 GB	1,6 Bilhão	0,99828	0,99831	0,99829

Tabela 1. *Métricas de qualidade do ParallelNACluster aplicado a diferentes conjuntos de catálogos baseados no 2MASS*

Partindo do pressuposto que o casamento destes dois catálogos esteja correto, assumimos que essas são distâncias entre o mesmo objeto presente nos dois catálogos e as usamos para construir o *Golden Standard*. Consideramos cada distância da distribuição como um erro na posição do mesmo objeto em diferentes catálogos. Assim, podemos gerar *n* catálogos sintéticos, a partir de um catálogo real e ter um *Golden Standard*, pois sabemos qual objeto real gerou um sintético, e um casamento entre eles deve ser considerado correto.

O hardware utilizado nos testes com o ParallelNACluster é um *cluster*[5] formado por 5 máquinas comuns presentes no *DEXL Lab* (laboratório pertencente ao Laboratório Nacional de Computação Científica): **Master**: Processador core i7-4790, com 4 núcleos e 8 *threads* de 3,60Ghz. Além de memória RAM de 16G; **slaves 1, 2 e 3**: Processador AMD Phenom(tm) II X2 B55 com 2 núcleos de 3,00 Ghz e 8GB de memória RAM; **slave 4**: Processador core i5-3470, com 4 núcleos de 3,20Ghz e memória RAM de 8G.

4.1. Qualidade na execução de grandes catálogos

A Tabela 1 apresenta as medidas de qualidade dos casamentos feitos através do ParallelNACluster quando aplicado a grandes conjuntos simulados de catálogos gerados a partir de todos os objetos do *2MASS* com tamanhos variados de 31 GB a 54 GB e particionados em 500 partições. Cada linha corresponde à uma execução em um conjunto de catálogos de tamanho diferente. A primeira coluna, *#Catálogos*, apresenta a quantidade de catálogos. A segunda coluna mostra o tamanho do arquivo e a terceira coluna, *#Objetos*, representa a quantidade aproximada de objetos. As próximas colunas, *Precisão*, *Abrangência* e *Medida-F*, cujos nomes são autoexplicativos, apresentam seus respectivos valores.

Observamos que as medidas de qualidade permanecem por volta de 0,99, mesmo aumentando a quantidade de catálogos presentes nos conjuntos. O ParalellNACluster produziu um resultado cuja a diferença média apresentada foi em torno de 10^{-4} de precisão se comparado ao algoritmo centralizado.

Também medimos a qualidade do casamento dos objetos influenciados pelas fronteiras, ou seja, aqueles identificados pelo *SCIBoundary* e casados na etapa *Reduce*. Como cada *dataset* possui 500 partições, identificamos 500 fronteiras. Nos conjuntos de catálogos testados, as fronteiras contêm aproximadamente $0,14\%$ do total de clusters. A Tabela 2 apresenta a qualidade do casamento ocorrido com os objetos da fronteira. A primeira coluna contém a quantidade de catálogos do *dataset*. A segunda mostra a quantidade total de clusters da fronteira. A terceira apresenta o número de agrupamentos da fronteira classificados de forma correta segundo o *Golden Standard*, enquanto a quarta coluna mostra a quantidade de clusters errados. Esses valores foram usados para calcular

[5]Cluster no sentido de ser formado por computadores ligados que trabalham em conjunto.

#Catálogos	#Clusters	#Clusters Corretos	#Clusters Errados	Precisão
3	646.361	613.845	32.516	0,9497
4	674.410	660.499	13.911	0,9794
5	692.301	659.987	32.314	0,9533
6	716.592	680.947	35.645	0,9503

Tabela 2. *Qualidade do casamento da fronteira na etapa Reduce dos mesmos conjuntos de catálogos da Tabela 1.*

a precisão, apresentada na coluna seguinte. O processo de clusterização realizado na etapa Reduce produziu um resultado cuja a diferença média apresentada foi em torno de 10^{-2} no valor da precisão se comparado ao NACluster e ParallelNAcluster, pois sua amostra é pequena, ou seja, a sua quantidade de *clusters* é muito menor que nos demais.

4.2. Escalabilidade: Volume x Tempo de execução

O objetivo dos próximos experimentos é avaliar a escalabilidade do ParallelNA-Cluster, ou seja, se ele está preparado para o crescimento do volume de dados. Usamos quatro diferentes conjuntos de catálogos com as seguintes quantidades aproximadas de objetos: 947 milhões, 1.18 Bilhão, 1.41 Bilhão e 1.65 Bilhão. E fixamos as seguintes configurações no Spark: 500 partições, –num-executors 4, –executor-cores 2, –excutor-memory 5G.

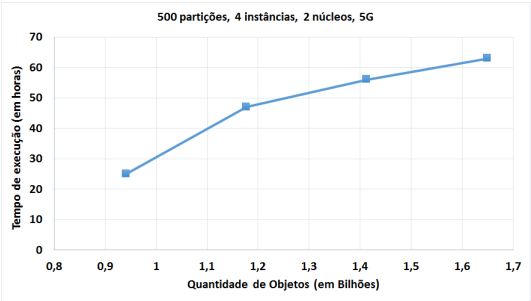


Figura 5. Volume x Tempo de uma Execução

Na Figura 5 apresentamos o tempo das execuções. No eixo x temos a quantidade de objetos e no eixo y o tempo de execução. Observamos que mesmo aumentando o tamanho do conjunto de catálogos para grandes volumes de dados, na ordem de bilhões de objetos, a aplicação executa por completo. É importante ressaltar que neste caso não temos a média do tempo, pois a aplicação demora de um a dois dias para executar grandes volumes de dados em nosso pequeno cluster, e problemas como energia e rede, impossibilitaram realizar dez rodadas para cada conjunto de catálogos.

Concluímos que mesmo com grandes volumes de dados é possível encontrar um equilíbrio entre o número de partições, o volume de dados e a memória disponível para cada instância. Logo, podemos crescer o tamanho do conjunto de catálogos contanto que ajustemos os parâmetros do Spark para um valor adequado ao tamanho dos dados. Em nossos experimentos encontramos uma escalabilidade linear até 1,7 bilhão de objetos em um cluster barato formado por apenas 4 máquinas.

5. Trabalhos Relacionados

Em [Dai and Lin 2012] foi proposta a estratégia DBSCAN-MR, uma abordagem que melhora a escalabilidade do DBSCAN dividindo os dados de entrada em partes menores e enviando-os aos nós para processamento paralelo. Em [Kwon et al. 2010] foi apresentado o algoritmo dFoF (*distributed Friends-of-Friends*, *amigos-de-amigos* distribuído, em português). Ele possui estratégia semelhante ao SCIBoundary e é a versão paralela do *FoF (Friends of Friends)*, um algoritmo de clusterização, semelhante ao DBSCAN, que aceita uma lista de objetos (*pid, x, y, z*) como entrada e retorna uma lista de tuplas de clusters (*pid, ClusterID*). Para calcular os agrupamentos, o algoritmo examina apenas a distância entre os objetos. *FoF* define dois objetos como amigos, se a distância entre eles é inferior a ϵ. No algoritmo $dFoF$ os dados são primeiramente particionados, depois agrupados localmente e, finalmente, os clusters locais são mesclados aos das partições vizinhas através do processamento da fronteira. Esses trabalhos relacionados propõem criar algoritmos de clusterização que funcionam de forma distribuída e dão ênfase ao particionamento dos dados de entrada, bem como do tratamento das fronteiras. No entanto, existem algumas diferenças quanto à natureza dos dados e, consequentemente, o objetivo final dos algoritmos apresentados nesta seção é diferente do objetivo da nossa proposta. Enquanto os algoritmos aqui apresentados estão interessados em unir os objetos próximos para identificar os grupos com formas arbitrárias, nós estamos interessados em formar agrupamentos de objetos próximos, mas que pertençam a diferentes catálogos.

Em [Zhao et al. 2009] apresentou-se o Parallel k-means, um algoritmo de agrupamento que classifica um conjunto de elementos de entrada em k clusters de forma paralela através de uma abordagem Map Reduce do k-means. A estratégia de casamento do ParallelNACluster é semelhante ao mapeamento de objetos a grupamentos no Parallel k-means em alguns aspectos como, por exemplo, no cálculo do centroide do cluster, no qual a posição de cada centroide é a média da posição dos objetos do cluster e ambos utilizam a abordagem de alocar o objeto ao cluster de centroide mais próximo. No entanto, apesar de existirem características semelhantes entre as duas abordagens, existem várias diferenças. Dentre elas o relaxamento do k, pois diferentemente do Parallel k-means, o número de clusters formados, k, não é dado como entrada no ParalllelNACluster e a quantidade final de clusters só é descoberta ao final da execução. Além disso, no Parallel K-Means os centroides estão dispostos globalmente, enquanto que no ParallelNAClusters, os centroides são controlados localmente. Outra diferença é que no ParallelNAcluster os centroides iniciais são pré definidos e o resultado é determinístico na maioria dos casos, enquanto que no Parallel k-means, os centroides iniciais são aleatórios e a sequência na formação dos clusters é geralmente diferente a cada execução.

Ainda sobre as diferenças entre os dois algoritmos, no ParallelNACluster, o número de objetos mapeados para um agrupamento não pode ultrapassar a quantidade de catálogos existentes. Cada cluster pode conter apenas um objeto de cada catálogo. No Parallel k-means não existe restrição quanto ao tamanho do cluster e o tipo de catálogo que ele pode alocar. Adicionalmente, no ParallelNACluster é possível que um objeto seja removido do cluster ao qual ele tenha sido adicionado anteriormente e seja alocado em outro cluster caso nele ocorra uma colisão de objetos do mesmo catálogo, ou seja, um objeto tentar ser adicionado em um cluster onde já existe outro objeto do mesmo catálogo. Neste caso, aquele que for mais próximo do centroide permanece no cluster. No Parallel k-means não existe tratamento de colisão e objetos de diferentes catálogos podem

fazer parte do mesmo cluster. Com tantas diferenças, podemos concluir que o Parallel k-means não iria satisfazer os pré-requisitos de casamento do ParallelNACluster. A falta das restrições no Parallel k-means, bem como o k fixo, fariam com que um cluster não representasse um objeto real.

6. Conclusão

O ParallelNACluster é uma solução distribuída que pode ser utilizada por toda a comunidade através do uso de máquinas baratas. O particionamento permite que sejam utilizados *datasets* de tamanho na ordem de bilhões de objetos, tornando nossa proposta uma solução atual e robusta, que funciona mesmo com o grande aumento do volume de dados, bastando apenas aumentar o número de partições, utilizando o mesmo *cluster*, à medida que aumenta o tamanho dos dados. Dentre os trabalhos futuros propomos realizar experimentos com o ParallelNACluster em hardwares de alto desempenho, adicionar na função de distância outros atributos com o intuito de melhorar a qualidade do casamento nas regiões ambíguas, criar uma estratégia para calcular as probabilidades de casamento e desenvolver um método para tratar casamentos de catálogos com escalas diferentes e permitir múltiplos *matchings*;

7. Agradecimentos

Este trabalho recebeu o apoio financeiro da CAPES (bolsa de doutorado). Fabio Porto foi parcialmente financiado pela Bolsa CNPq de produtividade processo: 310109/2015-9.

Referências

Dai, B.-R. and Lin, I.-C. (2012). Efficient Map/Reduce-Based DBSCAN Algorithm with Optimized Data Partition. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, pages 59–66, Washington, DC, USA. IEEE Computer Society.

Freire, V. P., Porto, F., Akbarinia, R., and de Macêdo, J. A. F. (2014). NACluster: A Non-supervised Clustering Algorithm for Matching Multi Catalogues. In *2014 IEEE 10th International Conference on e-Science*, pages 83–86. IEEE.

Gaspar, D. and Porto, F. (2014). A Multi-Dimensional Equi-Depth Partitioning Strategy for Astronomy Catalog Data.

Kwon, Y., Nunley, D., Gardner, J. P., Balazinska, M., Howe, B., and Loebman, S. (2010). Scalable clustering algorithm for N-body simulations in a shared-nothing cluster. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6187 LNCS:132–150.

Zaschke, T., Zimmerli, C., and Norrie, M. C. (2014). The PH-tree: A Space-efficient Storage Structure and Multi-dimensional Index. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 397–408, New York, NY, USA. ACM.

Zhao, W., Ma, H., and He, Q. (2009). Parallel k-means clustering based on mapreduce. In *IEEE International Conference on Cloud Computing*, pages 674–679. Springer.

paper:171414

Particionamento como Ação de Sintonia Fina em Bancos de Dados Relacionais

Ana Carolina Almeida[1], Antony Seabra de Medeiros[2], Rogério Luís de Carvalho Costa[2], Sérgio Lifschitz[2]

[1]Depto de Informática e Ciência da Computação UERJ

[2]Departamento de Informática PUC-Rio

ana.almeida@ime.uerj.br, {amedeiros, rogcosta, sergio}@inf.puc-rio.br

Abstract. *Table partitioning is usually applied for distributing data. It may also be considered for performance issues, since it allows full scans involving only those partitions that satisfy the queries predicates. This work presents heuristics that select a partitioning strategy for each table and estimate their workload benefits. We also present some studies on the use of our proposed heuristics with the TPC-H benchmark. We show that partitioning can be effectively used as a first-order database tuning technique.*

Resumo. *O particionamento é uma técnica utilizada para distribuição de dados que também possui potencial para ajuste de desempenho, pois permite que a varredura das tabelas seja realizada somente nas partições que satisfazem os predicados das consultas. Este trabalho apresenta heurísticas que permitem selecionar uma estratégia de particionamento para cada tabela e estimar seus benefícios para a carga de trabalho. Aplica-se as heurísticas com o benchmark TPC-H e mostra-se que o particionamento pode ser utilizado como uma ação de sintonia fina de primeira ordem.*

1 – Introdução

A sintonia fina de bancos de dados consiste na realização de alterações visando obter melhor desempenho para a execução de comandos de bancos de dados. As ações de sintonia fina podem reduzir de forma significativa o tempo de execução dos comandos, fazendo, por exemplo, com que comandos e scripts que levariam vários minutos para serem concluídos passem a ser executados em apenas alguns poucos segundos.

Dentre as estratégias de sintonia fina mais comumente utilizadas por administradores de bancos de dados, estão a construção de estruturas de acesso, tais como índices, índices parciais e visões materializadas, a desnormalização e a reescrita de consultas. Por outro lado, o particionamento de tabelas consiste na transformação de uma tabela em dois ou mais fragmentos, onde cada fragmento é, na verdade, um subconjunto dos dados da tabela (linhas ou colunas) gerado com base em critério pré-definido.

Neste artigo, estudamos o uso do particionamento como ação de sintonia fina. A seleção dos critérios de fragmentação é uma das maiores dificuldades no uso do particionamento. Propomos heurísticas que permitem selecionar estratégias de

particionamento com base nos benefícios gerados em termos da redução do tempo de execução de uma carga de trabalho. Realizamos, também, diversos estudos que permitem validar a aplicação das heurísticas propostas e observar o uso do particionamento como ação de sintonia fina.

Na próxima seção, relembramos alguns conceitos fundamentais relacionados ao particionamento de tabelas e mencionamos alguns trabalhos relacionados relevantes. Na Seção 3 apresentamos as heurísticas propostas. A Seção 4 apresenta as avaliações experimentais realizadas e os resultados obtidos. Finalmente, a Seção 5 traz algumas conclusões e comentários finais.

2 – Fundamentos e Trabalhos Relacionados

O particionamento (ou fragmentação) de uma tabela pode ser horizontal, vertical ou híbrido. No particionamento horizontal, divide-se a tabela em subconjuntos disjuntos de linhas ou tuplas (Valduriez, 2011), sendo cada subconjunto (ou partição) definido com base em um critério, composto por um ou mais atributos e os respectivos valores para cada partição. A fragmentação horizontal será denominada *primária* se a chave do particionamento for um atributo da tabela que está sendo particionada ou *derivada* no caso contrário. No particionamento vertical, divide-se a tabela em subconjuntos de colunas, repetindo-se sua chave primária em cada partição. Por fim, no particionamento híbrido, combina-se o particionamento horizontal e o vertical.

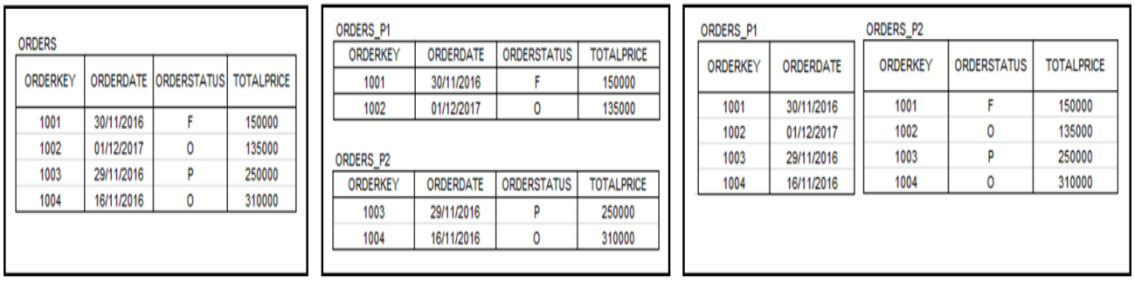


Figura 1: Particionamento horizontal e vertical

Na Figura 1, a tabela *orders* possui 4 tuplas. No particionamento horizontal, por exemplo, usando *totalprice* como chave do particionamento, pode-se ter duas partições: a partição *orders_p1* contendo as 2 tuplas com *totalprice<200000* e a partição *orders_p2* contendo as 2 tuplas com *totalprice≥200000*. Já no particionamento vertical, por exemplo, a partição *orders_p1* contém as colunas *orderkey* e *orderdate* das 4 tuplas, enquanto a partição *orders_p2* contém as colunas *orderkey*, *orderstatus* e *totalprice* das 4 tuplas.

Ambos os tipos de particionamento possuem potencial para sintonia fina na medida em que podem reduzir a quantidade de operações de entrada e saída (I/O) a serem realizadas durante a execução de comandos SQL. Por exemplo, para a consulta *Select * From orders Where totalprice>=250000*, somente a partição *orders_p2* seria acessada, ao invés de toda a tabela *orders*, diminuindo o número de páginas a serem percorridas e, assim, reduzindo o custo de execução da consulta.

A redução do custo de acesso a dados é uma preocupação constante em grandes bases de dados. Para tal, utilizam-se técnicas de *sintonia fina*. Em (Monteiro, 2008), a sintonia fina automática é apresentada, incluindo a seleção automática de índices, visões materializadas, índices sobre visões e particionamento. O trabalho enfatiza que a manutenção automática do projeto físico deve conter a coleta da carga de trabalho, a seleção das estruturas apropriadas e a manutenção dessas estruturas. É apresentada a heurística HISAI e uma ferramenta para apoiar o uso da manutenção automática de índices, que pode ser utilizada com vários SGBDs padrão de mercado. A heurística apresenta um modelo para estimar o benefício de um índice como ação de sintonia fina.

Em (Almeida et. al, 2015) é apresentado DBX, ferramenta para a manutenção automática do projeto físico em bancos de dados relacionais. DBX é, de fato, um framework, dado que pode ser instanciado para atividades distintas de sintonia fina envolvendo visões materializadas e índices, e, potencialmente, outras ações. DBX é uma ferramenta não-intrusiva, independente de SGBD, que pode ser executada de forma contínua e sem intervenção de um DBA. As ações específicas de sintonia fina automática são determinadas em função da análise dos planos de execução capturados e possíveis modificações em planos hipotéticos, com implementação da heurística HISAI (Monteiro, 2008).

Existem ainda trabalhos na literatura que exibem o potencial de benefício do particionamento como ação de sintonia fina, como (Agrawal, 2004) e (Bellatreche, 2004). Outros, mais focados em selecionar estratégias de particionamento, como (Curino, 2010), utilizam grafos para representar uma carga de trabalho e algoritmos de particionamento de grafos para selecionar uma estratégia de particionamento para as tabelas envolvidas. Há também alguns que, como (Wang, 2013), utilizam a análise das consultas e atributos das tabelas para determinar uma estratégia de particionamento.

3 – Uso do particionamento para sintonia fina

Neste trabalho, propomos a heurística HISAP – *Heurística Integrada para Seleção e Acompanhamento de Partições* – cujo objetivo é analisar todas as consultas e atualizações da carga de trabalho e selecionar uma estratégia de particionamento para o esquema do banco de dados, com base no benefício obtido nos custos dos planos de execução correspondentes às consultas e atualizações.

3.1 – Seleção da estratégia de particionamento

A estratégia proposta é composta de seis passos, descritos a seguir:

Passo 1 – Inicialmente, a heurística constrói uma lista de predicados simples utilizados nas consultas e, para cada predicado, mantém o custo da operação, que vai sendo acumulado durante a execução da heurística, um contador para registrar a frequência de acesso, incrementado a cada consulta que utiliza este predicado, e outro contador para a frequência de atualização, incrementado a cada atualização que utiliza este predicado.

Note-se que os predicados são decompostos independentemente do tipo de operador utilizado no plano de execução. Na heurística HISAI importa analisar os predicados de operações *Sequential Scan*. São eles os candidatos a chaves de índices. Em HISAP, caso mantida a mesma restrição, a heurística estaria propondo o particionamento como

estratégia secundária, para os casos onde os índices não seriam considerados benéficos a ponto de deixarem de ser hipotéticos. Todas as operações são analisadas, incluindo *Index Scan* e *Index Seek*, porque é possível que predicados para os quais o otimizador selecione um índice sejam chaves de particionamento candidatas e de fato ofereçam um benefício maior com tal ação de sintonia fina.

Passo 2 – Com base nesta lista de predicados, uma lista de chaves candidatas é construída, acumulando-se o custo, a frequência de acesso e a frequência de atualização. Para cada chave candidata, o valor *(access frequency – update frequency)* é calculado e, para todos os valores negativos, a entrada correspondente é removida da lista.

Passo 3 – Em seguida, a lista é classificada em ordem decrescente de custo, o que garante tratar primeiramente os predicados com maior custo para a varredura das tabelas. A lista é percorrida e, para cada chave candidata, as subfases da seleção da estratégia de particionamento são executadas, onde são definidos a chave e o o tipo de particionamento, o número de partições e o valor da chave para cada partição.

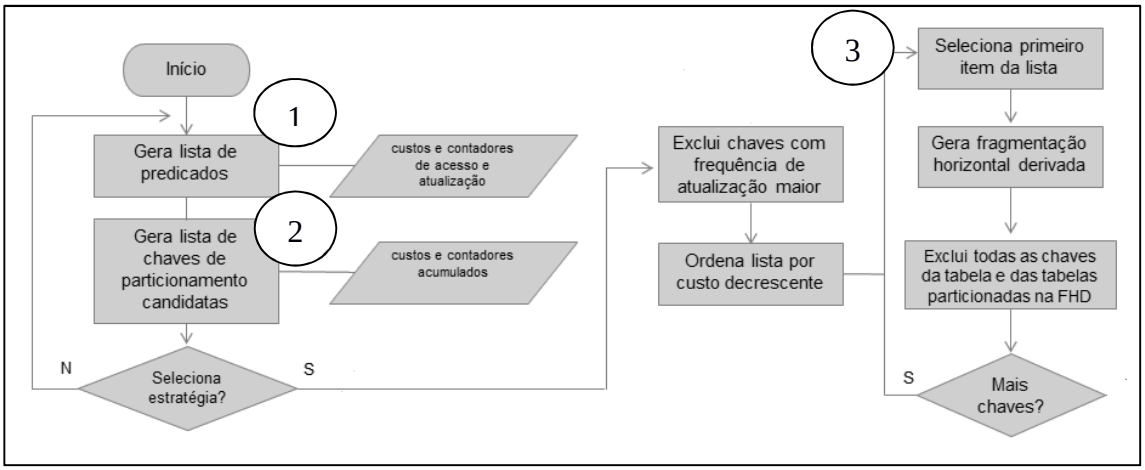


Figura 2: Fluxograma da heurística HISAP

Quando não há mais chaves de particionamento a serem definidas, termina-se o ciclo e inicia-se a definição dos tipos e dos valores da chave em cada partição de cada tabela.

Passo 4 – O tipo de particionamento é definido com base no tipo da coluna e no tipo de acesso mais frequente sobre a coluna. Tipos numéricos e data cujos predicados usam faixas de valores convergem para o tipo *range*. Tipos compostos por caracteres cujos predicados usam valores discretos convergem para o tipo *list*. E tipos numéricos usados em chaves de tabelas convergem para o tipo *hash*.

Passo 5 – As partições da tabela e os valores da chave para cada partição são gerados a partir do histograma da coluna. A heurística aceita dois parâmetros de entrada: o número máximo de partições e o tipo de balanceamento das partições – baseado em volume ou carga de trabalho. Os balanceamentos não são garantidos pela heurística, mas sim pela sua implementação. Por exemplo, um balanceamento de carga pode ser alcançado com o uso do algoritmo BEA (*Bond Energy Algorithm*), que faz uso de uma matriz de afinidades para gerar novos grupos com afinidades maiores. Em (Valduriez, 2011), o algoritmo é utilizado para o particionamento vertical. Uma matriz de afinidades

é obtida substituindo-se as *colunas* referenciadas nas consultas pelas *partições* acessadas nas consultas.

Passo 6 – Ao gerar o particionamento para uma tabela, são gerados os particionamentos das tabelas que compõem a fragmentação horizontal derivada. Assim, as chaves candidatas de todas as tabelas consideradas nesta operação são removidas da lista. Em seguida, inicia-se um novo ciclo que analisa novos subconjuntos de tabelas do esquema do banco de dados. O processo se repete até o término da lista (para particionar todas as tabelas do banco de dados) ou até que seja alcançado um limiar (baseado em um fator de corte) que pode ser definido pelo DBA.

3.2 – Estimativa de benefício do particionamento

Seja *W* uma carga de trabalho submetida a um banco de dados e $B(Q_i)$ o benefício de um particionamento para a consulta $Qi \in W$. Para uma dada consulta *Qi* da carga de trabalho, seu plano de execução utiliza operadores de varredura em estruturas de acesso, a saber, uma tabela ou um índice. Quando um índice não é utilizado, tem-se um operador do tipo *Sequential Scan* no plano de execução. Nota-se que quando uma tabela é particionada, a varredura ocorre somente na(s) partição(ões) da tabela que satisfaz(em) o predicado. Logo, o benefício $B(Q_i)$ de um particionamento para uma consulta assim formada pode ser calculado pela diferença entre o custo do *Sequential Scan* na tabela inteira e o custo do *Sequential Scan* na tabela particionada (Medeiros, 2017).

Por outro lado, quando um índice é utilizado, tem-se um operador do tipo *Index Scan* (ou *Index Seek*) no plano de execução. O benefício $B(Q_i)$ de um particionamento para uma consulta assim formada poderia ser calculado pela diferença entre o custo do *Index Scan* (ou *Index Seek*) no índice não-particionado e o custo do *Index Scan* (ou *Index Seek*) no índice particionado. Entretanto, como as seletividades dos predicados nesta tabela poderão sofrer uma diminuição, considera-se que o índice não será utilizado e, portanto, o custo é calculado pela diferença entre o custo do *Index Scan* (ou *Index Seek*) no índice não-particionado e o custo do *Sequential Scan* na tabela particionada.

Seja $C(SS)$ o custo do *Sequential Scan* em uma tabela, $C(IS)$ o custo do *Index Scan* (ou *Index Seek*) em um índice e $C(SSP_k)$ o custo do *Sequential Scan* em uma tabela hipoteticamente particionada através da chave de particionamento P_k.

Considerando $n=1..t$ o número de varreduras contidas no plano de execução de Q_i, o benefício $B(Q_i)$ é o somatório dos benefícios alcançados para cada varredura:

```
SE operador = Sequential Scan
    B(Qi) = B(Qi) + ∑ n=1..t ( C(SSn) – C(SSPkn) )
SENÃO
    B(Qi) = B(Qi) + ∑ n=1..t ( C(ISn) - C(SSPkn) )
FIM-SE
```

Considerando a criação de partições balanceadas, isto é, com uma distribuição uniforme do volume de dados em cada uma delas, efetuar uma varredura em uma estrutura em uma única partição significa percorrer o número total de linhas da tabela dividido pelo número total de partições.

Seja NP_k o número de partições gerado pelo particionamento de uma tabela usando a chave P_k e $NP(SSP_k)$ o número de partições acessadas em uma busca na estrutura particionada. Logo, o custo do *Sequential Scan* em uma estrutura particionada, *C(SSPk)*, pode ser assim estimado:

$$C(SSP_k) = (C(SS) / NP_k) * NP(SSP_k)$$

Da mesma forma, o custo do *Index Scan* num índice, *C(ISPk)*, poderia ser estimado como:

$$C(ISP_k) = (C(IS) / NP_k) * NP(ISP_k)$$

Ao analisar uma carga de trabalho, o benefício acumulado de um particionamento hipotético *AB(W)* será, portanto, o somatório dos benefícios alcançados em cada consulta da carga de trabalho multiplicados pela frequência de cada consulta.

$$AB(W) = \sum(B(Q_i) \times F(Q_i))$$

Duas considerações fundamentais são necessárias na implementação do particionamento hipotético. Primeiro, a fragmentação horizontal derivada gera um particionamento em cadeia, que pode melhorar o desempenho das consultas que usam junção nessas tabelas e o benefício gerado pelo particionamento. Quando um particionamento hipotético é criado para uma determinada tabela, um particionamento hipotético para todas as tabelas que fazem parte da cadeia deve ser criado também. Como exemplo, no banco de dados do TPC-H, o particionamento da tabela *orders* leva ao particionamento da tabela *lineitem*. Isto pode garantir que numa consulta com junção entre as duas tabelas, não somente as partições das tabelas que satisfazem os predicados sejam acessadas, mas também que somente as partições-filhas das partições-mães sejam acessadas. Segundo, o particionamento não somente altera os volumes de dados a serem percorridos pelos operadores de um plano de execução, mas pode influenciar as decisões do otimizador na medida em que tais operadores são escolhidos com base nesses volumes de dados e nas estatísticas correspondentes. Ao alterar estratégias de junção, por exemplo, o benefício do particionamento deixa de ser tão somente a diferença no custo da varredura (Medeiros, 2017).

Com bases nestas duas considerações, o benefício do particionamento para uma consulta e, portanto, para a carga de trabalho, pode sofrer variações. Estas variações dependem fundamentalmente da implementação do particionamento no SGBD e pode ser necessário fazer ajustes nas estimativas.

4 – Avaliações experimentais

A seguir apresentam-se os resultados encontrados nas simulações de execução da heurística HISAP. Utilizou-se o *benchmark* de referência TPC-H (TPC, 2017). As bases de dados, com volumes de 1GB, 10GB e 50GB, foram instaladas sob o SGBD *PostgreSQL 9.6.1* sob a configuração padrão em um servidor Intel Core i3 com 4 GB de memória RAM e HD de 500 GB, rodando sistema operacional Windows 7.

Foram criados três bancos de dados, *tpch*, *tpch-i* e *tpch-p*, para coletar os resultados das consultas com, respectivamente, nenhuma estratégia de sintonia fina, índices como ação de sintonia fina e particionamento como ação de sintonia fina. Foram utilizados 3 cenários de teste com características distintas para avaliação das heurísticas de seleção da estratégia de particionamento. Em todos os cenários são utilizadas as consultas fornecidas pelo próprio *benchmark*, submetidas aos SGBDs através de *scripts Powershell*. Considerando as características de um *data warehouse* típico com dados de 5 anos, optamos por definir o número máximo de partições em *cinco* e o tipo de balanceamento *por volume*. A métrica usada para avaliação dos resultados é o tempo de execução da consulta. A contagem do tempo (em milissegundos) é realizada pelo próprio SGBD e o tempo de execução de cada consulta apresentada a seguir é a média de três execuções, a partir da segunda execução, ou seja, a primeira execução é descartada.

4.1 – Cenário 1 – Particionamento, índices completos e índices parciais

Este cenário de teste é composto pela execução de variações da consulta 1 (Figura 3) e objetiva estudar o comportamento das estratégias de sintonia fina quando variamos a seletividade dos predicados em uma consulta sem joins, CPU-bound.

```
SELECT L_RETURNFLAG, L_LINESTATUS, SUM(L_QUANTITY) AS SUM_QTY,
SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS SUM_DISC_PRICE,
SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1+L_TAX)) AS SUM_CHARGE,
AVG(L_QUANTITY) AS AVG_QTY, AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
AVG(L_DISCOUNT) AS AVG_DISC, COUNT(*) AS COUNT_ORDER
FROM LINEITEM WHERE L_SHIPDATE <= '1993-01-01'
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS
```

Figura 3: Consulta 1

O maior custo nos planos de execução associados às variações desta consulta está no *Sequential Scan* da tabela *lineitem*. Uma ação de sintonia fina (sugerida por HISAI) é a criação de um índice *btree* nas colunas (*l_shipdate*, *l_returnflag*, *l_linestatus*), justamente as colunas que aparecem nas cláusulas *Where* e *Group By*. O otimizador seleciona este índice quando a seletividade é suficientemente alta (*tpch-i*). Na execução da heurística HISAP, as chaves de particionamento candidatas são *l_shipdate*, *l_returnflag* e *l_linestatus*, sendo que *l_shipdate* possui o maior custo associado. Como a coluna é do tipo *date* e as consultas utilizam predicados com faixas de valores, o tipo do particionamento é *range*. Através do histograma da coluna, são geradas cinco partições utilizando o balanceamento por volume.

Quando a consulta é satisfeita em *tpch-p* com o acesso de uma única partição de *orders*, temos os resultados mostrados no gráfico da Figura 4. Além disso, quando temos um índice parcial que contemple o mesmo predicado satisfeito com o acesso a uma única partição, temos um desempenho (*tpch-pi*) semelhante à estratégia de particionamento.

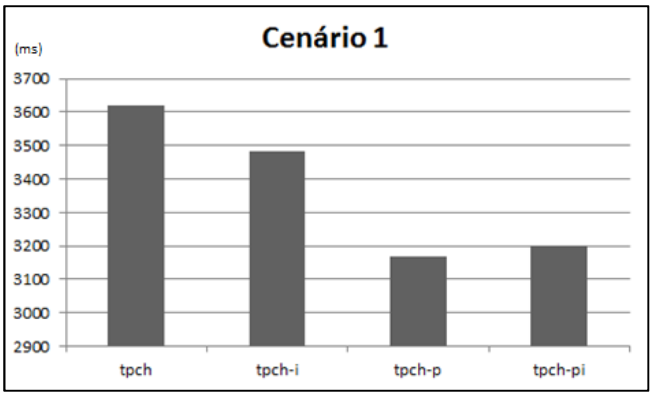


Figura 4: Cenário 1 – resultados

Quanto mais abrangente for a faixa de valores contemplada pelo predicado da consulta, maior o número de partições a serem acessadas e, consequentemente, maiores o custo da consulta e o tempo necessário para executá-la. Por exemplo, com o filtro *Where l_shipdate ≤ '1996-01-01'*, o *Sequential Scan* é executado sobre as partições que contém as linhas correspondentes, neste caso 3 partições. Por outro lado, ao aumentar-se a faixa de valores do predicado, sua seletividade é diminuída, o que acarreta diretamente na diminuição da probabilidade do otimizador selecionar um índice para seu plano de execução. O plano de execução desta mesma consulta no banco de dados *tpch-i* não seleciona o índice anteriormente utilizado nas colunas (*l_shipdate, l_returnflag, l_linestatus*). Ao contrário, observa-se um *Sequential Scan* na tabela inteira. Portanto, o crescimento do número de partições acessadas por uma consulta está associado à diminuição da seletividade do predicado e, por consequência direta, a diminuição da probabilidade do uso de índices.

Forçamos a varredura em n=1, 2, 3, 4 e 5 partições da tabela *lineitem* e estudamos o comportamento das estratégias de sintonia fina. Em n=1 o custo em C_P (configuração de particionamento em HISAP) é bem menor. Para n=2, com o aumento da faixa de valores que resulta no acesso a duas partições da tabela, o custo em C_I é quase igual ao aferido em C, sendo o custo em C_P ainda bem menor. Para n=3 e n=4, o índice já não é mais selecionado e C_P se mantém menor que C, resultado do acesso a 3 e 4 partições, respectivamente. Para n=5, todas as partições são acessadas e os custos são similares, conforme mostrado na Figura 5.

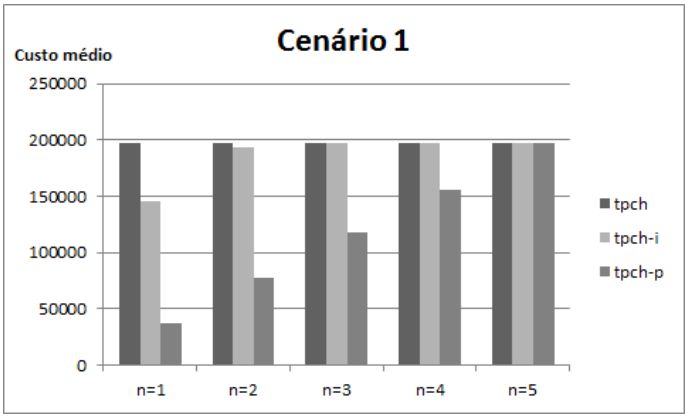


Figura 5: Cenário 1 – comportamento das estratégias de sintonia fina

4.2 – Cenário 2 – Benefício do particionamento para a carga de trabalho

O cenário 2 é composto pela execução de variações das consultas 3 e 8 e objetiva mostrar o benefício do particionamento em consultas I/O-bound. A consulta 3 (Figura 6) utiliza dois predicados nas colunas *o_orderdate* e *l_shipdate*. Ainda, incrementa a frequência de acesso para *l_orderkey*, *o_orderdate* e *o_shippriority* na cláusula *Group By* e incrementa mais uma vez a frequência de *o_orderdate* na cláusula *Order By*.

```
SELECT L_ORDERKEY, SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS REVENUE,
O_ORDERDATE, O_SHIPPRIORITY
FROM CUSTOMER, ORDERS, LINEITEM
WHERE C_MKTSEGMENT = 'BUILDING' AND C_CUSTKEY = O_CUSTKEY
AND L_ORDERKEY = O_ORDERKEY AND O_ORDERDATE < '1994-03-15'
AND L_SHIPDATE > '1999-03-15'
GROUP BY L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY
ORDER BY REVENUE DESC, O_ORDERDATE LIMIT 10
```

Figura 6: Consulta 3

Seu plano de execução mostra, inicialmente, um *Sequential Scan* em *customer* e um *Sequential Scan* em *orders*, e uma junção usando *(orders.o_custkey = customer.c_custkey)*. O maior custo está associado ao *Sequential Scan* da tabela *orders*. Em *tpch-i*, o otimizador seleciona os índices *i_orders_orderdate* e *i_lineitem_shipdate*; assim, um *Index Scan* é realizado em ambos os índices, fazendo com que o custo deste plano seja menor que o custo do plano gerado em *tpch*.

A consulta 8 (Figura 7) em *tpch* é resolvida com varreduras sequenciais e a aplicação dos filtros apropriados durante as varreduras. As linhas de *orders* e *lineitem* são unidas através de um *Nested Loop Join* e o maior custo no plano de execução está relacionado ao *Sequential Scan* de *orders*. Em *tpch-i* o índice *i_orders_orderdate* é selecionado para a varredura de *orders*, enquanto para *lineitem* a varredura sequencial é preferida. O maior custo entre os predicados em ambas as consultas está associado à varredura da tabela *orders*, com filtro na coluna *orderdate*. Em ambos os planos esta junção é implementada por um operador *Nested Loop Join*. A execução da heurística seleciona, portanto, o particionamento horizontal na coluna *o_orderdate* da tabela *orders*. As faixas de valores são construídas a partir do histograma da coluna para cinco partições.

```
SELECT O_YEAR, SUM(CASE WHEN NATION = 'BRAZIL' THEN VOLUME ELSE 0
END)/SUM(VOLUME) AS MKT_SHARE
FROM (SELECT O_ORDERDATE AS O_YEAR, L_EXTENDEDPRICE*(1-L_DISCOUNT) AS
VOLUME, N2.N_NAME AS NATION FROM PART, SUPPLIER, LINEITEM, ORDERS, CUSTOMER,
NATION N1, NATION N2, REGION
WHERE P_PARTKEY = L_PARTKEY AND S_SUPPKEY = L_SUPPKEY
AND L_ORDERKEY = O_ORDERKEY AND O_CUSTKEY = C_CUSTKEY
AND C_NATIONKEY = N1.N_NATIONKEY
AND N1.N_REGIONKEY = R_REGIONKEY AND R_NAME = 'AMERICA'
AND S_NATIONKEY = N2.N_NATIONKEY
AND O_ORDERDATE BETWEEN '1995-01-01' AND '1996-12-31'
AND P_TYPE= 'ECONOMY ANODIZED STEEL') AS ALL_NATIONS
GROUP BY O_YEAR
ORDER BY O_YEAR
```

Figura 7: Consulta 8

A fragmentação horizontal derivada ocorre a partir da tabela particionada *orders*. Sua tabela-filha, *lineitem*, cuja chave primária é composta por (*l_orderkey*, *l_linenumber*), será particionada por referência a *orders*. Em *tpch-p* a execução da consulta 3 apresenta varreduras completas somente nas partições da tabela *orders* que satisfazem o predicado e nas partições correspondentes de *lineitem*. O plano de execução mostra *Nested Loop Join* para juntar as linhas das duas tabelas particionadas baseado no critério *orders.o_orderkey = lineitem.l_orderkey*. Nota-se que o otimizador percorreu somente as partições da tabela *lineitem* que correspondem às partições da tabela *orders*. A execução da consulta 8 em *tpch-p* também apresenta o mesmo comportamento (Figura 8). Para *orders*, somente as partições que satisfazem o predicado em *orderdate* são acessadas. Na figura 8 observa-se o benefício da estratégia.

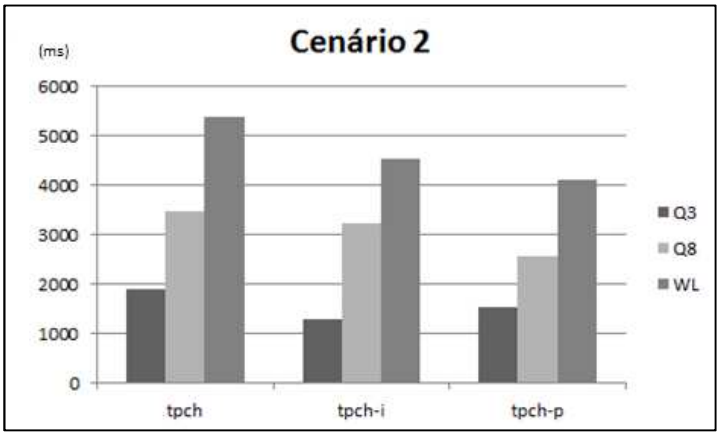


Figura 8: Cenário 2 – resultados

Qual o aumento do benefício do particionamento quando o volume de dados aumenta? Realizamos os mesmos testes em bases de dados com tamanhos 10GB e 50GB (Figura 9). Para Q8 o ganho de benefício relativo da estratégia de particionamento em relação à estratégia de índice foi de 21,67% no crescimento a 10GB e de 24,85% no crescimento a 50GB. Para a carga de trabalho (WL) o ganho de benefício relativo da estratégia de particionamento em relação à estratégia de índice foi de 16,58% no crescimento a 10GB e de 19,61% no crescimento a 50GB.

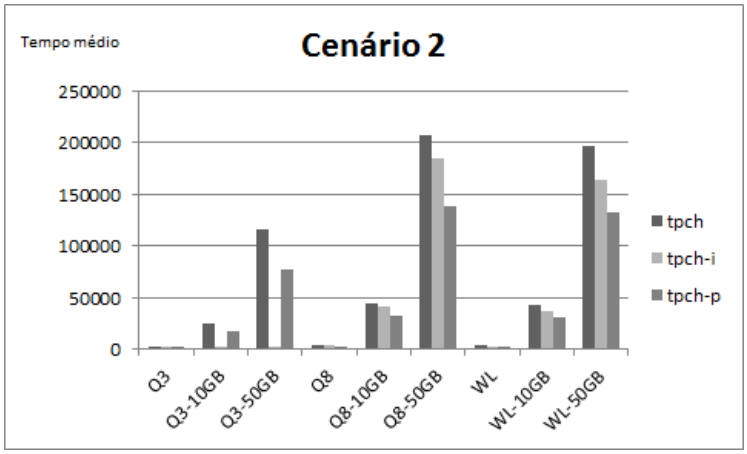


Figura 9: Cenário 2 - Aumento do volume de dados

4.3 – Cenário 3 - Atualizações

Seja a seguinte atualização executada no cenário 2, no qual a tabela *orders* foi particionada por *o_orderdate* e a tabela *lineitem* foi particionada por referência a *orders*.

```
UPDATE orders SET o_comment='Lorem ipsum dolor sit amet, '
WHERE o_orderdate BETWEEN '1994-06-01' AND '1997-10-01'
```

Esta atualização é executada em *tpch* através de um *Sequential Scan* em 29923,080 milisegundos. Em *tpch-p* esta mesma atualização é executada em 32237,294 milisegundos com *Sequential Scans* nas partições 2 a 5 de *orders*. Seja agora a atualização abaixo executada em outro cenário, no qual a tabela *lineitem* tenha sido particionada por *l_shipmode* (Cenário 3 – Figura 10).

```
UPDATE lineitem SET l_comment='Lorem ipsum dolor sit amet, '
WHERE l_shipmode IN ('MAIL', 'SHIP')
```

Esta atualização é executada em *tpch* através de um *Sequential Scan* em 150492,298 milisegundos. Em *tpch-p* esta mesma atualização é executada em 63459,862 milisegundos com *Sequential Scan* somente na partição correspondente de *lineitem*.

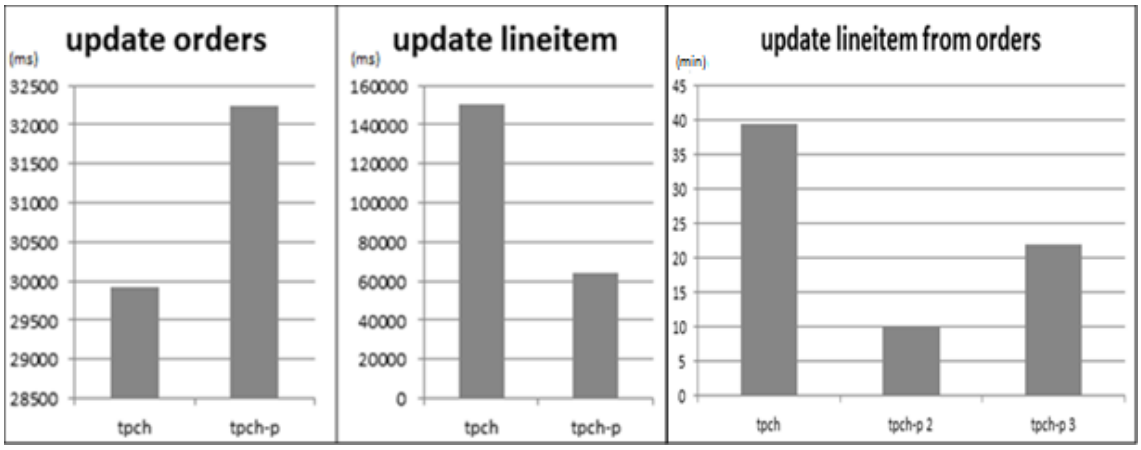


Figura 10: Cenário 3 – resultados

Os tempos do *update* no cenário 2 são semelhantes, porém os tempos medidos para a atualização no cenário 3 indicam que a atualização teve um desempenho melhor com a configuração de particionamento. Seja agora a atualização abaixo.

```
UPDATE lineitem AS l
SET l_comment='Lorem ipsum dolor sit amet, '
FROM orders AS o WHERE l.l_orderkey = o.o_orderkey
AND o_orderdate BETWEEN '1994-06-01' and '1997-10-01'
```

Esta atualização é executada em *tpch* em 39,33 minutos. Em *tpch-p* no cenário 2 esta mesma atualização é executada em 10,13 minutos e em *tpch-p* no cenário 3 esta mesma atualização é executada em 21,92 minutos.

5 – Conclusões e Trabalhos Futuros

Diante dos cenários estudados e dos resultados coletados, conclui-se que o particionamento horizontal é uma estratégia principal de sintonia fina, cujo ganho de desempenho pode ser comparado aos de índices, índices parciais e visões materializadas. Além disso, atualizações de dados também podem se beneficiar do particionamento.

Como trabalhos futuros estamos implementando as heurísticas na ferramenta de sintonia fina DBX (Almeida et. al, 2015), desenvolvida pelo grupo de pesquisa BioBD da PUC-Rio. Além disso, pretende-se (i) integrar o particionamento com as outras estratégias, como índices e visões materializadas, de forma que os benefícios das estratégias sejam estimados em conjunto, ao invés de o serem de forma independente, e (ii) estudar a evolução da heurística para bases de dados já particionadas, com a avaliação de possíveis cenários com alterações nos critérios de particionamento, incluindo a alteração na chave ou intervalo de particionamento utilizados, a criação de novos níveis de particionamento ou, simplesmente, a eliminação de partições.

Referências

Agrawal, S.; Narasayya, V. & Yang, B. (2004). Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design. Anais da ACM International Conference on Management of Data (SIGMOD), pp. 359–370.

Almeida, A.C.B.; Brayner, A., Monteiro, J.M.; Lifschitz, S.; & Oliveira, R.P. (2015). DBX: um framework para auto-sintonia fina baseado em planos hipotéticos. Anais do SBBD Sessão de Demos, pp.149-154.

Bellatreche L., Schneider M., Lorinquer H. & Mohania M. (2004). Bringing Together Partitioning, Materialized Views and Indexes to Optimize Performance of Relational Data Warehouses, Anais da International Conference on Data Warehousing and Knowledge Discovery. DaWaK, pp. 15-25.

Costa, R. L. C.; Lifschitz, S.; Noronha, M. & Salles, M.V. (2005). Implementation of an agent architecture for automated index tuning. Anais da International Conference on Data Engineering (ICDE) Workshop on Self-managing Database Systems, pp. 1215.

Curino, C.; Jones, E.; Zhang, Y. & Madden, S. (2010). Schism: a workload-driven approach to database replication and partitioning. J. of PVLDB 3, I1-2, pp. 48–57.

Medeiros, A. S. (2017). Particionamento com Ação de Sintonia Fina. Dissertação de Mestrado, Departamento de Informática, PUC-Rio.

Monteiro, J. M. (2008). Uma abordagem não intrusiva para a manutenção automática do projeto físico de bancos de dados. Tese de Doutorado, Departamento de Informática, PUC-Rio.

TPC (2017). TPC-H Benchmark, http://www.tpc.org/tpch [Acessado em 24/05/2017].

Valduriez, P. & Özsu, M.T. (2011). Principles of Distributed Database Systems. 3rd. edition. Springer.

Wang, X.; Chen, J.; & Du, X. (2013). ASAWA: An Automatic Partition Key Selection Strategy. APWeb LNCS 7808, pp. 609–620.

paper:171357

Relational graph data management on the edge: Grouping vertices' neighborhood with Edge-*k*

Lucas C. Scabora[1], Paulo H. Oliveira[1], Daniel S. Kaster[2], Agma J. M. Traina[1], Caetano Traina-Jr[1]

[1]Institute of Mathematics and Computer Sciences – University of São Paulo (USP)
São Carlos – SP – Brazil

[2]Department of Computer Science – University of Londrina (UEL)
Londrina – PR – Brazil

{lucascsb,pholiveira}@usp.br,dskaster@uel.br,{agma,caetano}@icmc.usp.br

Abstract. *As the amount of data represented as graph grows, several frameworks are employing relational databases to manage them. However, the existing solutions store graphs creating a row for each edge in an edge table. In this paper, we propose Edge-k, a novel storage approach that combines additional columns in the edges table, allowing to tune the number of edges stored in a single row by taking into account the overall neighborhood of the vertices, thus providing a better table organization. Compared to the existing approaches in the literature, experiments reveal that our proposal was able to reach a speedup of 66% over a representative real dataset and up to 57% in synthetic datasets when processing Single Source Shortest Path queries. Hence, our solution advances the state of the art in the context of graph data management within relational databases systems.*

1. Introduction

Complex networks are present everywhere, from communication infrastructures to social networks and urban street organization [Barabási and Pósfai 2016]. These networks are usually represented as graphs, in which the nodes are mapped as vertices and their relationships as edges. The amount of applications using graph structures to represent data has increased significantly. Many of them focus on analyzing the characteristics of the graphs, applying algorithms such as finding connected components, page rank and Single Source Shortest Paths (SSSP) [Silva et al. 2016].

For example, consider a graph composed of researchers and their respective publications. The relationship between two authors is defined by their common publications, *i.e.* one of them is the other's coauthor. Suppose the weight w of the edge corresponding to such relationship can be determined by $1 \div n_p$, where n_p is the number of shared publications between any pair of authors. In this case, $0 \leq w \leq 1$, and weights close to 0 indicate stronger relationships, whereas weights close to 1 indicate weaker relationships. In this scenario, consider a query searching for a specific researcher's collaborative network, which aims at identifying not only its direct coauthors, but also the coauthors of their coauthors and so on. To run such a query, which would allow the analysis of the researcher's influence on its study field, the SSSP algorithm can be employed. Given those characteristics of the weights, a weight close to 0 could also be seen as a short

distance between authors, due to the large amount of shared publications. Conversely, a weight close to 1 could be seen as a long distance between them, due to the small amount of shared publications (or none at all). Hence, the SSSP algorithm can determine the strongest connections among the authors starting at a specified author, since it is achieved by identifying the minimal paths from that author to the other ones.

As the amount of applications grows, the volume of data also increases to the point that it does not fit in main memory, compelling them to focus on I/O operations in disks. Relational Database Management Systems (RDBMS) provide an infrastructure to support graph data management with some useful features such as storage, data buffering, index and optimizations. The main problem is that the Structured Query Language (SQL) is not adequate to express graph related queries. To solve this problem, a number of frameworks have been proposed. The FEM (Frontier, Expand and Merge) framework has been proposed to translate analytical graph queries into SQL commands [Gao et al. 2011, Gao et al. 2014]. This framework implements an SSSP query through tracking frontier vertices and iteratively expanding them, merging the new elements with the set of visited ones. The Grail framework [Fan et al. 2015], as well as its adaptation in the RDBMS Vertica [Jindal et al. 2015], follows a similar approach, detailing query execution plans and allowing query optimizations. However, both frameworks are limited to a single graph representation, not exploring the query execution performance in alternative representations. Both frameworks employ one relation for the set of edges and another for the set of vertices. The edges are organized as a list, each of them stored as a tuple in the table. The vertex degree (*i.e.* number of neighbors) is the number of edges incident to it, thus, in this representation, it determines the number of rows required to store the vertex.

In this paper we propose Edge-*k*, a novel and flexible strategy to store the neighborhood of a vertex in a RDBMS, grouping a predefined number of edges in the same entry. Particularly, since an entry is handled by the RDBMS as a table's row, we aim at reducing the overall number of rows required to store the graph while keeping the amount of *null* values in the edge table as small as possible, avoiding to generate wide tables and improving the performance of queries. Reducing the number of rows decreases the amount of data blocks used by the RDBMS as less space is dedicated to row headers, therefore minimizing the quantity of I/O operations. Furthermore, by grouping neighbor vertices in the same row, Edge-*k* ensures that at least part of them will be contiguously stored in disk, which can also contribute to achieve higher processing performance.

Regarding the aforementioned frameworks, we adapted the SSSP query to Edge-*k*, which was used to evaluate both real and synthetic datasets. This paper focuses on the SSSP algorithm because it is widely employed in many applications, such as discovering indirect relationships in a social network and finding minimal paths to interesting places in a city. The average query execution time revealed a strong correlation with the number of data blocks required to store the graph data. Thus, decreasing the amount of blocks, which is related to reducing both rows and *null* values, also speeds up queries. In the synthetic datasets, we achieved gains in query time from 46% to 57%. On the other hand, the real dataset allowed a gain of up to 66% regarding the first 2 iterations of the SSSP algorithm, whereas for 4 iterations the gain achieved 49%.

The remainder of this paper is organized as follows. Section 2 describes the related work, regarding existing approaches for graph data management using RDBMS. Section 3

details the proposed Edge-*k* storage approach. We provide an experimental analysis and discussion of our proposal in Section 4. Finally, Section 5 presents the conclusions.

2. Related Work

Existing storage schemas for graph data management employing RDBMS include: triple store schema; binary tables; and n-ary tables [Levandoski and Mokbel 2009]. The triple store schema organizes the data into a triple <*subject, property, object*>, in which *subject* refers to an entity instance (*e.g.* the identifier of a vertex or edge) and *property* represents an attribute of the entity, whose value is denoted by *object*. Considering an entity-relationship model, each attribute and relationship of an entity is described by *property* [Neumann and Weikum 2010]. Grouping each distinct *property* into a specific table corresponds to the binary representation, in which the table has only two columns: entity identifier and property value. Including other properties as new columns in the same table leads to the n-ary representation, and a special case in which all properties are assigned to the same table results in a property table [Levandoski and Mokbel 2009].

The disadvantage of placing every property into a distinct column is that it increases the data sparsity due to the heterogeneity in a entity set. In order to deal with this issue, DB2RDF [Bornea et al. 2013] proposed to aggregate sets of properties in the same column. DB2RDF defines vertex rows with a maximum of k properties, each represented by a pair of property and value columns. If there are more than k properties, they are split in several rows. This approach reduces the space required to store the dataset because it requires a smaller number of physical columns to store properties. SQLGraph [Sun et al. 2015] extended DB2RDF employing the same approach in non-relational databases. However, both works focused only on associating vertices with their properties, not dealing with relationships among vertices.

An alternative way to represent adjacency lists employs a column of array type [Chen 2013]. In this approach, each row in the table stores all the neighborhood of a vertex in a single column. Although this representation disrespects the first normal form, the authors state that it is well suited for sparse graphs since it avoids *null* values. However, such proposal focuses on scenarios in which all vertices have a small degree. Furthermore, the authors did not explore the scenario in which the row size is bigger than the data block size. The same idea of splitting rows adopted in DB2RDF and SQLGraph can be a solution here, but the insertion of new neighbors of a vertex needs yet to be analyzed.

Our work draws inspiration also from the edge sharding approach introduced by the GraphChi framework [Kyrola et al. 2012], a disk-based system for computing graph algorithms efficiently. GraphChi splits its data into intervals, namely shards, in which the storage of edges is sorted by their source. The framework requires that each shard fits entirely in memory. However, GraphChi does not follow an RDBMS-based approach, which our proposal employs. Moreover, regarding a database system, these shards can be analogously seen as the data blocks in which the tables are stored.

Therefore, a similar approach can be used to improve RDBMS-based graph processing by grouping the neighborhood of a vertex in the edge table according to distinct values of k, which is a feature that neither DB2RDF nor SQLGraph provide for an RDBMS. This approach is the basis of Edge-*k*, which is presented in the next section.

3. The Edge-*k* Table Organization

A graph G is denoted as $G = \{V, E\}$, where V is a set of $|V|$ vertices and E is a set of $|E|$ edges, where both vertices and edges can have properties. Our work deals with the representation of edges. In the context of the Relational Model, on which our proposal is based on, edges and vertices are mapped into two distinct relations. Each vertex in V has a unique identifier attribute (*id*) and as many attributes as needed for the properties. An edge in E is defined as an ordered pair $\langle src, des \rangle$, where src is the vertex *source id* and des is the *destination* vertex *id*. Additional information for the edges can be expressed as extra attributes in the relation, allowing to further detail the relationships among the vertices. For example, an edge is defined as $\langle src, des, weight \rangle$ when storing the measure $weight$ of the relationship between src and des vertices.

Our novel technique Edge-*k* implements the vertex and edge relations as regular tables, and groups multiple *destinations* of a vertex in a same row in the edge table, up to a defined maximum amount of k *destinations* per row. Thus, a tuple in the edge table can store up to k edges starting from the same *source* vertex. If the edges have a property assigned, an additional column is included for each distinct *destination* vertex. Considering that every edge has a weight, the edge table definition becomes $E_k = \langle src, des_1, w_1, ..., des_k, w_k \rangle$, where w_i is the $weight$ between src and des_i, $1 \leq i \leq k$. This simple, but effective organization reduces the table overhead with tuple headers, as well as stores contiguously adjustable pieces of the vertex neighborhood, enabling to reduce storage space, number of disk accesses and query response time.

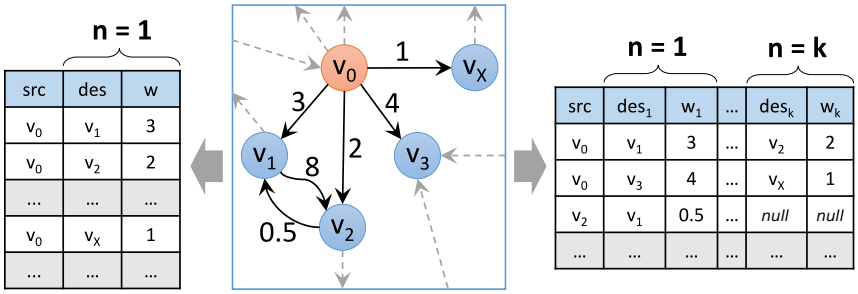


Figure 1. Comparison between the conventional and proposed approaches.

Figure 1 illustrates the proposed technique for a directed graph with a *source* vertex v_0 and a neighborhood of size n and each edge has a weigh w as an assigned property. The table at the left shows the conventional approach, which stores one *destination* per row. On the right is the proposed approach, which allows to store up to k *destinations* in the same row. For values of n smaller than k, all the neighborhood fits in the same row and the remaining values are filled with *null* values. If n and k are equal, which is the ideal case, a single completely occupied row contains the whole vertex neighborhood. For values of n larger than k, the neighborhood is stored in groups of size k. The last group may have less than k neighbors and thus the row is completed with *null* values. In the figure, the number of neighbors around vertex v_0 is divisible by k, fitting in two complete rows. On the other hand, the *source* vertex v_2 requires *null* values to complete its row.

By adopting an edge table that groups k *destination* vertices in a single row, we have a reduction of the number of rows at the cost of allowing *null* values. Therefore, the efficiency of our proposal is related to the parameter k, which is a function of the number

of rows and the exceeding space of the edge table. Minimizing both quantities reduces the number of tuples required to the edge table be stored and, consequently, improves query performance. However, our proposal introduces an additional cost when performing individual update operations. This occurs because inserting an edge requires searching an empty destination vertex on the Edge-k table before inserting it. As such operations are less frequent than the search operations, its impact on the overall query load is small.

3.1. Estimation of the number of rows and *null* values

This section presents measures useful for the Edge-*k* method. They are calculated based on the value k chosen and on the edge attributes. In Equation 1, for each vertex v, the amount of rows is defined as the smallest integer value that is greater than its total neighborhood size ($degree_v$) divided by the defined value of k. Then, by summing the number of rows for each $v \in V$, we obtained the total amount (η_{rows}) of edge table's rows.

$$\eta_{rows} = \sum_{v \in V} \left\lceil \frac{degree_v}{k} \right\rceil \tag{1}$$

In Equation 2, $degree_v \bmod k$ obtains the number of remaining neighbors that do not fill a complete row in the edge table. By subtracting that number from k, and employing the modulo operator once again, we determine number of *null* values for each vertex. Finally, by summing them, the total amount of *null* values (η_{nulls}) is obtained.

$$\eta_{nulls} = \sum_{v \in V} (k - (degree_v \bmod k)) \bmod k \tag{2}$$

The optimal edge storage approach of a graph in an RDBMS table would use a single row to store the neighborhood of each vertex. In this case, both the row overhead imposed by the row header (which is necessary to store internal control data) and the space wasted due to *null* values are minimal. However, such organization would be properly represented only for the particular case in which every vertex has a fixed number of neighbors. This premise is not valid for most complex networks. As aforementioned, there are works that suggest employing RDBMS extensions, such as arrays. However, such approaches do not follow the Relational Model anymore, thus loosing robustness. Our work, on the other hand, strictly follows the Relational Model, employing a configurable control of the wasted space via the parameter k. The exceeding space ($exceeding$) occupied by the Edge-*k* table is given by Equation 3.

$$exceeding = (\eta_{nulls} * null_size) + (\eta_{rows} - |V_{neigh}|) * (size(v_{id}) + overhead) \tag{3}$$

where $V_{neigh} \subseteq V$ is the subset of vertices of the graph containing a neighborhood and $size(v_{id})$, $overhead$ and $null_size$ are, respectively, the sizes in bytes of the vertex *id*, of the row overhead and of the storage of a *null destination*. This equation considers space occupied by *null* values plus the wasted space occupied by vertices v_{id} without a neighborhood. The $null_size$ depends on the implementation. If an RDBMS does not compact *null* values, it will reserve the standard size in bytes of the corresponding attribute. In

such a case, $null_size = size(v_{id})$. However, RDBMS usually apply techniques to save space when storing *null* values, and the real size varies according to the product.

3.2. Adaptation of Single Source Shortest Path (SSSP) procedure

A path $p = v_0 \overset{p}{\rightsquigarrow} v_n$ is a sequence of non-repeating edges $\langle v_0, v_1\rangle, \langle v_1, v_2\rangle, \ldots, \langle v_{n-1}, v_n\rangle$, in which there is one *source* vertex v_0 and one *destination* vertex v_n. A common additional attribute in the set of edges is a real-valued weight w, which defines a numerical measure between two adjacent edges in a path. The weight of a path is determined by the sum of the weigths of its constituent edges. The Single Source Shortest Path (SSSP) problem consists of finding the shortest path from a given source vertex v_0 to each vertex $v_i \in V$, that is, the paths $v_0 \overset{p}{\rightsquigarrow} v_i$ of minimal weight. To calculate SSSP in Edge-*k*, we adapted the corresponding procedure of the existing frameworks. Our implementation is illustrated in Figure 2. In short, the procedure iteratively joins a temporary table containing the weight of the current shortest path of every discovered vertex with the edge table, to visit new vertices and expand the frontier.

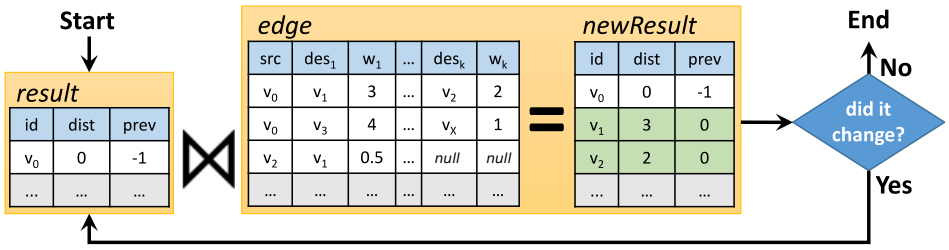


Figure 2. General idea of the SSSP procedure in our proposal.

The procedure is as follows. First, a temporary table result is created to register all visited vertices. For each of them, it keeps track of the unique identifier (*id*), the minimal distance to the path source vertex (*dist*) and the previous vertex (*prev*) in the current shortest path, allowing to identify which edges compound such path. The procedure starts by inserting the *source* vertex v_0 in table result, which has a distance of 0 and no previous vertex (*i.e.* -1). Thereafter, result is joined with the edge table to expand the neighborhood of result's vertices and update the shortest path of every destination vertex, producing a new temporary table (called table newResult). To update the distances in each iteration, the procedure sums the accumulated distance ($dist$ in table result) and the respective distance to the next neighboring vertex (recall that columns $w_1 \ldots w_k$ of the edge table store the distances from src to $des_1 \ldots des_k$ in edge E_k).

Regarding the example in Figure 2 (which corresponds to the graph of Figure 1), after first iteration the distances from v_0 to v_1 and from v_0 to v_2 are, respectively, (0+3) and $(0 + 2)$. The procedure aims at minimizing such distances, prioritizing the minimal paths discovered so far. Therefore, in second iteration, the weight of path $v_0 \overset{p}{\rightsquigarrow} v_1$ is updated to 2.5, through the expansion of vertex v_2, and so on. If tables result and newResult are equal, then all the shortest paths have already been found and the procedure finishes. Otherwise, table result is replaced by newResult and another iteration is executed.

Let's look at the major competitors of Edge-*k*, in which the key difference is in joining table result with the edge table. In the existing frameworks, the edge table stores

only one destination per row. Therefore, when joining the tables result and edge, the procedure just executes an ordinary join operation. In Edge-*k*, on the other hand, for any value of k such join operation is accompanied by an unnesting operation, which consists of transforming the multiple $des_1 \ldots des_k$ columns into separate rows.

4. Experimental Results

4.1. Setup

We evaluate our Edge-*k* method on both real and synthetic datasets, and due to space limitations we present the results on representative datasets. The real dataset was extracted from the *Digital Bibliography & Library Project*[1] (DBLP), containing information about authors and their publications. We modeled the set of authors as vertices and their relationships as edges, measuring the collaboration among them with a weight value. To analyze the impact of varying the number of vertices in the graph and the overall rate of neighboring vertices, we employed two synthetic graphs, generated through the Networkx[2] framework. The first one, called Newman-Watts-Strogatz dataset, models a small-world graph. Initially, it creates one ring including all $|V|$ vertices of the graph. Then, each vertex in the ring may be randomly connected to other X vertices, which creates shortcuts between vertices in the ring structure. The second synthetic graph, named Erdos-Renyi dataset, is defined by assigning edges between pairs of vertices with a probability p. Parameters $|V|$, X and p are provided by the user.

Table 1 details the characteristics of all datasets, showing the number of vertices and edges, as well as the minimum (*min*), maximum (*max*) and average (*avg*) vertex *degrees*. For the Newman-Watts-Strogatz graphs, we fixed X to 200 shortcuts and varied the number of vertices $|V|$. Notice that their vertex degree size do not display a significant change. For the Erdos-Renyi graphs, we fixed $|V|$ and varied the probability p of edge creation with the values 1%, 3% and 5%. In this case, the vertex degree varies significantly, allowing the validation of more distinct scenarios. The experiments were run in the RDBMS PostgreSQL 9.5.6. We used a machine equipped with an Intel® Core™ i7-2600 @ 3.40GHz processor, 8GB of DDR3 1333MHz RAM memory, two SATA 6Gb/s 7200RPM hard disks set up in RAID 0, and Linux operating system Fedora release 25.

Table 1. Characteristics of the real and synthetic datasets.

Measure	DBLP	Newman-Watts-Strogatz			Erdos-Renyi		
		$X = 200$	$X = 200$	$X = 200$	$p = 1\%$	$p = 3\%$	$p = 5\%$
$\|V\|$	1,909,226	9,000	18,000	27,000	9,000	9,000	9,000
$\|E\|$	19,194,624	1,979,410	3,959,970	5,940,340	810,570	2,428,600	4,048,600
min *degree*	1	203	204	205	56	214	369
max *degree*	2100	245	238	247	126	334	536
avg *degree*	10.05	220.12	219.99	220.01	90.06	269.84	449.84

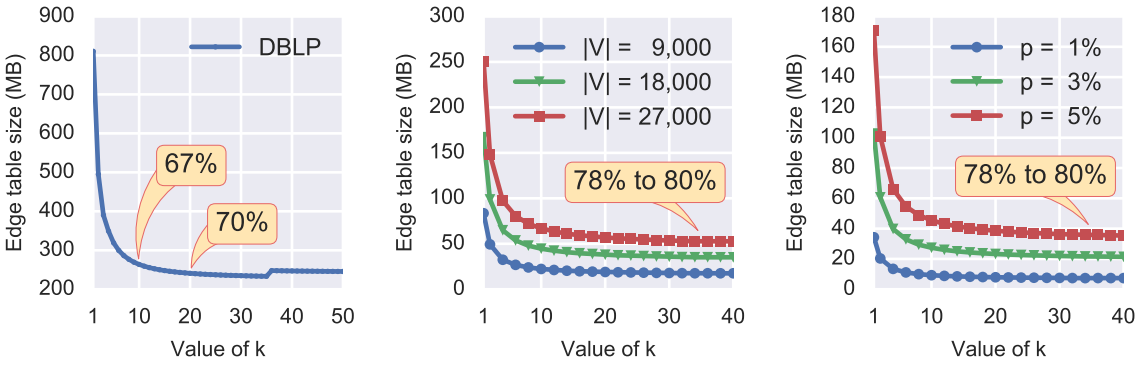
4.2. Dataset Size Reduction

This section presents the dataset size reduction achieved when employing Edge-*k* to group neighboring vertices in a same row in the edge table. Consequently, the overall amount

[1]Collected at March 16th, 2017 from `http://dblp.uni-trier.de/xml/`

[2]`https://github.com/networkx/networkx`

of row overhead space is decreased, along with the number of times that each *source* vertex is stored. Figure 3 shows the size reduction achieved for each dataset. Considering the results for the DBLP dataset shown in Figure 3a, we observe a reduction of 67% at $k = 10$, this value of k coincides with the average degree as shown in Table 1. As k increases, the overall reduction stabilizes in approximately 70% ($k = 20$ and beyond).



(a) DBLP. (b) Newman-Watts-Strogatz. (c) Erdos-Renyi.

Figure 3. Size reduction of the edge table achieved in each dataset as *k* varies.

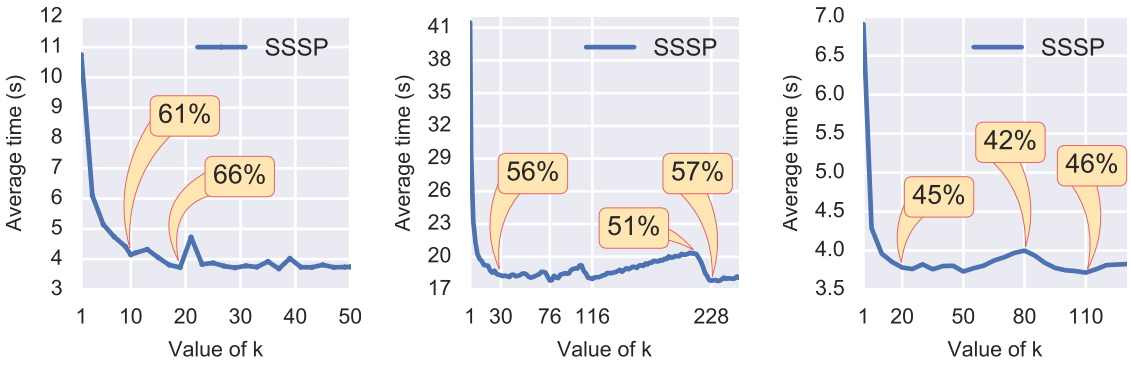
Regarding the synthetic datasets, both Figure 3b and 3c depict a similar behavior. Regardless of varying the number of vertices (in the Newman-Watts-Strogatz graph) or the vertex degree (in the Erdos-Renyi graph), the reduction achieved for $k \geq 30$ ranges from 78% to 80%. As explained by Equation 1, the sharp reduction in the dataset size was analogous to the decrease of η_{rows}, for the same range of k values. It is worth noticing the potential of our approach in terms of relation size reduction. This is especially evident when k varies from 1 to 10, where a drastic reduction occurs in the size of the edge table — achieving 67% already for the first values. For greater values of k, the neighbors of a vertex are able to be stored in less rows — most of them eventually fit in one row. In this scenario, as k grows, the number of *null* values stored in the edge table also increases, since the table becomes larger than required. As *null* values occupy an irrelevant space in the RDBMS PostgreSQL (apart from the bitmap structure used to keep track of *null* values, if any, in each row), having more *null* values does not harm the storage needs, which is why the size reduction remains stable even for greater values of k.

4.3. SSSP Query Processing Time

To analyze the impact of the dataset size reduction on queries, this section focuses on the SSSP query processing times. For each dataset, we defined a query whose starting point was the vertex having the highest degree — the largest number of neighbors. Each query was executed 30 times and had their execution time measured. Thereafter, we obtained the average execution time, after discarding 10% of the shortest and 10% of the longest times so as to rule out outliers. For the DBLP dataset, we analyzed the average execution times from 2 to 5 iterations of the SSSP algorithm — *i.e.* the number of times the tables edge and result are joined to compute the minimal paths, as described in Section 3.2. The SSSP query processing time was not evaluated for the first iteration alone, since the same outcome could be obtained by simply querying the edge table, filtering it by the *source* vertex, rather than performing join operations. For the synthetic datasets (from the starting

point we defined), a maximum of 4 iterations was enough to visit all vertices in the graphs. Therefore, we executed the SSSP query on them without limiting the iterations.

Figure 4 presents the average SSSP query time when varying k for two iterations on the DBLP graph (Figure4a), for the Newman-Watts-Strogatz graph containing 27,000 vertices (Figure4b) and for the Erdos-Renyi graph with probability p = 1% (Figure4c). The query processing time reduction observed in Figure 4a achieved 61% for the average degree (*i.e.* for $k = 10$, as shown in Table 1), reaching 66% for $k = 20$ and remaining stable after this point. It is worth noticing that this behavior is analogous to the dataset size reduction. Figure 4b shows a similar behavior, presenting an initial reduction of 56% at $k = 30$ and stabilizing after this value with slight variations. Particularly, such variations range from 51% to 57% for values of k between 30 and 247, *i.e.* up to the maximum degree of the dataset, as presented in Table 1. This same behavior can be observed in Figure 4c, and in the other evaluated synthetic datasets, in which the reduction was of 45% for $k = 20$ and there were small variations ranging from 42% to 46% for values of k between 20 and 126. The observed peaks of Figures 4a to 4c correspond to the oscillations in the number of *nulls* (η_{nulls} from Equation 2) while evaluating different values of k.



(a) DBLP with 2 iterations.

(b) Newman-Watts-Strogatz with $|V| = 27,000$.

(c) Erdos-Renyi with $p = 1\%$.

Figure 4. SSSP query processing time in three scenarios for varying *k* in Edge-*k*.

According to Equation 1, the ideal cases occur when the overall degrees are divisible by k, which is when the average number of *null* values is smaller in the edge table. We can observe in Figure 4b that the best value between the minimum and maximum degrees is $k = 228$. However, that is not the single good value, since $k = 116$ and $k = 76$ also provide equivalent reductions in query processing time. The fact of several k values being appropriate is explained by the modulo operator in Equation 2: the number of *null* values rises — as does the query processing time — and falls at the next well-suited value of k — along with the query processing time as well —, by which the overall neighborhood is divisible. That is, minimizing both η_{rows} and η_{nulls} (Equations 1 to 3) is important when analyzing a proper value for k, which we empirically evaluated in the experiments.

Additional experiments not presented in this paper were performed for values of k greater than the maximum degree of the graph. However, we found that beyond this point the average query time only increases. This is because the amount of unnecessary columns in edge table keeps rising, adding more validations to the unnesting operation.

The adapted SSSP algorithm performs a loop through each one of the k *destination* vertices. Thus, assigning a value too large for k affects the overall performance.

As previously mentioned, the dataset size reduction is analogous to the average query processing time reduction, which indicates a correlation between them. To measure it, we calculated the Pearson coefficient between the amount of data blocks of edge table, while varying its k value, and the average elapsed time of SSSP queries, in which values closer to 1 imply a positive linear correlation. In the DBLP dataset, the coefficient obtained was 0.976 for 2 iterations and 0.827 for 5 iterations. In the synthetic datasets, the coefficient obtained was at least 0.943. Accordingly, reducing the dataset size (focusing on the edge table) also lowers the query processing time. This corresponds to the expected result, since reducing the dataset size should also decreases the amount of data blocks in the disk, minimizing the number of I/O operations and optimizing query processing.

4.4. Scalability Evaluation

We carried out another analysis to identify the highest reduction achieved in the SSSP query processing time varying: (i) the number of iterations performed over the DBLP graph; and (ii) the distinct parameter value used to generate the synthetic graphs. Figure 5 summarizes the highest query processing time reduction achieved over the DBLP dataset for each number of iterations employed. In the figure we can observe that the highest reduction has been achieved with 2 iterations, and as the number of iterations increases, the reduction decreases. This behavior shows that, as the tables edge and result are repeatedly joined, the query performance tends to degrade in both storage approaches. Nevertheless, solutions that employ too many joins are not usually well suited for RDBMS. For example, considering our real dataset, performing more than 5 iterations means looking for collaborations spanning at least 5 authors, which tends to be less frequent. Hence, in terms of a real application, it would not be particularly interesting to keep running the SSSP procedure any further than the number of iterations we already ran for.

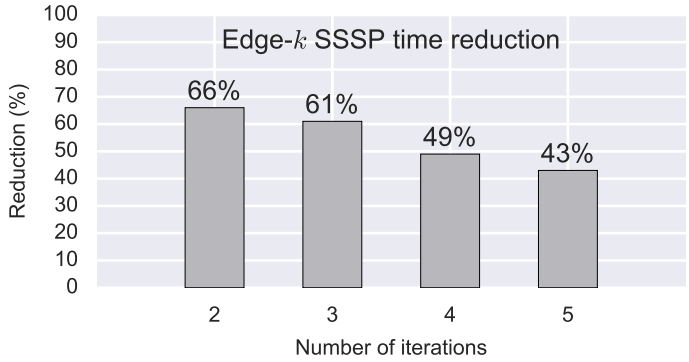
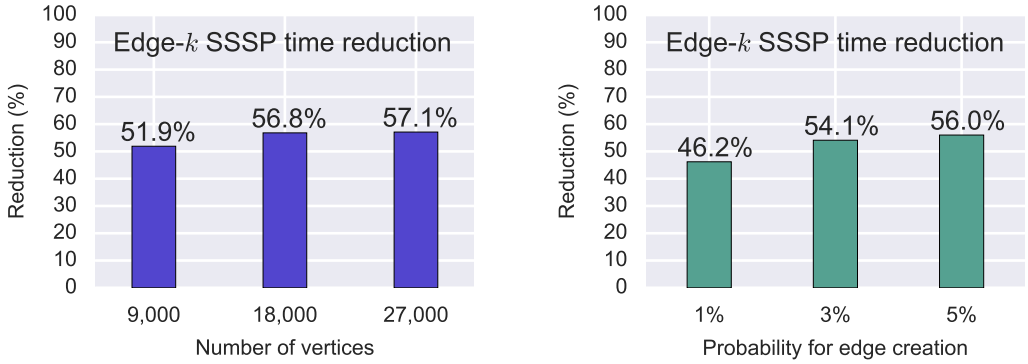


Figure 5. Highest query processing time reduction achieved, thought Edge-*k* method, for each number of iterations over the DBLP dataset.

Considering the synthetic datasets, Figure 6 shows the highest query processing time reductions according to the parameters considered. By increasing the number of vertices $|V|$ in the Newman-Watts-Strogatz graph, as shown in Figure 6a, query performance also increases, reaching up to 57% of query processing time reduction for the greatest number of vertices $|V|$ = 27,000. Specifically, such larger reduction was achieved when

k = 76, k = 116 or k = 228, as previously discussed regarding Figure 4b. Figure 6b depicts a similar behavior: by increasing the probability p of edge creation — which also changes the minimal, maximum and average degrees —, the corresponding query processing time reduction also increases. Both figures spot that our proposal provides even better results as the volume of data grows. That is, graphs that have more rows in the edge table (and, consequently, a higher amount of repeated *source* vertex in the conventional implementation) allow a higher reduction in both query processing time dataset size.



(a) Newman-Watts-Strogatz. (b) Erdos-Renyi.

Figure 6. Highest query processing time reduction achieved, thought Edge-*k* method, for each parameter of Table 1 over the synthetic datasets.

The experimental results show that, regardless of the number of vertices and of resulting vertex degrees, it is possible to accomplish good performance improvements. Considering the number of iterations performed by the SSSP procedure, the reduction in query processing time decreases as the number of iterations increases, as already seen in various RBDMS-based solutions involving too many join operations. However, regarding the real scenario evaluated, an indirect relationship between two authors having more than three coauthors in the path joining them (*i.e.* requiring more than 5 iterations) is less likely to occur, thus irrelevant for our experimental analyses.

5. Conclusion

Graph data stored in RDBMS are growing in volume and quantity, requiring efficient management applications to properly deal with them. Despite the several ways to represent graph data in RDBMS, most of the frameworks focus on a single approach to store the edges of a graph. In this context, we have proposed Edge-*k*, a storage approach for the edge table of a graph, which consists of grouping at least part of the neighborhood of each vertex in a same row. This work has been conducted based on the assumption that this approach is able to reduce the dataset size and, consequently, the processing time of queries over the edge table. In fact, regarding query processing time, the experimental results show a reduction around 66% on real and 57% on synthetic datasets, which highlight the impact of our proposal. Although we have evaluated our storage approach for SSSP queries, many other graph applications benefit from it as well.

Due to space limitations, we focused on a frequent query type implemented by the general graph data management frameworks in relational databases. Nonetheless, this work can be extended in several directions. Firstly, a future work will explore the

use of index structures in our storage approach, as well as different applications in other scenarios. Secondly, our approach could be extended to include edges connected to three or more vertices at a time, that is, to hypergraphs. Lastly, other databases not necessarily adopting the Relational Model (*i.e.* NoSQL databases) will be analyzed, and we will compare our proposal to the Neo4j NoSQL datastore.

Acknowledgments This research was supported, in part, by CAPES, CNPq and FAPESP. The authors would like to thank Gabriel Spadon and Willian D. Oliveira for their contribution on the paper and the server infrastructure provided by FAPESP (grant project 2014/26678-6).

References

Barabási, A.-L. and Pósfai, M. (2016). *Network science*. Cambridge University Press.

Bornea, M. A., Dolby, J., Kementsietsidis, A., Srinivas, K., Dantressangle, P., Udrea, O., and Bhattacharjee, B. (2013). Building an efficient rdf store over a relational database. In *Proceedings of the 2013 SIGMOD*, pages 121–132, New York, NY, USA. ACM.

Chen, R. (2013). Managing massive graphs in relational DBMS. In *2013 International Conference on Big Data*, pages 1–8, Santa Clara, CA, USA. IEEE.

Fan, J., Raj, A. G. S., and Patel, J. M. (2015). The case against specialized graph analytics engines. In *Proceeding of the 2015 CIDR*, Asilomar, CA, USA. Online Proceedings.

Gao, J., Jin, R., Zhou, J., Yu, J. X., Jiang, X., and Wang, T. (2011). Relational approach for shortest path discovery over large graphs. *PVLDB Endowment*, 5(4):358–369.

Gao, J., Zhou, J., Yu, J. X., and Wang, T. (2014). Shortest path computing in relational DBMSs. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):997–1011.

Jindal, A., Madden, S., Castellanos, M., and Hsu, M. (2015). Graph analytics using vertica relational database. In *International Conference on Big Data*, pages 1191–1200. IEEE.

Kyrola, A., Blelloch, G., and Guestrin, C. (2012). Graphchi: Large-scale graph computation on just a pc. In *Proceedings of the 10th Conference on Operating Systems Design and Implementation*, pages 31–46, Berkeley, CA, USA. USENIX Association.

Levandoski, J. J. and Mokbel, M. F. (2009). RDF data-centric storage. In *2009 International Conference on Web Services*, pages 911–918, Los Angeles, CA, USA. IEEE.

Neumann, T. and Weikum, G. (2010). The RDF-3X engine for scalable management of RDF data. *The VLDB Journal*, 19(1):91–113.

Silva, D. N. R. D., Wehmuth, K., Osthoff, C., Appel, A. P., and Ziviani, A. (2016). Análise de desempenho de plataformas de processamento de grafos. In *31º Simpósio Brasileiro de Banco de Dados, SBBD*, pages 16–27, Salvador, BH, Brasil. SBC.

Sun, W., Fokoue, A., Srinivas, K., Kementsietsidis, A., Hu, G., and Xie, G. (2015). Sqlgraph: An efficient relational-based property graph store. In *Proceedings of the 2015 SIGMOD*, pages 1887–1901, New York, NY, USA. ACM.

paper:171411

STACY: Um Novo Algoritmo para Automaticamente Classificar a Força dos Relacionamentos ao Longo dos Anos

Michele A. Brandão, Pedro O. S. Vaz de Melo, Mirella M. Moro

[1]Universidade Federal de Minas Gerais – Belo Horizonte – MG

{micheleabrandao,olmo,mirella}@dcc.ufmg.br

Abstract. *Understanding the relationships between people in a social network and measuring their strength over time are interesting problems with distinct applications. Here, we propose a new algorithm (STACY) to automatically classify tie strength in eight different classes by considering the temporal aspect. Our results show: such classes represent different behaviors, and STACY identifies strong relationships that persist more than the ones classified by a state of the art algorithm.*

Resumo. *Entender relacionamentos entre pessoas em uma rede social e medir suas forças ao longo do tempo são problemas interessantes com várias aplicações. Aqui, propomos um novo algoritmo (STACY) para automaticamente classificar a força dos relacionamentos em oito diferentes classes considerando o aspecto temporal. Os resultados mostram que tais classes representam e distinguem relacionamentos, e STACY é capaz de identificar relacionamentos fortes que persistem mais do que os classificados por um algoritmo do estado-da-arte.*

1. Introdução

Tempo é um fator fundamental na caracterização da natureza e força dos relacionamentos. Por exemplo, conhecidos podem se tornar amigos (e vice-versa). Relações que variam ao longo do tempo podem ser modeladas como uma rede social temporal, ou grafo temporal, em que cada nó é uma pessoa e há uma aresta entre dois nós em um dado tempo se eles compartilharam qualquer relação naquele tempo. Abordagens que focam em grafos agregados estáticos (não-temporais) dão o mesmo grau de importância para todas as interações anteriores. No entanto, os relacionamentos mais recentes são geralmente mais representativos da classe da relação [Gilbert and Karahalios 2009]. Se em grafos estáticos esses aspectos temporais são agregados e, portanto, escondidos, em grafos temporais eles estão naturalmente presentes, servindo como um modelo apropriado para redes dinâmicas.

No entanto, a computação de propriedades temporais de redes sociais é um desafio. Por exemplo, o coeficiente de agrupamento de uma rede no tempo t_1 não é necessariamente o mesmo no tempo t_2, pois as interações podem aparecer ou desaparecer ao longo do tempo. Além disso, a precisa ordenação temporal das arestas influencia essencialmente a noção de adjacência dos nós e acessibilidade em tais redes [Nicosia et al. 2013]. Portanto, conceitos e métricas projetados e aplicados à análise de redes estáticas devem ser adaptados e estendidos a redes que variam em função do tempo. A força do relacionamento (também chamada força dos laços) é um desses conceitos, definido originalmente como uma fusão do tempo da relação, da força emocional, da intimidade e dos serviços recíprocos que representam um relacionamento entre pessoas [Granovetter 1973].

Neste artigo, propomos um novo algoritmo intitulado *STACY - Strength of Ties Automatic-Classifier over the Years*. *STACY* usa recursos de redes sociais para classificar a força dos relacionamentos em oito classes (*strong*, *bridge+*, *bridge*, *transient*, *periodic*, *bursty*, *weak* e *random*). Ele baseia-se em um algoritmo existente – RECAST (*Random rElationship ClASsifer sTrategy*) [Vaz de Melo et al. 2015], que foi aplicado para medir a força de relacionamentos em redes móveis, e não redes sociais de co-autoria que são o nosso foco. Aqui, também melhoramos o desempenho de tal algoritmo (agora denominado *fast-RECAST*) e comparamos os resultados com os gerados pelo *STACY*.

Nesta pesquisa, vemos a força de um relacionamento como a probabilidade de sua (re) aparência no futuro. Estimamos essa probabilidade usando três características das arestas das redes sociais relacionadas à força do relacionamento (persistência das arestas, sobreposição de vizinhos e frequência de co-autoria). Ademais, contrastamos os resultados estimando essa probabilidade com persistência das arestas e sobreposição de vizinhos (ambas são consideradas em fast-RECAST). Essas propriedades captam a regularidade da interação e a similaridade entre os indivíduos envolvidos nessa interação. Nosso principal objetivo é verificar se as atuais definições da força dos relacionamentos mantêm-se para redes sociais temporais. Para isso, analisamos o dinamismo da força dos relacionamentos observando a persistência e a transformação ao longo do tempo.

A seguir, a Seção 2 apresenta os trabalhos relacionados, e a Seção 3 descreve conceitos e o RECAST original. A Seção 4 detalha a força dos relacionamentos, o RECAST com multiprocessamento e novo algoritmo *STACY*. A Seção 5 analisa persistência e transformação dos relacionamentos, e a Seção 6 conclui este trabalho.

2. Trabalhos Relacionados

A força dos relacionamentos pode ser calculada considerando propriedades topológicas e/ou semânticas na rede social [Alves et al. 2016, Castilho et al. 2017]. As propriedades topológicas capturam características estruturais do grafo que constitui a rede social [Zaki and Meira Jr 2014]. Por exemplo, Brandão e Moro [2015] usam sobreposição de vizinhos para medir a força dos relacionamentos em redes de co-autoria. Por outro lado, as propriedades semânticas captam características não estruturais de nós e arestas em redes sociais. Por exemplo, Gilbert e Karahalios [2009] definem relacionamentos no Facebook considerando o histórico de interações. Por fim, também é possível combinar ambos os tipos de propriedades. Por exemplo, Zignani et al. [2016] usam propriedades de topológicas do grafo e temporais (semânticas) para prever a força dos relacionamentos.

Uma importante propriedade semântica é o aspecto temporal. Mesmo com tanta pesquisa em redes sociais, a combinação de força dos relacionamentos e aspectos temporais ainda não foi amplamente explorada. Por exemplo, Karsai et al. [2014] usa força de empate para caracterizar o impacto de interações heterogêneas e variáveis no tempo sobre a propagação de rumores. Eles consideram a evolução temporal da força dos laços, mas não propõem uma nova maneira de medir essa propriedade incluindo o tempo. Por outro lado, Kostakos [2009] e Nicosia et al. [2013] propõem um conjunto de propriedades de grafos que consideram o aspecto temporal em seu cálculo. Tais estudos mostram que muitas dessas propriedades precisam ser calculadas de forma diferente das redes estáticas.

Um problema relacionado é como definir relacionamentos fortes e fracos em redes temporais. Por exemplo, Laurent et al. [2015] definem laços fortes como interações

frequentes que conectam os nós intra-comunidades e modelam a estrutura da rede localmente, ao passo que os laços fracos são interações infrequentes situadas entre comunidades e mantêm a estrutura de rede globalmente conectada. Karsai et al. [2014] consideram tanto a quantidade de interações como o tempo das interações para definir a força dos relacionamentos. Então, os laços fortes representam interações frequentes ao longo do tempo entre pares de indivíduos, enquanto laços fracos ocorrem apenas ocasionalmente. Por outro lado, Nicosia et al. [2013] definem dois nós i e j como fortemente conectados se estiverem em uma relação não simétrica (i está temporariamente conectado a j, mas não vice-versa), enquanto que estão fracamente conectados se estão em uma relação simétrica (ambos i está temporariamente conectado a j, e j está temporariamente conectado a i).

Neste artigo, consideramos o conceito de laços fortes e fracos para redes sociais temporais baseados na ideia de Karsai et al. [2014], ou seja, um laço forte persiste ao longo do tempo, e um laço fraco ocorre esporadicamente. No entanto, Karsai et al. [2014] caracterizam a força dos relacionamentos com base em uma única janela temporal. Já nós verificamos experimentalmente se a janela temporal é um fator para caracterizar a força dos relacionamentos, analisando a persistência e a transformação dos relacionamentos ao longo do tempo com o nosso novo algoritmo *STACY*. Então mostramos que, de fato, a força dos relacionamentos é muito sensível à janela de tempo usada para calculá-la.

3. Conceitos Fundamentais

Nesta seção, descrevemos modelos para redes sociais temporais e o RECAST original.

Modelos para Redes Sociais Temporais. Inicialmente, definimos formalmente um modelo para redes sociais temporais. Em vez de propor um novo modelo, utilizamos a ideia de Vaz de Melo et al. [2015], que consiste em modelar uma rede temporal para estudar redes móveis. Seguindo tal ideia, associamos uma hora de início e uma duração a cada co-autoria. Em seguida, uma *rede social de co-autoria temporal* é modelada como um grafo $G_k(\mathcal{V}_k, \mathcal{E}_k)$ em que o tempo é discretizado em etapas de duração Δ, a duração $\delta = 1\ Ano$, e k é o período de tempo em que ocorre uma co-autoria. O conjunto de nós $\mathcal{V}_k$ é formado por todos os nós da rede durante o k-th tempo, e o conjunto de arestas $\mathcal{E}_k$ é composto de co-autorias durante o mesmo período de tempo. Assim, há uma aresta em $\mathcal{E}_k$ entre dois nós i e j com $i, j \in \mathcal{V}_k$, se i e j são co-autores de uma publicação durante o tempo k. Uma representação da variação temporal em redes de co-autoria pode ser definida por um grafo temporal acumulado $G_t(\mathcal{V}_t, \mathcal{E}_t)$, onde $G_t = G_1 \cup G_2 \cup ... \cup G_t$. Então, $\mathcal{V}_t$ e $\mathcal{E}_t$ são o conjunto de todos os nós e arestas nas redes, respectivamente, no período de tempo 0 a t. Como G_t acumula todas as co-autoria dos conjuntos de dados e evolui ao longo do tempo, tal grafo agregado contém relacionamentos sociais e aleatórios. Também de acordo com Vaz de Melo et al. [2015], é necessária uma versão aleatória G_t^R do grafo temporal acumulado G_t para analisar os padrões dessa rede. O grafo aleatório deve ter características topológicas da rede social semelhantes ao grafo G_t (número de nós, arestas e distribuição dos graus), e os nós estão conectados de forma diferente de G_t.

O RECAST Original. Seguindo a descrição do modelo, resumimos a implementação original do algoritmo, chamado RECAST [Vaz de Melo et al. 2015]. Uma contribuição deste trabalho é modificá-lo para medir a força de laços em grandes redes sociais temporais. Escolhemos RECAST por ser o único que atribui classes diferentes à força dos relacionamentos em redes temporais. Esse algoritmo diferencia interações aleatórias das

sociais (*amigos* - chamados como fortes, *pontes* e *conhecidos* - chamados de fracos). RECAST implementa o modelo descrito anteriormente construindo tanto G_t como G_t^R. Dois algoritmos são necessários para gerar G_t^R de G_t: `RND` e `T-RND` [Vaz de Melo et al. 2015]. Dado um grafo $G(\mathcal{V}, \mathcal{E})$, $\texttt{RND}(G)$ retorna um grafo aleatório $G_t(\mathcal{V}^R, \mathcal{E}^R)$ com o mesmo número de nós, arestas e distribuição de graus que G. A única diferença entre G e G^R é a conexão entre os nós, que é o foco de nosso estudo. Portanto, `RND` atribui uma aresta entre nós i e j com probabilidade $p_{i,j} = (d_i \times d_j)/\sum_{k=1}|V|d_k$, em que a distribuição de graus é $D = (d_1, d_2, ..., d_n)$ de G com n nós. O segundo algoritmo `T-RND` é uma extensão de `RND` e gera grafos aleatórios para redes temporais G_t. Assim, `T-RND` $(G_1 \cup G_2 \cup ... \cup G_t)$ recebe um conjunto de grafos de eventos consecutivos G_t e retorna um grafo temporal aleatório G_t^R. Esse algoritmo constrói G_t^R executando `RND` em cada grafo de evento G_t e depois acumulando-o como $G_t^R = \texttt{RND}(G_1) \cup \texttt{RND}(G_2) \cup ... \cup \texttt{RND}(G_t)$. Em resumo, tanto `RND` quanto `T-RND` reproduzem aleatoriamente o número total de co-autores com autores (no nosso contexto) distintos que cada pessoa tem em um instante de tempo.

4. Algoritmos para Medir a Força dos Relacionamentos

Nesta seção, revisitamos o conceito da força dos relacionamentos (Seção 4.1). Também propomos fast-RECAST (Seção 4.2) e *STACY* (Seção 4.3).

4.1. Revisitando o Conceito da Força dos Relacionamentos

Dado um grafo temporal $G_k(\mathcal{V}_k, \mathcal{E}_k)$, onde k é o intervalo de tempo em que ocorre uma co-autoria, um relacionamento (i, j) é provável que seja forte se estiver presente em G_k para a maioria dos valores de k. Já o relacionamento (i, j) é provável que seja fraco se estiver presente em G_k para alguns valores de k. Em outras palavras, é provável que os relacionamentos fortes persistam ao longo do tempo, e os fracos provavelmente ocorrem esporadicamente. Outra característica de um relacionamento forte (i, j) é que provavelmente i e j têm muitos vizinhos em comum. Como discutido anteriormente, os nós que têm muitos vizinhos em comum são mais propensos a persistirem ao longo do tempo.

Dadas estas duas características, fast-RECAST agrupa os relacionamentos em quatro classes: *strong* (amigos), *weak* (conhecidos), *bridges* e *random*. Cada classe define um nível de força dos laços: *strong* são relacionamentos que persistem ao longo do tempo e compartilham muitos vizinhos em comum; *weak* não persistem ao longo do tempo, mas compartilham muitos vizinhos em comum; *bridges* persistem ao longo do tempo, mas não possuem muitos vizinhos em comum; e *random* não persistem ao longo do tempo e compartilham poucos vizinhos em comum. Utilizando essas quatro classes de relacionamentos, investigamos se a força dos laços se transformam ao longo do tempo. Com essa análise, somos capazes de aprofundar em redes sociais temporais e responder a perguntas como: relacionamentos *strong* são mais propensos a permanecer *strong* no futuro? Ou relacionamentos *weak* são mais propensos a tornar-se *strong* ou *random*? Além disso, considerando uma terceira propriedade, frequência de co-autoria, um relacionamento forte (i, j) é provável que i e j tenham uma alta frequência de co-autoria. *STACY* usa essas três características para classificar relacionamentos em oito classes (Seção 4.3).

4.2. RECAST com Multiprocessamento

A construção de G_t^R por meio de `T-RND` aumenta a complexidade de RECAST para $O(t \times (|\mathcal{V}_t| + |\mathcal{E}_T^R|))$. Dessa forma, aplicamos um módulo Pool de multiprocessamento do Python

Algoritmo 1 fast-RECAST: um código paralelizado para classificar arestas de G_t em aleatórias e sociais – strong, weak ou bridge.

Require: $p_{rnd} \geq 0$
1: **return** $class(i,j)\ \forall (i,j) \in U_t E_t$
2: Construir G_t^R e o conjunto $\texttt{RND}(G_1), ..., \texttt{RND}(G_t)$ utilizando `T-RND` com **pool.map_async**
3: Obter $\overline{F}_{to}(x)$ e $\overline{F}_{per}(x)$ de G_t^R utilizando **pandas dataframe**
4: Obter $\overline{x}_{to} | \overline{F}_{to}(\overline{x}_{to})$ e $\overline{x}_{per} | \overline{F}_{per}(\overline{x}_{per}) = p_{rnd}$ com **pool.apply_async**
5: **for all** arestas$(i,j) \in E_t$ **do**
6: **if** $per(i,j) > \overline{x}_{per}$ e $to(i,j) > \overline{x}_{to}$ **then**
7: $class(i,j) \longleftarrow Strong$
8: **else if** $per(i,j) > \overline{x}_{per}$ e $to(i,j) \leq \overline{x}_{to}$ **then**
9: $class(i,j) \longleftarrow Bridges$
10: **else if** $per(i,j) \leq \overline{x}_{per}$ e $to(i,j) > \overline{x}_{to}$ **then**
11: $class(i,j) \longleftarrow Weak$
12: **else**
13: $class(i,j) \longleftarrow Random$

(módulo baseado em processos de comunicação para escrever programas concorrentes[1]) em tal etapa do RECAST, a fim de reduzir a sua complexidade. Chamamos essa nova versão do algoritmo com multiprocessamento de fast-RECAST. A ideia é que mais de um grafo aleatório G_t^R é construído de cada vez em um computador multi-core. Assim, o novo custo computacional é $O(\frac{t}{p} \times (|\mathcal{V}_t| + |\mathcal{E}_t^R|))$, onde p é o número de processos. Após construir G_t^R, a complexidade da classificação é $O(|E_t^R| \times |\mathcal{V}_t|)$, onde $O(|V_t|)$ é o custo de computar as duas características de uma aresta na rede social. Também adicionamos um módulo Pool de multiprocessamento do Python para chamar as funções para calcular a persistência das arestas e a sobreposição de vizinhos dos grafos agregados. Ambos os recursos são computados em paralelo e de forma assíncrona.

O Algoritmo 1 resume o código de fast-RECAST[2] com multiprocessamento (linhas 2 e 4) e uma otimização na utilização da memória, aplicando Pandas dataframe do python para armazenar os grafos antes de processá-los (linha 3). Por ser redes de co-autoria, renomeamos as arestas de *friend* para *strong* e *acquaintance* para *weak*.

4.3. STACY - *Strength of Ties Automatic-Classifier over the Years*

Agora propomos um algoritmo para classificar automaticamente a força dos relacionados chamado *STACY - Strength of Ties Automatic-Classifier over the Years*, uma melhoria do fast-RECAST que considera o peso das arestas (frequência de co-autoria) diferente de 1. Para distribuir a frequência de co-autoria no grafo aleatório G_t^R, usamos o mesmo algoritmo para distribuir o grau das arestas proposto por Miller and Hagberg [2011]. A diferença é que atribuímos um peso às arestas entre i e j com probabilidade $p_{ij} = (w_i \times w_j) / \sum_{k=1}^{|V|} w_k$ para uma distribuição de pesos $D_w = (w_1, w_2, ..., w_n)$ de G com N nós.

Estendendo a descrição de G_t na Seção 3, definimos um grafo temporal acumulado com peso $G_t^W = G_1^W \cup G_2^W \cup ... \cup G_t^W$. $\mathcal{V}_t$ e $\mathcal{E}_t$ são o conjunto de todos os nós e arestas com peso na rede social, respectivamente, no período de tempo 0 a t. Como G_t^W acumula todas as co-autorias dos conjuntos de dados e evolui ao longo do tempo, o grafo agregado possui relacionamentos sociais e aleatórios. Além disso, consideramos uma versão com

[1]Python e multiprocessamento: `docs.python.org/2/library/multiprocessing.html`
[2]Código fonte disponível em *`github.com/lab-csx-ufmg/RECAST`*

Tabela 1. Classes de relacionamentos do STACY.

Classe	Persistência das arestas	Sobreposição de vizinhos	Frequência de co-autoria
1 - strong	social	social	social
2 - bridge+	social	random	social
3 - transient	random	social	social
4 - periodic	social	social	random
5 - bursty	random	random	social
6 - bridge	social	random	random
7 - weak	random	social	random
8 - random	random	random	random

Algoritmo 2 STACY: um código paralelizado para classificar arestas com pesos de G_t^W em oito classes diferentes.

Entrada: Grafo temporal agregado com peso - G_t^W

Require: $p_{rnd} \geq 0$

1: **return** $class(i,j)\ \forall (i,j) \in U_t E_t$
2: Construir $G_t^{R,W}$ e obter $\texttt{RND}(G_1^W), ..., \texttt{RND}(G_t^W)$ usando `T-RND` com **pool.map_async**
3: Obter $\overline{F}_{to}(x)$ e $\overline{F}_{per}(x)$ e $\overline{F}_{coAfrequency}(x)$ de $G_t^{R,W}$ usando **pandas dataframe**
4: Obter $\overline{x}_{to}|\overline{F}_{to}(\overline{x}_{to})$ e $\overline{x}_{per}|\overline{F}_{per}(\overline{x}_{per})$ e $\overline{x}_{coAfrequency}|\overline{F}_{coAfrequency}(\overline{x}_{coAfrequency})$ = p_{rnd} com **pool.apply_async**
5: **for all** arestas$(i,j) \in E_t$ **do**
6: ClassificaAresta(*per*, *to*, *coAfrequency*) *//Realizado de acordo com Tabela 1*

peso $G_t^{R,W}$ do grafo temporal agregado G_t^W, que é necessário para analisar os padrões da rede. O grafo aleatório deve ter características topológicas similares ao grafo G_T^W (número de nós, arestas, e distribuição do grau), os nós são conectados de modo diferente a partir de G_T^W e o peso (frequência de co-autoria) são distribuídos aleatoriamente através das arestas. É importante ressaltar que a frequência de co-autoria de cada aresta em G_t^W é a soma de todas frequências de co-autoria em cada período de tempo.

STACY classifica as arestas em oito classes diferentes: sete sociais e uma aleatória. Essas classes de relacionamento são descritas na Tabela 1. Uma propriedade da rede com valor igual a "social" indica uma probabilidade quase zero do valor ser produzido de forma aleatória. Por outro lado, um valor da propriedade da rede é denominada "random" se há uma alta probabilidade do valor ser produzido de forma aleatória. Note que *strong* define relacionamentos fortes já que todos os valores das propriedades são "social", enquanto que *random* representa um relacionamento completamente aleatório. Além disso, as classes *bridge+* e *bridge* denotam pontes, isto é, persistem ao longo do tempo, mas têm um pequeno número de vizinhos comuns. A classe *bridge+* representa pontes com uma alta frequência de co-autoria e *bridge* com uma pequena. Ademais, a classe *transient* denota uma relação que acontece com alta intensidade (alta frequência de co-autoria e alta sobreposição de vizinhos), mas apenas em um momento específico. Já *periodic* representa um relacionamento que persiste ao longo do tempo e têm um elevado número de vizinhos comuns, mas pequena frequência de co-autoria (por exemplo, uma co-autoria entre colegas de um mesmo departamento que acontece uma vez no ano). *bursty* define uma relação com alta frequência de co-autoria, mas não persiste e não compartilha muitos vizinhos. Esta relação tende a ser isolada na rede. Por fim, *weak* representa relacionamentos fracos, pois não persistem ao longo do tempo e a frequência de co-autoria é pequena.

Como o RECAST, o único parâmetro do *STACY* é o p_{rnd} (Seção 3), que determina

Tabela 2. Informações principais sobre os conjuntos de dados.

Conjunto de dados	№ de nós	№ de arestas	Período
DBLP Articles	837,583	2,935,590	2000 to 2015
DBLP Inproceedings	945,297	3,760,247	2000 to 2015
PubMed	443,784	5,550,294	2000 to 2016
APS	180,718	821,870	2000 to 2013

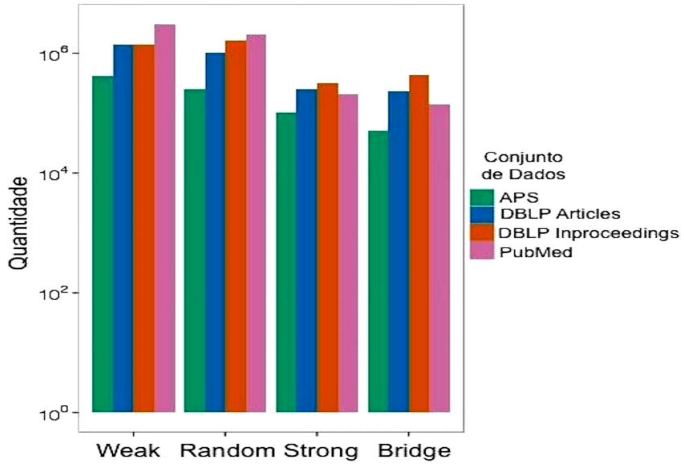


Figura 1. Pares de autores por classe gerada pelo fast-RECAST

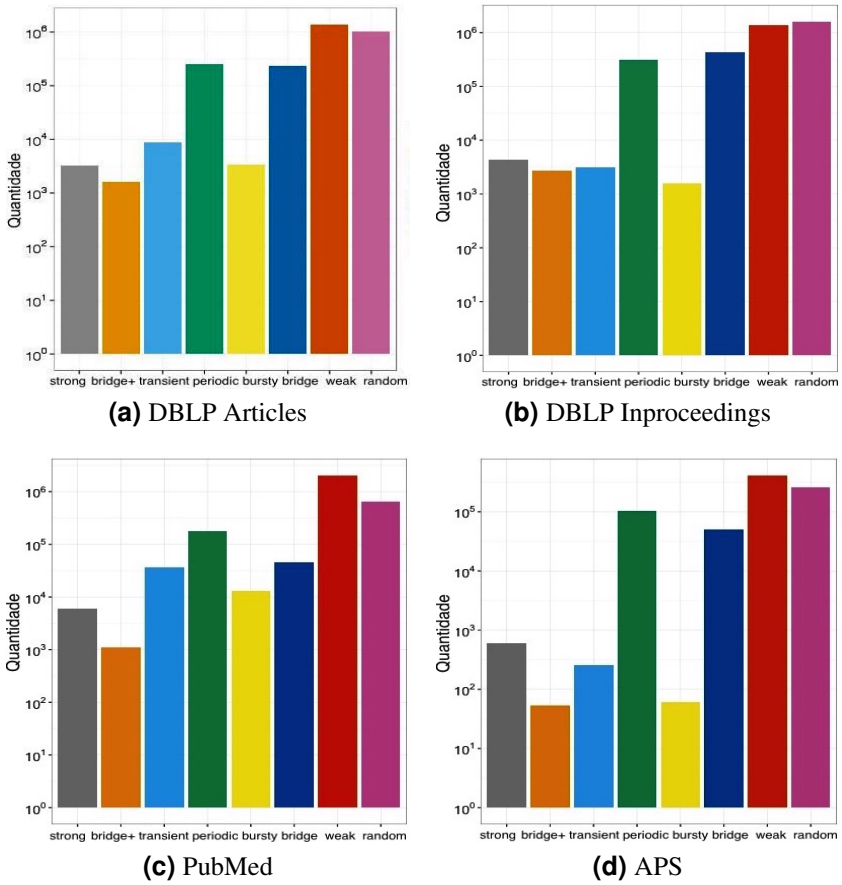
quando o valor de propriedade da rede é social ou aleatória. O Algoritmo 2 apresenta a classificação do *STACY*, paralelizada como o fast-RECAST.

5. Experimentos e Resultados

Utilizamos três conjuntos de dados para avaliar *STACY* e fast-RECAST: DBLP[3], PubMed[4] e APS[5]. A partir deles, construímos quatro redes sociais temporais cujas principais estatísticas estão na Tabela 2. Note que DBLP Inproceedings e DBLP Articles possuem a maior quantidade de nós (pesquisadores), mas PubMed tem a maior quantidade de arestas. A seguir, são detalhados três conjuntos de experimentos: aplicamos fast-RECAST e *STACY* nas redes para calcular a força dos laços; e dividimos as redes em duas janelas temporais para analisar a persistência e a transformação dos relacionamentos.

Fast-RECAST versus *STACY*. RECAST foi usado para classificar interações em redes móveis (Seção 3), mas os padrões sociais e suas características são diferentes das de redes de co-autoria. Neste trabalho, verificamos se o algoritmo é capaz de identificar os diferentes tipos de relacionamentos entre os co-autores. Também fizemos a mesma verificação para o *STACY*. Inicialmente, precisamos configurar o valor do parâmetro para p_{rnd}. Vaz de Melo et al. [2015] variou p_{rnd} em quatro ordens de magnitude e observou que o número de arestas por classe mantém-se na mesma magnitude. Portanto, o algoritmo não precisa de uma definição muito precisa do parâmetro para classificar as arestas de forma consistente. Executamos fast-RECAST e *STACY* para $p_{rnd} = 0.01$ e $p_{rnd} = 0$; como obtivemos resultados similares, apresentamos resultados apenas para $p_{rnd} = 0$.

[3]DBLP: `http://dblp.uni-trier.de`
[4]PubMed: `http://www.ncbi.nlm.nih.gov`
[5]APS: `http://www.aps.org`



(a) DBLP Articles **(b)** DBLP Inproceedings

(c) PubMed **(d)** APS

Figura 2. Pares de autores por classe gerada pelo *STACY*.

A Figura 1 apresenta a classificação das co-autorias em cada classe gerada por fast-RECAST para as quatro redes sociais. Em DBLP Articles, PubMed e APS, a maioria das co-autorias são classificadas como relacionamentos fracos, isto é, arestas com baixo (ou *aleatório*) sobreposição de vizinhos e persistência das arestas. Em tais redes, a maioria das co-autorias são mais classificadas como relacionamentos fortes (e não pontes). A exceção é DBLP Inproceedings, em que a maioria das arestas são mais atribuídas à classe aleatória e mais co-autorias são pontes ao invés de laços fracos. Uma possível explicação é que DBLP Articles, PubMed e APS possuem publicações em periódicos, enquanto que DBLP Inproceedings apenas em conferências. Conforme estudos recentes (e.g., [Lima et al. 2013]), Computação possui um comportamento incomum, no qual conferências são para publicação de ideias inovadoras e periódicos para fins de arquivamento. Desse modo, as redes de co-autoras de periódicos incluem autores que já publicaram juntos, apresentando assim relacionamentos mais fortes.

A Figura 2 tem a classificação da *STACY* para cada rede de coautoria. Similar ao fast-RECAST, a maioria dos relacionamentos são classificados como *weak* e *random*. Além disso, muitos relacionamentos são classificados como *periodic* e *bridge*. A grande quantidade de *periodic* revela que os pesquisadores da mesma comunidade (por exemplo, equipe, departamento, etc) tendem a publicar juntos com pequena frequência em um ano. Igualmente, a grande quantidade de *bridge* indica que a maioria das pontes tendem a ter uma pequena frequência de co-autoria em cada ano. Note que menos relacionamentos

Tabela 3. fast-RECAST versus *STACY* em divisão 80/20%

fast-RECAST	DBLP Articles		DBLP Inproceedings		PubMed		APS	
	80%	**20%**	**80%**	**20%**	**80%**	**20%**	**80%**	**20%**
Strong	75,128	16,083 (0.21)	136,159	19,608 (0.14)	91,143	19,555 (0.21)	45,020	30,046 (0.67)
Bridge	133,071	28,090 (0.21)	368,177	55,327 (0.15)	50,903	11,239 (0.22)	50,464	31,767 (0.63)
Weak	767,143	28,683 (0.04)	750,837	16,244 (0.02)	1,790,986	67,752 (0.04)	201,978	102,108 (0.51)
Random	931,796	76,298 (0.08)	1,340,167	69,661 (0.05)	1,021,710	63,986 (0.06)	249,711	128,479 (0.51)
	DBLP Articles		**DBLP Inproceedings**		**PubMed**		**APS**	
STACY	**80%**	**20%**	**80%**	**20%**	**80%**	**20%**	**80%**	**20%**
Strong	1,238	485 (0.39)	2,562	674 (0.26)	6,003	2,230 (0.37)	93	17 (0.18)
Bridge+	886	368 (0.41)	2,498	573 (0.23)	1,113	305 (0.27)	8	2 (0.25)
Transient	0	0	0	0	37,157	2,771 (0.07)	120	95 (0.79)
Periodic	1,070,400	64,249 (0.06)	1,149,339	53,445 (0.05)	175,179	34,215 (0.2)	58,663	12,122 (0.21)
Bursty	0	0	0	0	12,862	1,372 (0.1)	4	3 (0.75)
Bridge	834,614	84,052 (0.1)	1,440,941	106,148 (0.07)	45,419	8,718 (0.19)	36,720	6,840 (0.19)
Weak	0	0	0	0	2,042,114	76,552 (0.04)	256,564	13,908 (0.05)
Random	0	0	0	0	634,895	36,369 (0.05)	195,001	15,573 (0.08)

80% representa o passado (2000-2012 para DBLP articles e inproceedings, 2000-2013 PubMed e 2000-2010 APS) e 20% é o presente (2013-2015 para DBLP articles e inproceedings, 2014-2016 PubMed e 2011-2013 APS).

são classificados como *strong*, *bridge+*, *transient* e *bursty*. Essas quatro classes têm em comum o valor "social" para a propriedade frequência de co-autoria na rede social (as outras quatro classes têm um valor "random" para esta propriedade). Tal resultado mostra que a frequência de co-autoria é uma característica importante para medir a força dos relacionamentos, uma vez que ajuda a diferenciar as classes.

Análise da Persistência dos Relacionamentos. O objetivo aqui é investigar se os relacionamentos caracterizados com um determinado nível de força são prováveis de persistir. Para tal: (*i*) dividimos as redes em duas janelas temporais, que chamamos de *passado* e *presente*[6]; (*ii*) aplicamos fast-RECAST e *STACY* no passado; e (*iii*) verificamos se as arestas em cada classe mantêm-se na mesma classe no presente. As redes sociais foram divididas em duas janelas temporais de duas formas: uma janela temporal com 80% do timestamp inicial (*passado*) e outra com 20% do timestamp final (*presente*); e a mesma divisão para 70% (*passado*) e 30% (*presente*). Com resultados similares, a Tabela 3 apresenta apenas para a partição 80% e 20%. Os valores na coluna 80% são o número absoluto de arestas dos 80% dos anos das publicações atribuídos a cada classe. Os valores na coluna 20% são o número de arestas do passado que também estão no presente (proporções entre parênteses). Observamos que relacionamentos fortes e pontes tendem a persistir ao longo dos anos mais do que laços fracos e aleatórios.

Considerando os resultados do fast-RECAST, enfatizamos as diferenças nos resultados da rede APS nas partições 80% - 20% (e 70% - 30%). Na primeira divisão, a proporção de relacionamentos fortes e pontes do passado para o presente é muito alta, enquanto que na segunda divisão essa proporção é menor. Esse resultado pode indicar que a rede social de co-autoria da APS muda mais ao longo dos anos do que as outras redes. Outra possibilidade é que os pesquisadores da Física não alteram muito o nível de co-autoria com seus colaboradores ao longo do tempo, e esse é um padrão de pesquisadores mais recentes (observe que 80% dos dados consideram co-autoria mais recentes do que 70%). Deixamos tais análises para etapas futuras do trabalho.

[6]Pode-se ver o presente como o *timestamp* das janelas temporais.

Tabela 4. fast-RECAST e transformação dos relacionamentos: (a) DBLP Articles, (b) DBLP Inproceedings, (c) PubMed, (d) APS

(a)	Strong	Bridge	Weak	Random	Disappear
Strong	43,711 (0.11)	27,134 (0.07)	0	0	312,765 (0.82)
Bridge	14,650 (0.04)	13,874 (0.035)	0	0	361,041 (0.925)
Weak	0	0	0	0	0
Random	0	0	0	0	0
(b)	**Strong**	**Bridge**	**Weak**	**Random**	**Disappear**
Strong	34,761 (0.08)	26,411 (0.06)	0	0	351,935 (0.86)
Bridge	13,601 (0.02)	16,298 (0.024)	0	0	659,608 (0.96)
Weak	0	0	0	0	0
Random	0	0	0	0	0
(c)	**Strong**	**Bridge**	**Weak**	**Random**	**Disappear**
Strong	349 (0.02)	387 (0.02)	3,267 (0.16)	2,664 (0.13)	17,044 (0.67)
Bridge	66 (0.01)	97 (0.01)	659 (0.07)	667 (0.07)	8,643 (0.84)
Weak	10,532 (0.02)	10,425 (0.02)	94,800 (0.18)	73,039 (0.13)	346,559 (0.65)
Random	1,476 (0.01)	1,792 (0.01)	13,105 (0.06)	11,941 (0.05)	195,803 (0.87)
	Strong	**Bridge**	**Weak**	**Random**	**Disappear**
Strong	836 (0.03)	571 (0.02)	2,219 (0.09)	1,691 (0.06)	19,625 (0.8)
Bridge	450 (0.02)	421 (0.02)	918 (0.04)	910 (0.04)	19,173 (0.88)
Weak	4,013 (0.03)	2,071 (0.02)	14,185 (0.11)	7,154 (0.06)	99,844 (0.78)
Random	1,561 (0.013)	1,158 (0.01)	4,072 (0.03)	3,625 (0.03)	107,452 (0.92)

Em relação ao *STACY*, os relacionamentos fortes tendem a persistir mais que os outros em DBLP Articles, DBLP Inproceedings e PubMed nas partições 80% - 20% (e 70% - 30%). *STACY* é capaz de classificar melhor (do que o fast-RECAST) os relacionamentos fortes que persistem ao longo do tempo: um aumento de 0,18 para DBLP Articles, 0,12 para DBLP Inproceedings e 0,16 para PubMed na partição 80% - 20% (para 70% - 30%, o crescimento é ainda maior: 0,22 para DBLP Articles, 0,16 para DBLP Inproceedings e 0,22 para PubMed). A exceção é APS, em que a maioria dos relacionamentos em *transient* e *bursty* tendem a persistir ao longo do tempo. Este é um resultado inesperado, pois ambas as classes têm um valor "random" para a propriedade persistência das arestas. Analisando melhor a causa, observamos que as co-autorias em tais classes ocorrem de 2009 a 2013, ou seja, nos últimos anos das partições (os 80% incluem 2009 e 2010, e os 70% incluem 2009). Assim, o valor de persistência das arestas é pequeno, porque as co-autorias ocorrem nos anos incluídos no presente. Além disso, nenhuma aresta é classificada como *transient*, *bursty*, *weak* e *random* em DBLP Articles e DBLP Inproceedings em ambas as partições. Isso revela que, em tais redes, as co-autoria transitórias, *bursty*, fracas e aleatórias são relações recentes, porque elas são encontradas na versão completa dessas redes sociais (como mostra a Figura 2). Ademais, laços fracos e aleatórios são os que menos persistem ao longo do tempo na PubMed e APS.

Análise da Transformação das Classes dos Relacionamentos. Agora, avaliamos a quantidade de relacionamentos de uma classe no passado que continua na mesma classe (ou muda) no presente. Para análise mais uniforme, dividimos as redes em duas janelas temporais de 50% do timestamp, aplicamos fast-RECAST e *STACY* em ambas, e analisamos a transformação dos relacionamentos por meio das classes. A Tabela 4 mostra os resultados para fast-RECAST e a Tabela 5 para *STACY*. Os valores em cada coluna representam a quantidade e a proporção (entre parênteses) de laços do passado que persistem ou mudam de classe no presente (e.g., valores 43,711 e 0,11 na Tabela 4 são o número e a proporção de pares *strong* no passado que permanecem *strong* no presente).

Analisando os resultados de fast-RECAST, não há relacionamentos classificados como *weak* e *random* em DBLP Articles e DBLP Inproceedings na Tabela 4 partes (a) e (b). Isso indica que as propriedades (persistência das arestas e sobreposição de vizinhos) dessas redes têm valores altos (ou *social*). Além disso, a maioria dos relacionamentos do passado tendem a desaparecer no presente, especialmente *bridges*. Esse resultado pode

Tabela 5. *STACY* e transformação dos relacionamentos: (a) DBLP Articles, (b) DBLP Inproceedings, (c) PubMed, (d) APS

(a)	Strong	Bridge+	Transient	Periodic	Bursty	Bridge	Weak	Random	Disappear
Strong	0	1 (0.002)	0	54 (0.09)	0	19 (0.03)	0	0	549 (0.88)
Bridge+	0	0	0	8 (0.03)	0	9 (0.03)	0	0	238 (0.93)
Transient	0	0	0	0	0	0	0	0	0
Periodic	58 (1e-04)	7 (1.39e-05)	0	59,823 (0.12)	0	19,568 (0.04)	0	0	423,247 (0.84)
Bursty	0	0	0	0	0	0	0	0	0
Bridge	24 (8.9e-05)	4 (1.5e-05)	0	13,465 (0.05)	0	6,329 (0.02)	0	0	249,772 (0.92)
Weak	0	0	0	0	0	0	0	0	0
Strong	0	0	0	0	0	0	0	0	0
(b)	**Strong**	**Bridge+**	**Transient**	**Periodic**	**Bursty**	**Bridge**	**Weak**	**Random**	**Disappear**
Strong	0	0	0	28 (0.06)	0	21 (0.05)	0	0	387 (0.88)
Bridge+	0	0	0	21 (0.03)	0	7 (0.01)	0	0	596 (0.96)
Transient	0	0	0	0	0	0	0	0	0
Periodic	28 (6.79e-05)	5 (1.2e-05)	0	44,665 (0.1)	0	16,425 (0.04)	0	0	351,548 (0.85)
Bursty	0	0	0	0	0	0	0	0	0
Bridge	26 (3.8e-05)	6 (8.7e-06)	0	19,148 (0.03)	0	10,691 (0.02)	0	0	659,012 (0.95)
Weak	0	0	0	0	0	0	0	0	0
Random	0	0	0	0	0	0	0	0	0
(c)	**Strong**	**Bridge+**	**Transient**	**Periodic**	**Bursty**	**Bridge**	**Weak**	**Random**	**Disappear**
Transient	0	0	0	91 (0.14)	0	74 (0.12)	0	0	478 (0.74)
Bridge+	0	0	0	4 (0.05)	0	3 (0.03)	0	0	75 (0.91)
Transient	0	0	0	344 (0.19)	0	106 (0.06)	0	0	1348 (0.74)
Periodic	0	1 (4.1e-05)	0	4,780 (0.2)	0	2,440 (0.1)	0	0	17,192 (0.7)
Bursty	0	0	0	27 (0.05)	0	18 (0.03)	0	0	494 (0.9)
Bridge	0	0	0	473 (0.09)	0	290 (0.05)	0	0	4,675 (0.86)
Weak	35 (5.7e-05)	7 (1.1e-05)	0	137,563 (0.22)	0	62,939 (0.1)	0	0	416,963 (0.67)
Random	1 (7.2e-06)	0	0	10,216 (0.07)	0	5,854 (0.04)	0	0	123,557 (0.88)
(d)	**Strong**	**Bridge+**	**Transient**	**Periodic**	**Bursty**	**Bridge**	**Weak**	**Random**	**Disappear**
Strong	0	0	0	0	0	2 (0.3)	0	0	5 (0.7)
Bridge+	0	0	0	0	0	0	0	0	3 (1.0)
Transient	0	0	0	0	0	0	0	0	0
Periodic	0	0	0	836 (0.03)	0	569 (0.02)	2,219 (0.09)	1,691 (0.07)	19,620 (0.8)
Bursty	0	0	0	0	0	0	1 (1.0)	0	0
Bridge	0	0	0	450 (0.02)	0	421 (0.02)	918 (0.04)	910 (0.04)	19,170 (0.9)
Weak	11 (1e-04)	2 (1e-05)	0	4,002 (0.03)	0	2,069 (0.02)	14,185 (0.11)	7,154 (0.06)	99,844 (0.8)
Random	4 (3e-05)	2 (1e-05)	0	1,557 (0.01)	1 (8e-06)	1,156 (0.01)	4,071 (0.03)	3,624 (0.03)	107,452 (0.9)

ser explicado pela natureza das co-autorias, pois os pesquisadores colaboram durante um período por um objetivo comum e, em seguida, começam a colaborar com outros. Na Tabela 4, as partes (c) e (d) mostram comportamento semelhante para PubMed e APS, e a maioria dos relacionamentos tendem a desaparecer, especialmente pontes e aleatórios. Desconsiderando os relacionamentos que desaparecem, relacionamentos mais fortes e fracos tornam-se fracos ou aleatórios. Surpreendentemente, relacionamentos fracos são os que mais se mantêm na mesma classe, comparando com os outros em ambas as redes.

Para *STACY*, também não há relacionamentos *weak* e *random* em DBLP Articles e DBLP Inproceedings na Tabela 5 partes (a) e (b). Assim, a frequência de co-autoria dessas redes também possuem um valor alto (ou *social*). Os relacionamentos também não são classificados como transitórios em DBLP Articles, DBLP Inproceedings e APS (Tabela 5 parte (d)), o que revela a ausência dessas co-autorias em períodos anteriores nessas redes. Ademais, DBLP Articles e DBLP Inproceedings não têm relacionamentos *bursty*, o que indica que os relacionamentos com alta frequência de co-autoria também compartilham um grande número de vizinhos nessas redes no período coberto pelos 50% de dados (isso também é confirmado pela presença de relacionamentos em *transient*). Como fast-RECAST, a maioria dos relacionamentos também tende a desaparecer quando classificados por *STACY*. A diferença é que usando *STACY*, observamos que os relacionamentos de diferentes classes tendem a mudar para *periodic* e *bridge* ao longo do tempo, especialmente, em DBLP Articles, DBLP Inproceedings e PubMed (Tabela 5 parte (c)).

6. Conclusões

Neste artigo, propusemos o *STACY*, um algoritmo paralelo e rápido que classifica os relacionamentos de coautoria em oito classes. Também detalhamos o fast-RECAST, uma

versão paralela e mais rápida do RECAST que gera quatro classes. Tais algoritmos são úteis por serem capazes de automaticamente detectar classes de relacionamentos em redes sociais, além de diferenciar relacionamentos sociais dos aleatórios. Essas classes podem auxiliar a entender o processo de difusão de informação, melhorar algoritmos que detectam comunidades e recomendam pessoas. De fato, ao agrupar as arestas em classes utilizando redes sociais reais, analisamos o dinamismo da força dos relacionamentos ao longo do tempo. A análise da persistência revela que relacionamentos fortes e pontes tendem a persistir ao longo dos anos mais que outros tipos, confirmando nossa hipótese de que os laços fortes persistem mais. Além disso, *STACY* foi capaz de encontrar laços fortes que persistem mais que os encontrados por fast-RECAST. A análise de transformação dos relacionamentos revelou que a maioria dos laços tendem a desaparecer ao longo do tempo. Isso pode ocorrer devido à natureza das co-autorias, por exemplo, os pesquisadores tendem a publicar com alunos durante um período e quando os alunos se formam, o processo de publicação em conjunto é finalizado. Como trabalhos futuros, planejamos aplicar *STACY* em outros tipos de redes sociais, por exemplo, no GitHub que revela interações entre desenvolvedores durante o processo de desenvolvimento de softwares (estendendo [Alves et al. 2016]). Também pretendemos adicionar propriedades ao *STACY* para diferenciar relacionamentos recentes dos antigos enquanto mede a força.

Agradecimentos. Trabalho parcialmente financiado por CAPES, CNPq e FAPEMIG.

Referências

Alves, G. B., Brandão, M. A., Santana, D. M., da Silva, A. P. C., and Moro, M. M. (2016). The strength of social coding collaboration on github. In *SBBD*, pages 247–252, Salvador, Brazil.

Brandão, M. A. and Moro, M. M. (2015). Analyzing the strength of co-authorship ties with neighborhood overlap. In *DEXA*, pages 527–542, Linz, Austria.

Castilho, D., Vaz de Melo, P. O. S., and Benevenuto, F. (2017). The strength of the work ties. *Information Sciences*, 375:155–170.

Gilbert, E. and Karahalios, K. (2009). Predicting tie strength with social media. In *SIGCHI*, pages 211–220, Boston, USA.

Granovetter, M. S. (1973). The strength of weak ties. *American Joumal af Sociology*, 78(6):1360–1380.

Karsai, M., Perra, N., and Vespignani, A. (2014). Time varying networks and the weakness of strong ties. *Scientific reports*, 4.

Kostakos, V. (2009). Temporal graphs. *Physica A: Statistical Mechanics and its Applications*, 388(6):1007–1023.

Laurent, G., Saramäki, J., and Karsai, M. (2015). From calls to communities: a model for time-varying social networks. *The European Physical Journal B*, 88(11):1–10.

Lima, H., Silva, T. H., Moro, M. M., Santos, R. L., Meira Jr, W., and Laender, A. H. (2013). Aggregating productivity indices for ranking researchers across multiple areas. In *JCDL*, pages 97–106, NYC, USA.

Miller, J. C. and Hagberg, A. (2011). Efficient generation of networks with given expected degrees. In *WAW*, pages 115–126, Atlanta, USA.

Nicosia, V. et al. (2013). *Temporal Networks*, chapter Graph Metrics for Temporal Networks, pages 15–40. Springer Berlin Heidelberg, Berlin, Heidelberg.

Vaz de Melo, P. O. S. et al. (2015). Recast: Telling apart social and random relationships in dynamic networks. *Performance Evaluation*, 87:19–36.

Zaki, M. J. and Meira Jr, W. (2014). *Data mining and analysis: Fundamental concepts and algorithms*. Cambridge University Press.

Zignani, M., Gaito, S., and Rossi, G. P. (2016). Predicting the link strength of newborn links. In *WWW*, pages 147–148, Montreal, Canada.

paper:171428

Um *Survey* sobre Soluções para Publicação de Dados na Web sob a Perspectiva das Boas Práticas do W3C

Lairson Emanuel R. de Alencar Oliveira[1], Marcelo Iury S. Oliveira[1,2], Bernadette Farias Lóscio[1]

[1]Centro de Informática -– Universidade Federal de Pernambuco (UFPE)
Caixa Postal 50.740–560 -– Recife -– PE -– Brasil

[2]Unidade Acadêmica de Serra Talhada – Universidade Federal Rural de Pernambuco

{lerao, bfl, miso}@cin.ufpe.br, marcelo.iury@ufrpe.br

Abstract. *The growing interest in using the Web as platform for data sharing has motivated the development of data publishing solutions. These solutions support various tasks related to the Data on the Web lifecycle. However, since there are plenty of solutions, selecting the most suitable for a specific usage scenario is not a trivial task. This task becomes more complex given that significant subset of the most commonly used solutions are not specified or evaluated in scientific studies. In this work, it is presented a survey and analysis of Data on the Web publishing solutions with regards to the Data on The Web Best Practices recommended by the W3C.*

Resumo. *O crescente interesse no uso da Web para compartilhamento de dados tem motivado o desenvolvimento de soluções para publicação de dados. Essas soluções auxiliam em várias tarefas relacionadas ao ciclo de vida dos dados na Web. Entretanto, como há uma abundância de soluções, selecionar a mais apropriada para um cenário específico não é uma tarefa trivial. Esse trabalho se torna ainda mais complexo pelo fato de que uma parcela significativa das soluções mais usadas não estão especificadas ou avaliadas em trabalhos científicos. Neste artigo, é realizado um survey das soluções de publicação de dados na Web, apresentando um comparativo sob o ponto de vista do conjunto de Boas Práticas para Dados na Web recomendado pelo W3C.*

1. Introdução

Uma série de iniciativas vem sendo desenvolvidas em todo o mundo objetivando o compartilhamento e consumo dos dados na Web, dentre elas destacam-se a publicação de dados abertos (*Open Data*) [Dietrich et al. 2009] e de dados conectados (*Linked Data*) [Isotani and Bittencourt 2015]. Em conjunto com o crescente interesse nessas iniciativas, tem aumentado o interesse no desenvolvimento de soluções para publicação de dados a fim de facilitar tanto a publicação quanto o acesso, o consumo e a manutenção dos dados publicados [Lóscio et al. 2015].

A proposição de soluções para publicação de dados na Web tem sido alvo de vários estudos reportados na literatura [Milić et al. 2015, Goldacre and Gray 2016, Hoxha and Brahaj 2011], entretanto muitos trabalhos ainda estão em fase conceitual com poucas validações práticas. Em contraposição, também existem diversas soluções prontas

para uso e de propósito genérico, tais como CKAN, Socrata, OGDI, Junar e OpenDataSoft, as quais são aplicadas em diversas iniciativas de dados abertos e portais de dados abertos governamentais.

Essas soluções são bastante heterogêneas quanto a diversos aspectos relacionados à publicação e ao consumo de dados, tais como formatos de dados adotados para a publicação (*e.g.*, XML, JSON, CSV), o uso de APIs (do inglês, *Application Programming Interface*) para o acesso aos dados e o uso de vocabulários para descrição dos dados e metadados (*e.g.*, DCAT[1]). Como há uma abundância de soluções, selecionar qual a solução mais apropriada para um cenário especifico não é uma tarefa trivial.

Uma alternativa seria medir a qualidade das soluções. Entretanto, determinar a qualidade e a eficácia de uma solução também não é uma tarefa trivial devido aos diversos aspectos técnicos e humanos envolvidos na sua utilização. Diante dessa problemática, diversas áreas fazem uso de especificações ou recomendações técnicas com o objetivo tanto de guiar processos de avaliação de qualidade quanto reduzir os riscos inerentes em avaliações subjetivas. Exemplos clássicos de especificações de qualidade são os padrões da família ISO 9000[2] que tem sido usado em diversos casos desde avaliação de software até processos de negócio.

Um conjunto de critérios promissor para análise e avaliação das soluções de publicação de dados na Web é a recomendação do W3C intitulada *Data on the Web Best Practices* (DWBP) [Lóscio et al. 2016a]. Essa recomendação propõe um conjunto de boas práticas a serem implementadas pelos produtores de dados com o intuito de produzir conjuntos de dados que sejam úteis e atendam às expectativas dos consumidores de dados [Lóscio et al. 2016a]. No total, são 35 boas práticas que abordam diferentes aspectos relacionados à publicação e ao consumo de dados, tais como formatos de dados, acesso a dados, metadados e uso de identificadores. Ao seguir essas boas práticas, espera-se que uma série de benefícios distintos possam ser alcançados, tais como a compreensão, processabilidade, descoberta, reúso, acesso e interoperabilidade de dados [Lóscio et al. 2016a]. De modo geral, a recomendação DWBP não considera abordagens ou tecnologias específicas para publicação de dados, assim ele pode ser utilizado como modelo de referência para a medição de qualidade de soluções de publicação e consumo de dados na Web.

Diante desse cenário, neste artigo é apresentado um *survey* comparativo de soluções de publicação de dados na Web. Para guiar a realização do *survey*, foi utilizada a seguinte questão: *Quais são as soluções mais utilizadas atualmente para a publicação de dados na Web e como elas estão alinhadas com as Boas Práticas recomendadas pelo W3C?*. Além disso, foram seguidas quatro etapas bem definidas:(i) definição dos critérios de avaliação, (ii) seleção das soluções, (iii) extração dos dados e (iv) análise. Os critérios de avaliação são representados pelas boas práticas e, de maneira geral, busca-se avaliar se a solução de publicação atende, atende parcialmente ou não atende a respectiva boa prática. Para o *survey*, foram selecionadas sete soluções de publicação usadas em uma amostra representativa de portais de publicação de dados na Web.

Os resultados da análise demonstraram que existem boas práticas que são cobertas plenamente ou parcialmente por todas as soluções, como aquelas relacionadas aos metada-

[1]https://www.w3.org/TR/vocab-dcat/

[2]https://www.iso.org/standard/45481.html

dos, informações de proveniência, reúso de vocabulários, uso de representações complementares para os dados e na disponibilização de diferentes mecanismos de acesso como API e *bulk download*. No entanto, algumas boas práticas ainda são pouco ou não são atendidas pelas soluções, como aquelas relacionadas ao versionamento, atualização, qualidade e preservação dos dados.

O restante deste artigo está estruturado como se segue. Na Seção 2, são apresentados os trabalhos relacionados. Na Seção 3, é descrito o método de pesquisa adotado e as soluções encontradas. Na Seção 4, é apresentada a análise das soluções. Por fim, na Seção 5, é apresentada uma pequena discussão sobre o trabalho realizado e, na Seção 6, são apresentadas as conclusões e sugestões para os trabalhos futuros.

2. Trabalhos Relacionados

Dados na Web é um campo de pesquisa que tem sido alvo de muitos estudos reportados na literatura. No que diz respeito à avaliação e *surveys* de soluções para publicação de dados na Web, há trabalhos que fazem um levantamento de algumas soluções, apresentando suas características gerais.

Os estudos mais exaustivos foram realizados por [Stråle and Lindén 2014], [Lisowska 2016] e [Millette and Hosein 2016]. Apesar desses trabalhos serem os mais similares ao nosso em termos de representatividade de amostra de soluções e em relação à interseção de alguns critérios, como os metadados e aspectos técnicos, eles diferem em relação ao foco.

Stråle e Lindén (2014) realizaram uma avaliação comparativa das principais soluções existentes em relação a uma solução proprietária utilizada pelo governo da Suécia. Em outras palavras, levantaram as principais características do CKAN, Socrata e OpenDataSoft com o intuito de verificar quais aspectos a solução usada pelo governo Sueco poderia melhorar ou, alternativamente, substituir em sua solução. Millette e Hosein (2016) seguiram a mesma abordagem ao apresentar um levantamento de características do CKAN, Socrata e Junar e as compararam com uma nova solução conceitual com o foco no consumo de dados. Assim, eles identificaram as características gerais das soluções, verificando a disponibilidade de ferramentas para transformação e enriquecimento de dados, além da disponibilização e criação de APIs.

Por outro lado, o trabalho [Lisowska 2016] analisou as soluções sob a perspectiva da interoperabilidade dos metadados. Nesse trabalho, foram analisados 55 portais de dados abertos gerados a partir de diferentes soluções de publicação de dados com o intuito de testar a interoperabilidade de metadados. Assim, os autores analisaram a interoperabilidade e correspondência entre os vocabulários de representação de metadados (*e.g.*, DCAT) usados por CKAN, Socrata, OpenDataSoft, Junar, DKAN e ArcGis Open Data.

Esses trabalhos destacam a importância e o surgimento de soluções de publicação de dados como campo de pesquisa. No entanto, eles se concentram na análise de características gerais ou analisam uma quantidade pequena de soluções, geralmente mais populares ou citadas em outros artigos. Além disso, nenhum deles cobre a quantidade de soluções avaliadas neste artigo, assim como não consideram aspectos diversos dispostos nas Boas Práticas para Dados na Web do W3C.

3. Método de Pesquisa

O principal objetivo deste trabalho é apresentar um *survey* comparativo das soluções de publicação de dados na Web. Em geral, *surveys* permitem o levantamento de conceitos, modelos, características e tendências emergentes que aparecem de forma desconexa e esparsa na literatura. Visto que a temática deste trabalho envolve o uso de soluções mercadológicas, estas podem não ser cobertas por mapeamentos e revisões sistemáticas da literatura. Então, *surveys* são essenciais, porque possibilitam a descoberta de trabalhos oriundos de diversos veículos, incluindo artigos científicos e literatura cinzenta, tais como sites, manuais, relatórios técnicos e *white papers*.

O presente *survey* foi realizado em quatro etapas bem definidas:(i) definição dos critérios de avaliação, (ii) seleção das soluções, (iii) extração dos dados e (iv) análise. Enquanto a análise das soluções selecionadas é apresentada na Seção 4, as demais etapas são apresentas nas subseções a seguir.

3.1. Critérios de Avaliação

Neste trabalho, os critérios de avaliação das soluções de publicação de dados na Web foram construídos essencialmente a partir da recomendação de Boas Práticas para Dados na Web do W3C. Dessa forma, os critérios de avaliação são representados por cada boa prática e as soluções são avaliadas para verificar se atendem, atendem parcialmente, ou não atendem a respectiva boa prática. Na Tabela 1, é apresentado o conjunto de boas práticas.

Seguindo as boas práticas, alguns benefícios podem ser alcançados mais facilmente e as chances de reutilização dos conjuntos de dados tendem a melhorar. Assim, será possível uma melhor compreensão sobre a estrutura e o significado dos dados, bem como as máquinas ou agentes de software poderão descobrir, processar e manipular automaticamente os dados. Somado a isso, será possível acessar dados atualizados em uma variedade de formas, será mais fácil chegar a um consenso entre os produtores e consumidores de dados e a confiança dos consumidores nos conjunto de dados tende a melhorar [Lóscio et al. 2016a].

Para a realização do *survey*, algumas boas práticas foram desconsideradas, pois não dizem respeito às soluções em si. Este foi o caso das boas práticas 10, 16, 22 e 28, que estão diretamente relacionadas aos dados ao invés das soluções. Também foram desconsideradas as Boas Práticas 31, 33, 34 e 35, que se aplicam aos casos de republicação de dados, dependendo fortemente do consumidor ao invés da solução.

3.2. Seleção das Soluções

A seleção das soluções mostrou-se uma tarefa complexa, pois o presente trabalho teve como desafio cobrir uma amostra representativa de soluções usadas na prática para publicação de dados na Web. Contudo, grande parte das soluções existentes não estão documentadas em artigos científicos, impossibilitando, dessa forma, fazer uma busca utilizando métodos tradicionais de pesquisa como o uso de base de dados científicas (*e.g.*, ACM, IEEE ou Scopus).

Ao invés de procurar diretamente pelas soluções disponíveis na Web, optou-se por identificar as soluções de publicação de dados que estão sendo utilizadas em portais de dados disponíveis na Web. Essa abordagem permite identificar quais são as soluções usa-

Tabela 1. Boas práticas para dados na Web. Fonte: [Lóscio et al. 2016a]

ID	Descrição
BP1	Prover metadados
BP2	Prover metadados que possibilitem interpretar a natureza dos conjuntos de dados e suas distribuições
BP3	Prover metadados que descrevem o esquema e a estrutura interna de uma distribuição
BP4	Prover um link ou cópia do contrato de licença que controle o uso dos dados
BP5	Prover informações de proveniência dos dados (informações completas sobre a origem dos dados e as alterações que realizadas nos mesmos).
BP6	Prover informações sobre qualidade e adequação de dados para fins específicos
BP7	Prover um indicador de versão, que atribua e indique um número ou data de versão para cada conjunto de dados
BP8	Prover um histórico de versões completo que explique as mudanças feitas em cada versão.
BP9	Empregar URI persistentes como identificadores de conjuntos de dados
BP10	Empregar URI persistentes como identificadores dentro de conjuntos de dados
BP11	Atribuir URIs a versões individuais de conjuntos de dados, bem como à série de versões como um todo
BP12	Disponibilizar dados em um formato de dados padronizado e legível por máquina que seja adequado ao seu uso potencial ou pretendido
BP13	Empregar representações de dados neutras que não sejam influenciadas por aspetos culturais ou relacionados a localidade, por exemplo a formatação de datas e números. Quando isso não for possível, forneça metadados sobre as representações e formatações usadas pelos valores de dados.
BP14	Disponibilizar dados em vários formatos quando mais de um formato se adéqua ao seu uso potencial ou pretendido
BP15	Reusar termos de vocabulários, de preferência padronizados, para codificar dados e metadados
BP16	Optar por um nível de semântica formal que possa encaixar os dados e as aplicações mais prováveis
BP17	Prover um serviço de *download* em massa *(bulk download)* para permitir que usuários recuperem múltiplos conjuntos de dados com uma única solicitação
BP18	Prover opções para efetuar operações e realizar *download* de subconjuntos dos dados
BP19	Usar negociação de conteúdo para permitir o *download* dos dados em vários formatos
BP20	Disponibilizar dados em tempo real ou próximo de tempo real, quando os mesmos são produzidos em tempo real
BP21	Prover dados atualizados conforme frequência de atualização do mundo real
BP22	Prover uma explicação do porquê os dados não estão indisponíveis, como os dados podem ser acessados e quem pode acessar
BP23	Disponibilizar dados através de uma API
BP24	Empregar os Padrões da Web como base das APIs, tais como REST and SOAP
BP25	Prover documentação completa para a API
BP26	Evitar alterações na API para impedir quebra de código para quem já as está usando
BP27	Preservar o identificador e fornecer informações sobre o recurso arquivado
BP28	Avaliar a cobertura de um conjunto de dados antes da seu arquivamento
BP29	Prover um meio facilmente descoberto para que usuários ofereçam *feedback* a respeito dos dados
BP30	Tornar *feedback* dos usuários a respeito dos conjuntos de dados e distribuições publicamente disponíveis
BP31	Enriquecer seus dados, gerando novos dados ao fazê-lo, aumentará seu valor
BP32	Prover representações complementares para os dados, como visualizações, tabelas, aplicativos Web ou sínteses estatísticas
BP33	Prover *feedback* para o produtor original dos dados
BP34	Seguir os requisitos da licença original do conjunto de dados quando os mesmos estão sendo re-publicados
BP35	Citar a publicação original em seus metadados

das na prática, bem como aquelas que possuem nível de maturidade suficiente para serem analisadas.

Dessa forma, o levantamento dos portais foi realizado usando como base os sites DataPortals.org[3] e Opendatasoft.com[4]. Ambos os sites apresentam um catálogo de portais de dados de diversos países do mundo. Esses catálogos são reconhecidos como os mais abrangentes em relação a quantidade de portais indexados[5]. No total, identificam mais de 3300 portais de dados.

Contudo, esses catálogos apresentam alguns problemas de inconsistência, tais como portais inativos e redundância entre os portais indexados. Além disso, há portais

[3]http://dataportals.org/ Acesso em 10/05/2017

[4]https://www.opendatasoft.com/a-comprehensive-list-of-all-open-data-portals-around-the-world/ Acesso em 10/05/2017

[5]http://okfnlabs.org/projects/dataportals-org/ Acesso em 10/05/2017

que utilizam soluções proprietárias que tornam difícil ou até impossível de serem avaliadas. Por esses motivos, foi realizada uma filtragem com base nos seguintes critérios de exclusão: (i) O portal está inativo; (ii) O portal é redundante (foi identificado previamente); (iii) O portal usa uma solução proprietária para publicação de dados; (iv) O portal é uma página de teste ou representa uma prova de conceito da solução; (v) O portal não faz uso de solução de publicação de dados. Aplicando os critérios de exclusão, restaram ainda 2400 portais para serem analisados.

A identificação da solução de publicação de dados foi realizada de forma manual, analisando individualmente cada portal, ou de forma automática na qual um *crawler* envia requisições HTTP para cada portal e analisa o cabeçalho das respostas recebidas em busca de *tokens* e palavras chaves que identifiquem a solução usada como plataforma de construção dos portais. Essa segunda abordagem, apesar de promissora, não possui uma boa precisão, pois não há uma API padronizada usada por todas as soluções, tornando, assim, difícil detectar soluções que ainda não são tão populares como CKAN ou Socrata.

Para aumentar a precisão da identificação de soluções, optou-se por uma abordagem mista, na qual parte dos portais foi analisada de forma automática e outra parte foi analisada manualmente. Entretanto, como o processo manual de checagem demanda tempo considerável para ser realizado, foi realizada uma amostragem aleatória dos portais para identificação manual das soluções de publicação adotadas pelos portais. Ao todo, foram checados 1091 portais (representando aproximadamente 45% da amostra total de portais).

Tabela 2. Soluções de Publicação de Dados na Web

Nome	**URL**	***Open Source?***	***Cloud computing?***
CKAN	http://ckan.org/	Sim	Não
Socrata	https://socrata.com/	Não	Sim
DKAN	http://getdkan.com/	Sim, mas tem uma versão paga	Não (gratuito) ou Sim (pago)
Junar	http://junar.com/	Não	Sim
ArcGIS Open Data	http://opendata.arcgis.com/	Não	Sim
OpenDataSoft	https://www.opendatasoft.com/	Não	Sim
Knoema	https://knoema.com/	Não	Sim

Após a aplicação das estratégia manual e automática de checagem, foram mapeadas 15 soluções. Entretanto, foram selecionadas 7 soluções que estão sendo usadas em pelo menos 10 portais atualmente. As soluções selecionadas são apresentadas na Tabela 2. É importante ressaltar que o propósito deste artigo não foi realizar uma análise exaustiva de todas as soluções existentes. Mas, identificar as principais soluções usadas na prática em portais de dados disponíveis na Web.

3.3. Extração dos Dados

Nesta fase, buscou-se obter informações das soluções de publicação para responder os critérios de avaliação previamente definidos. Os dados usados para avaliação das soluções foram coletados através de documentação oficial (*e.g.*, manuais, diagramas arquiteturais, *white papers*) e ambientes de demonstração providos pelas instituições responsáveis pelas soluções. A consolidação dos dados extraídos envolveu interpretação subjetiva. Contudo, com o intuito de mitigar o risco de má interpretação, os dados foram triangulados a partir de outras fontes, tais como sites e fóruns de usuários, vídeos, apresentações, páginas de divulgação da solução. Além disso, foram realizadas reuniões para se alcançar o consenso da interpretação dos dados.

4. Análise

Nesta seção, são apresentados os resultados da análise das soluções mais utilizadas para a publicação de dados na Web sob a perspectiva das Boas Práticas para Dados na Web. É importante ressaltar que, ao invés de considerar os conjuntos de dados ou portais de dados específicos como evidências, foram analisadas as soluções adotadas pelos portais para verificar se elas implementam as boas práticas. O resultado da análise é apresentado na Tabela 3 é discutido a seguir.

Tabela 3. Soluções para Publicação de Dados X Boas Práticas. Fonte: os autores

Boa Prática	CKAN	Socrata	DKAN	Junar	ArcGIS O. D.	OpenDataSoft	Knoema
BP1	A	A	A	A	A	A	A
BP2	A	A	A	A	A	A	A
BP3	AP	A	-	AP	A	-	AP
BP4	A	A	A	-	AP	AP	-
BP5	A	A	A	-	AP	AP	A
BP6	-	A	-	-	-	-	-
BP7	A	-	A	-	-	-	AP
BP8	-	AP	A	-	-	-	-
BP9	A	A	A	A	AP	A	AP
BP11	-	A	-	-	-	-	-
BP12	A	A	A	A	A	A	A
BP13	-	-	-	-	-	AP	A
BP14	-	A	-	A	A	A	A
BP15	A	A	A	A	AP	A	AP
BP17	A	A	A	A	A	A	A
BP18	A	A	A	A	A	-	A
BP19	-	A	-	A	A	A	-
BP20	-	-	-	-	A	-	-
BP21	-	AP	-	A	-	-	AP
BP23	AP	A	A	A	A	A	-
BP24	A	A	A	A	A	A	A
BP25	A	A	A	A	A	A	A
BP26	A	A	A	A	A	A	A
BP27	-	-	-	-	-	-	-
BP29	A	A	A	-	-	-	A
BP30	A	A	-	-	-	A	-
BP32	A	A	A	A	A	A	A

Legenda: A = Atende; AP = Atende Parcialmente; - = Não atende.

Os metadados ajudam os usuários a compreenderem o significado dos dados, sua estrutura, licença, instituição que produziu os dados, métodos de acesso e agendamento de futuras atualizações dos conjuntos de dados. Nesse contexto, todas as soluções fornecem metadados de uma forma geral (BP1), assim como fornecem metadados descritivos que ajudam na interpretação da natureza dos conjuntos de dados e suas distribuições (BP2). Com relação ao fornecimento de metadados estruturais (BP3), o Socrata e o ArcGIS Open Data atendem plenamente essa boa prática, sendo destacado o Socrata por fornecer mecanismos para interpretar automaticamente os metadados estruturais. Algumas soluções oferecem apenas uma cobertura parcial ou não disponibilizam os metadados igualmente em formatos legíveis por humanos e por máquinas. Dessa forma, o CKAN, o Knoema e o Junar implementam a BP3 parcialmente, pois não disponibilizam uma versão legível por humanos para os metadados estruturais, ou seja, não disponibilizam metadados que descrevam o esquema e a estrutura interna dos conjuntos de dados.

Dado que podem existir restrições quanto ao compartilhamento e reutilização dos dados, é importante informar a licença dos conjuntos de dados (BP4). Somado a isso, é

importante informar a origem dos dados (BP5), assim como informar metadados de qualidade (BP6). Em relação a disponibilização de informações acerca das licenças do conjunto de dados, apenas o Knoema não fornece esse tipo de informação. Já em relação a informações de proveniência, todas as soluções atendem parcialmente essa boa prática ao fornecer informações a respeito das entidades responsáveis pela produção e manutenção dos dados. No entanto, o ARCGIS e o OpenDataSoft implementam a BP4 e a BP5 de forma parcial por não oferecer uma maneira para representar os metadados de licença e proveniência legíveis por máquina. Finalmente, apenas o Socrata fornece informações a respeito da qualidade dos conjuntos de dados (BP6).

Como os conjuntos de dados também podem mudar ao longo do tempo, são consideradas boas práticas atribuir e indicar o número de versão ou data para cada conjunto de dados (BP7), assim como manter um histórico de todas as versões geradas (BP8). Apenas o CKAN, DKAN e Knoema apresentam metadados que indicam a versão do conjunto de dados. Enquanto o Knoema faz o registro das alterações em nível de dados, o CKAN e DKAN fazem o controle do conjunto de dados como um todo. Em relação a BP8, apenas o Socrata atende parcialmente essa boa prática ao permitir a listagem do histórico de versões, contudo essa funcionalidade não é acessível por API.

Identificadores são amplamente utilizados nos sistemas de informação com o intuito de se identificar sem ambiguidades e independente de localização um determinado recurso. São elencadas como boas práticas o uso de URIs persistentes como identificadores para conjuntos de dados ou para versões individuais (BP9) e para a série de versões como um todo (BP11). Todas as soluções fazem uso de URIs persistentes para identificação dos conjuntos de dados. No entanto, o Knoema atende parcialmente a BP9. Apesar de fazer uso de URIs para identificar os conjuntos de dados, no Knoema, as URIs são geradas através de funções hash[6] não permitindo, assim, que humanos possam reconhecer o conjunto de dados associado a um determinado conjunto de dados. No mais, somente o Socrata realiza a atribuição de URIs para as versões dos conjuntos de dados.

O formato no qual os conjuntos de dados são disponibilizados é fundamental para tornar os dados utilizáveis para os consumidores. Dessa forma, é considerada uma boa prática disponibilizar os dados em formato legível por máquina (BP12), como CSV, XML, RDF e outros. Também é aconselhado usar representações de dados que não sejam influenciadas por localidades ou, quando não for possível, fornecer metadados sobre a localidade ou idioma usado pelos conjuntos de dados (BP13). Todas as soluções permitem o uso de formatos legíveis por máquina para a distribuição dos conjuntos de dados. Por fim, apenas o OpenDataSoft e Knoema atendem a BP13, ainda que parcialmente, pois oferecem uma maneira de representar os metadados de localidade ou idioma para acesso legível por máquina.

Além disso, também constatou-se que as soluções fornecem opções para fornecimento de dados em diferentes formatos (BP14). É importante frisar que os formatos de dados podem depender do formato que o produtor deseja publicar, mas para análise foi considerado se a solução apresentava limitações quanto aos principais formatos usados nesse contexto.

Visando uma maior interoperabilidade e consenso entre os produtores de dados e

[6]https://en.wikipedia.org/wiki/Cryptographic$_{h}ash_{f}unction$

os consumidores, é considerado uma boa prática o reúso de vocabulários (BP15), dando preferência aos padronizados. Os vocabulários definem os conceitos e atributos utilizados para descrever e representar uma área de interesse, tais como o vocabulário DCAT que é utilizado para expressar os metadados relacionados aos conjuntos de dados. Praticamente todas as soluções fazem o uso de vocabulários compartilhados para padronização de metadados. Esses vocabulários tem sido utilizados especialmente para padronização de metadados descritivos dos conjuntos de dados. No mais, todas as soluções analisadas são compatíveis com o vocabulário DCAT, indicando que seus metadados são cobertos por ele.

Por padrão, a Web oferece acesso aos dados através de métodos HTTP, o que permite acesso em um nível de transação atômica. Com o objetivo de oferecer mais opções de acesso, é recomendado permitir o *download* em massa de múltiplos conjuntos de dados com uma única solicitação (BP17) e, em caso de conjuntos de dados grandes, pode-se oferecer opções por meio de APIs para efetuar operações e recuperar subconjuntos dos dados (BP18). Todas as soluções permitem o *download* em massa de múltiplos conjuntos. Já em relação a BP18, todas as soluções permitem acesso a subconjuntos dos dados por meio da API.

Ainda em relação à recuperação de conjuntos de dados, também é considerado uma boa prática o uso da negociação de conteúdo para permitir a recuperação dos conjuntos de dados em vários formatos (BP19). Por exemplo, a partir de uma mesma URI, usando a negociação de conteúdo, podem-se obter os dados em JSON, XML ou CSV. Exceto pelo Knoema, CKAN e DKAN, as demais ferramentas permitem recuperar os dados em diferentes formatos do que foi publicado, ou seja, fazem uso da negociação de conteúdo para pré-processamento interno ou para disponibilizar os dados em formatos distintos.

Nenhuma das soluções apresenta alternativa para tratar dados em tempo real (BP20). Todavia, vale salientar que o ArcGIS Open Data é o único que apresenta referência quanto a essa boa prática, indicando que ao ser integrado ao *ArcGIS for Server*, pode realizar a publicação de dados geográficos em tempo real para consumo.

Quando os dados não são produzidos em tempo real, é considerado uma boa prática fornecer os dados atualizados, deixando a frequência de atualização explícita (BP21). O Socrata e o Knoema apenas apresentam a informação da periodicidade de atualização nos metadados, mas a atualização não é realizada de forma automática. No entanto, a atualização no Socrata pode ser garantida por meio de uma aplicação adicional (DataSync[7]), mas que, ainda sim, não extrai os dados a serem publicados e necessita de integração com outras ferramentas. Contrapondo-se, o Junar permite que a frequência de atualização seja seguida quando os dados são consumidos por meio de uma API. Apesar de não apresentar a informação de quando os dados serão atualizados para os consumidores, por meio dos metadados, é apresentada a data da última consulta que foi realizada a atualização.

O uso de APIs para acesso a dados (BP23) é um consenso entre as soluções. Uma API oferece uma maior flexibilidade e capacidade de processamento para os consumidores de dados, permitindo também o uso de dados em tempo real e realização de filtros. Além disso, todas as soluções fazem uso de padrões da Web (BP24), tais como REST, na construção das APIs, assim como também fornecem uma documentação da API (BP25).

[7]https://github.com/socrata/datasync

Por fim, vale salientar que, até o momento, as soluções tem evitado alterações na API para não quebrar o código de quem está utilizando-a (BP26).

Tendo em vista que é provável que os produtores podem remover os dados da Web, é importante preservar seus identificadores. Assim, espera-se que seja possível dereferenciar o URI de um conjunto de dados mesmo ele não estando disponível e fornecer informações sobre o seu arquivamento (BP27). Foi observado que nenhuma das soluções permite a preservação dos identificadores dos conjuntos de dados. Quando um conjunto de dados não está mais disponível nas soluções, é retornada apenas uma mensagem de erro 404 (*NOT FOUND*). Logo, os identificadores não são mantidos e preservados nas soluções.

Com os dados publicados na Web, os consumidores podem acessá-los e criar suas próprias experiências. No entanto, os produtores de dados muitas vezes não tem o *feedback* dos consumidores sobre como os conjuntos de dados são usados e não oferecem maneiras eficazes para discutir essas experiências [Lóscio et al. 2016b]. São boas práticas o fornecimento de pelo menos um mecanismo para receber *feedback* (BP29) e a disponibilização dos *feedbacks* já coletados ao público (BP30). De forma geral, as soluções CKAN, Socrata, DKAN, Knoema atendem superficialmente a BP29. As mesmas disponibilizam opções muito simples para receber *feedback*, como formulários de contato. Além disso, apenas as soluções CKAN, Socrata e OpenDataSoft permitem a visualização de *feedback* emitidos por outros usuários.

Outro ponto bastante explorado pelas soluções é a possibilidade de gerar visualizações complementares de forma automática (BP32), como gráficos, tabelas interativas e sínteses estatísticas. Nesse ponto, é possível destacar o Knoema como a solução que mais apresenta tipos de visualizações distintas e possibilidades de filtros.

5. Discussão

As boas práticas apresentadas na Seção 3.1 auxiliam o entendimento do cenário de compartilhamento de dados na Web, uma vez que abordam os principais desafios a serem enfrentados na publicação e no consumo de dados na Web. É possível observar que a Web possibilita o compartilhamento de dados, *i.e.*, o acesso e a leitura de dados, de maneira bastante simples, sem a exigência de sistemas para controlar o acesso concorrente aos dados. Porém, a facilidade de compartilhamento em grande escala requer soluções flexíveis que possibilitem o consumo de dados por grupos de usuários previamente desconhecidos [Lóscio et al. 2016a].

De forma geral, foi verificado que as soluções existentes tem um foco maior na publicação dos dados, *i.e.*, no processo de disponibilização dos conjuntos de dados para uso na Web, compreendendo também os metadados, por meio de um catálogo. Contudo, constatou-se que nenhuma das soluções permite a extração e publicação de conjuntos de dados a partir de outras fontes de dados. Um exemplo seria a publicação de um conjunto de dados criado a partir um banco de dados hospedado em um Sistema de Gerenciamento de Banco de Dados. Esta funcionalidade permitiria garantir que os dados publicados na Web estejam sincronizados com a fonte de origem. Assim, à medida que ocorrerem atualizações nas fontes de origem, os dados publicados na Web também devem ser atualizados. Além disso, para possibilitar a publicação dos conjuntos de dados e garantir que estarão atualizados conforme sua fonte de dados de origem, as soluções deveriam prover mecanismos que permitam realizar a extração e transformação dos dados. De modo geral, deve ser possível

extrair os dados automaticamente a partir de diferentes fontes de dados e, posteriormente, realizar a transformação dos dados de origem para publicá-los na Web.

Como mencionado previamente, os conjuntos de dados podem ser alterados ao longo do tempo, seja para realizar uma melhoria, uma correção ou até mesmo a inserção de novos dados. Dessa forma, é importante manter versões dos conjuntos de dados e possibilitar o acesso as mesmas. No entanto, são poucos os meios disponibilizados pelas soluções analisadas para garantir um gerenciamento de versões adequado. Por exemplo, algumas soluções estão de acordo com a BP7 por apresentar um indicador de versão para cada conjunto de dados, mas elas não proporcionam ferramentas que permitam o gerenciamento das versões.

Somado a isso, todas as soluções de publicação de dados analisadas utilizam apenas um subconjunto do DCAT para coletar informações (metadados) sobre os conjuntos de dados e distribuições. Foi também identificado uma carência de maior suporte para coletar informações sobre qualidade e versionamento, além de prover o processo de curadoria de informações relacionadas aos consumidores e aplicações que fazem uso dos dados. Também é importante que as soluções adotem mais mecanismos para coleta de *feedback*, uma vez que boa parte só disponibiliza um simples formulário de contato. Além disso, nenhuma das soluções analisadas garante a preservação dos conjuntos de dados ao longo do tempo, ou seja, permitem a exclusão dos dados, mas não mantém os identificadores ativos para que os consumidores obtenham informações acerca do conjunto de dados.

Portanto, a fim de que as soluções estudadas neste artigo possam prover o compartilhamento de dados na Web de forma mais adequada, é importante que sejam resolvidas questões como as relacionadas à qualidade dos dados, *feedback*, versionamento, atualização e preservação dos dados, que são desafios até então pouco explorados.

6. Conclusão e Trabalhos Futuros

A quantidade de soluções para publicação de dados na Web tem crescido nos últimos anos. Todavia, por apresentarem diversidade de requisitos e de características, encontrar a mais apropriada para um cenário específico não é uma tarefa trivial. Essa tarefa torna-se ainda mais complexa uma vez que boa parte das soluções usadas não são especificadas ou avaliadas em trabalhos científicos. A fim de suprir essa lacuna, foi realizado um levantamento e análise das principais soluções de publicação de dados na Web.

Neste artigo, é apresentado um *survey* comparativo das soluções de publicação de dados sob o ponto de vista do conjunto de Boas Práticas para Dados na Web recomendado pelo W3C. O resultado do levantamento permitiu detectar 15 soluções *Open Source* ou comerciais, das quais 7 foram descritas e analisadas por serem utilizadas em pelo menos 10 portais que estão em uso atualmente. Como resultado da análise, foi identificado que as soluções analisadas atendem aos principais desafios de compartilhamento de dados na Web que foram descritos pelo W3C. De modo geral, a análise mostrou que existem limitações e lacunas nas soluções, como a carência de ferramentas que possibilitem o gerenciamento adequado de versões e a preservação dos conjuntos de dados, por exemplo.

Como trabalho futuro, pretende-se aumentar o número de soluções analisadas. Somado a isso, também pretende-se buscar um entendimento mais amplo dos principais desafios para publicação e consumo dos dados na Web. A partir desse entendimento, será

possível elencar os requisitos esperados de uma solução de publicação de dados na Web a fim de possibilitar o gerenciamento adequado dos conjuntos de dados publicados.

7. Agradecimentos

Este trabalho foi parcialmente apoiado pelas instituições de financiamento brasileiras Fundação de Amparo a Ciência e Tecnologia de Pernambuco (FACEPE) e Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). Marcelo Iury e Lairson Emanuel são bolsistas do CNPq na modalidade doutorado. Marcelo Iury também agradece a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa modalidade doutorado sanduíche. Os autores também gostariam de agradecer aos seus colegas do projeto Aladin por sua contribuição para este artigo.

Referências

Dietrich, D., Gray, J., McNamara, T., Poikola, A., Pollock, R., Tait, J., and Zijlstra, T. (2009). Open data handbook. http://opendatahandbook.org/guide/en/. Acesso em 05 de dezembro de 2016.

Goldacre, B. and Gray, J. (2016). Opentrials: towards a collaborative open database of all available information on all clinical trials. *Trials*, 17(1):164.

Hoxha, J. and Brahaj, A. (2011). Open government data on the web: A semantic approach. In *Emerging Intelligent Data and Web Technologies (EIDWT), 2011 International Conference on*, pages 107–113. IEEE.

Isotani, S. and Bittencourt, I. (2015). *Dados Abertos Conectados: Em busca da Web do Conhecimento*. NOVATEC.

Lisowska, B. (2016). Metadata for the open data portals. Technical report, Joined-up Data Standards. http://juds.joinedupdata.org/wp-content/uploads/2016/12/JUDS-DP6-Metadata-for-the-open-data-portals.pdf. Acesso em 21/05/2017.

Lóscio, B. F., Burle, C., and Calegari, N. (2016a). Data on the Web Best Practices. W3C Recommendation, World Wide Web Consortium (W3C). https://www.w3.org/TR/dwbp/.

Lóscio, B. F., Burle, C., and Calegari, N. (2016b). Data on the web best practices: Challenges and benefits. In *Open Data Reserach Symposium (ODRS 2016)*.

Lóscio, B. F., Oliveira, M. I. S., and Bittencourt, I. I. (2015). Publicação e Consumo de Dados na Web: Conceitos e Desafios. *Tópicos em Gerenciamento de Dados e Informações (Mini Cursos - SBBD 2015)*, d:39–69.

Milić, P., Veljković, N., and Stoimenov, L. (2015). Linked relations architecture for production and consumption of linksets in open government data. In *Conference on e-Business, e-Services and e-Society*, pages 212–222. Springer.

Millette, C. and Hosein, P. (2016). A consumer focused open data platform. In *Big Data and Smart City (ICBDSC), 2016 3rd MEC International Conference on*, pages 1–6. IEEE.

Stråle, J. and Lindén, H. (2014). An evaluation of platforms for open government data. http://kth.diva-portal.org/smash/get/diva2:723341/FULLTEXT01.pdf. Acesso em 20/05/2017.

paper:170764

Uma avaliação de Eficiência e Eficácia da Combinação de Técnicas para Deduplicação de Dados

Levy de Souza Silva[1]*, Dimas Cassimiro Nascimento Filho[2], Mirella M. Moro[1]

[1]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901 – Brasil

[2]Universidade Federal Rural de Pernambuco (UFRPE)
Unidade Acadêmica de Garanhuns (UAG) - Garanhuns - PE - Brasil

{levysouza,mirella}@dcc.ufmg.br, dimascnf@uag.ufrpe.br

Abstract. *Data Deduplication is the task of identifying and eliminating duplicate records in a single database. It is a complex process that involves several steps, including: defining blocking key, similarity function and indexing method. There are several approaches for each of these steps. In this context, the objective of this work is to find the best combination for such algorithms aiming to improve the efficiency and effectiveness of the deduplication process as a whole. To this end, we present an experimental evaluation using real and artificial datasets. The results point to distinct combinations that present better results in specific situations.*

Resumo. *Deduplicação de dados é a tarefa de identificar e eliminar registros duplicados em um única base de dados. É um processo complexo que envolve várias etapas, incluindo: definição de chave de bloco, função de similaridade e método de indexação. Existem diversas abordagens para cada uma dessas etapas. Então, o objetivo deste trabalho é encontrar a melhor combinação para tais algoritmos visando melhorar a eficiência e eficácia do processo como um todo. Para tal, apresentamos uma avaliação experimental utilizando bases de dados reais e artificiais. Os resultados apontam para combinações distintas que apresentam melhor resultados em situações específicas.*

1. Introdução

Devido a enorme quantidade de dados coletados e armazenados ao longo dos anos pelas empresas, agências de governo e projetos de pesquisa, o estudo de técnicas que permitam a análise, processamento e mineração dos dados de forma eficiente têm sido atraído pela acadêmia e indústria. Uma tarefa que tem ganhado relevância em muitos domínios de aplicação é a tarefa de deduplicação de dados. Deduplicação de dados é uma abordagem para identificar e eliminar registros duplicados em bases de dados. Registros duplicados são instâncias de dados que representam a mesma entidade no mundo real. Esta tarefa é referenciada por diversas nomenclaturas na literatura, incluindo: I) *Data Deduplication* e *Duplicate Detection*, quando o processo ocorre em uma única fonte de dados; e II) *Record Linkage*, *Entity Resolution*, *Data Matching* e *Object Identification*, quando são utilizadas várias fontes de dados [Christen 2012a].

*Trabalho desenvolvido enquanto estudava na UFRPE

No setor da saúde, em sistemas de vigilância, por exemplo, a deduplicação de dados tem o objetivo de detectar padrões suspeitos, tais como surtos de doenças contagiosas. Muitas empresas usam essa abordagem para melhorar a qualidade dos dados, compilar listas de discussão ou para comparar dados entre organizações. Órgãos do governo estão empregando cada vez mais as técnicas de deduplicação em serviços de segurança social buscando identificar pessoas que se inscreveram várias vezes para os programas sociais. Outra área de aplicação que tem ganhado interesse está relacionada com a detecção de crimes e fraudes [Christen 2012a]. Agências de segurança e investigadores de crimes buscam rapidamente fazer o cruzamento dos dados de criminosos, presentes em várias fontes, com o intuito de encontrar indivíduos que representam a mesma pessoa mas fornecem informações pessoais fictícias quando questionados em depoimentos.

O problema de encontrar registros duplicados não é aplicado somente a dados pessoais. Às vezes é preciso identificar duplicidade em registros de empresas, produtos de consumidores, publicações, citações bibliográficas, páginas web, resultados de sistemas de busca ou até mesmo sequências de genoma. No campo de recuperação de informação, por exemplo, é importante remover documentos duplicados (páginas web e citações bibliográficas) dos resultados retornados pelos motores de busca, pelas bibliotecas digitais ou até mesmo pelos sistemas automáticos de indexação de texto [Hajishirzi et al. 2010]. Outra aplicação com crescimento significante é a identificação de produtos duplicados em sistemas de lojas online [Christen 2012a].

A identificação de registros duplicados é um processo complexo e composto de várias epatas. A Figura 1 apresenta uma visão geral desta tarefa em uma abordagem de *Record Linkage*. O passo a passo é composto das seguintes etapas: I) o conjunto dos dados é submetido a um pré-processamento para deixar os dados mais limpos e concisos; II) todos os registros precisam ser indexados por uma chave de bloco (BK) com o intuito de criar pequenos grupos de registros que podem corresponder ao mesmo registro, evitando assim uma comparação produto cartesiano entre os dados; III) todos os registros pertencentes a cada um dos blocos são comparados por meio de funções de similaridade; IV) os registros são classificados como duplicados, não duplicados e como possíveis duplicados de acordo com um limiar de similaridade; e V) todo o processo é avaliado com o intuito de medir a eficácia e eficiência dos algoritmos aplicados em cada uma das etapas. Para identificar registros duplicados em uma única base de dados, como acontece na abordagem de *Data Deduplication*, o processo exibido na Figura 1 também pode ser aplicado, porém apenas uma fonte de dados é utilizada para prover os dados.

Nos últimos anos, vários estudos de deduplicação em diferentes contextos foram publicados [Borges et al. 2008, Canalle et al. 2016, Chen et al. 2012, Christen 2012b, Machado et al. 2016, Vesdapunt et al. 2014]. Assim, embora tenham sido desenvolvidas várias técnicas para cada uma das etapas do processo, uma pequena quantidade de estudos apresentam um levantamento experimental incluindo tais técnicas. Além disso, poucos estudos abordam métodos que buscam analisar, no mesmo ambiente experimental, quais os melhores algoritmos a serem utilizados em cada uma das etapas do processo.

Nesse sentido, este trabalho tem o objetivo de avaliar a eficiência e eficácia da combinação de algoritmos, técnicas e métodos de deduplicação de dados. Para isso, diversos experimentos foram executados em um conjunto de bases de dados reais e artificiais, com tamanho variando entre 10.000 e 270.00 instâncias, contendo registros originais

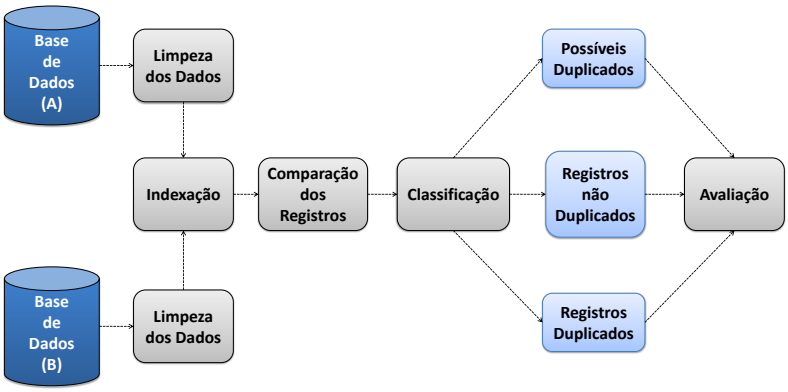


Figura 1. Processo universal de identificação de registros duplicados em uma abordagem de *Record Linkage*. Adaptado de [Christen 2012a].

e duplicados. Dessa forma, as principais contribuições deste trabalho são: I) realizar uma avaliação de eficiência entre as funções de similaridade, métodos de indexação e chaves de bloco; II) realizar uma avaliação de eficácia diante da combinação das técnicas aplicadas; e III) propor uma melhor combinação de algoritmos para serem utilizados nas etapas da deduplicação de dados.

As próximas seções estão organizadas da seguinte maneira. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 aborda a fundamentação teórica. A Seção 4 expõe os experimentos realizados, as discussões e os resultados alcançados. A Seção 5 apresenta as conclusões do trabalho e aponta trabalhos futuros.

2. Trabalhos Relacionados

Machado et al. [2016] apresentam um método para deduplicação de contatos de dispositivos móveis utilizando similaridade textual e aprendizagem de máquina. Os escores produzidos por meio das funções de similaridade são destinados a treinar um modelo de classificação que pode ser adotado como estratégia para integração dos contatos duplicados. A qualidade do método é avaliada em uma base de dados real com 2.000 contatos.

Canalle et al. [2016] apresentam uma abordagem de seleção de atributos mais relevantes para a etapa de comparação dos registros em um processo de integração de dados de bibliotecas digitais. Diversos critérios são extraídos dos dados com o intuito de identificar a relevância de cada atributo. Para avaliar a relevância, são consideradas métricas que calculam a repetição e densidade dos atributos, por exemplo, e métricas relacionadas a qualidade das fontes dos dados.

Vesdapunt et al. [2014] modelam o problema da identificação de registros duplicados como um grafo. Cada aresta denota a probabilidade do par de registros (vértices) representarem o mesmo objeto. O objetivo do trabalho é abordar o problema em um contexto de *crowdsourcing* e utilizar o conhecimento prévio de comparações para evitar comparações futuras, melhorando dessa forma o desempenho dos algoritmos. Por exemplo, considerando que $a = b$ e que $b = c$, pode ser inferido que $a = c$?

Christen [2012b] apresenta um *survey* de métodos de indexação e realiza uma avaliação experimental com os métodos apresentados. No trabalho foram analisados os

métodos *Standard Blocking*, *Sorted Neighborhood Indexing*, *Q-Gram-Based Indexing*, *Suffix Array-Based Indexing*, *Canopy Clustering* e *String-Map-Based Indexing*. O objetivo do trabalho foi avaliar a eficiência das técnicas a fim de responder a perguntas como: De que maneira os valores das chaves de bloco geradas influenciam na qualidade dos pares de registros candidatos criados? Como os diferentes métodos se comportam em diversos tipos de dados? Quais as práticas que se mostram com melhor escalabilidade às bases de dados de maior dimensão? Segundo o autor, a técnica *Q-gram Based Indexing* apresenta-se como uma das mais lentas, não sendo apropriada para deduplicação de grandes bases de dados. Por outro lado, ele afirma que as abordagens mais tradicionais, como a definição de blocos e a metodologia de vizinhança são as técnicas mais rápidas.

Chen et al. [2012] propoem um método baseado em aprendizagem de máquina para selecionar o atributo ou o grupo de atributos mais relevantes para a etapa de comparação dos registros. Além disso, o limiar e os pesos de similaridade são definidos automaticamente. Funções de similaridade como *Jaccard Distance*, *Jaro Distance* e *Edit Distance* foram utilizadas para comparação das instâncias e experimentos foram executados em uma portação da base de dados do *CORA*.

Borges et al. [2008] apresentam uma abordagem para identificação de registros duplicados em um contexto de bibliotecas digitais visando integrar dados de várias fontes. São especificadas diversas funções de similaridade, incluindo: *YearSim*, *IniSim*, *NameMatch* e *MetadataMatch*. Segundo os autores, essas funções identificam com maior eficácia os registros duplicados em um conjunto de metadados de objetos digitais e podem ser utilizadas para a etapa de comparação das instâncias.

Desse modo, observa-se que nenhum dos estudos citados, com exceção de Christen [2012b], apresenta uma metodologia para analisar ao mesmo tempo os melhores algoritmos a serem aplicados em cada uma das etapas do processo. Considerando o trabalho de Christen [2012b] que investigou métodos de indexação e seus comportamentos em diversos contextos de base de dados, o objetivo aqui não é investigar apenas os métodos de indexação, mas analisar no mesmo plano de experimentos todo o conjunto de algoritmos que são aplicados em cada uma das etapas do processo de deduplicação de dados.

3. Fundamentação Teórica

Duplicatas. Um registro duplicado ou simplesmente uma duplicata, no contexto de banco de dados, se refere a um registro gerado com base em uma entidade original, que sofreu algumas modificações ou até mesmo registros que são produzidos por meio de erros causados pelo usuário durante o processo de digitação.

3.1. Funções de Similaridade.

Diversas funções de similaridade são propostas pela literatura para comparação de registros. De forma geral, estas funções têm o objetivo de calcular computacionalmente o quanto dois valores são correspondentes, ou seja, calcular um valor de similaridade no intervalo $[0, 1]$, entre dois registros. Assim, o valor obtido é igual a 1 quando há concordância total entre os registros comparados.

***Jaro*.** Tem o objetivo de verificar o número de caracteres iguais em dois registros [Jaro 1989]. Essa função leva em conta o tamanho das variáveis e ainda os tipos de

erros que comumente ocorrem com variáveis alfanuméricas. Dessa forma, um caractere é considerado em comum se estiver presente em ambos os registros ao mesmo tempo. Além disso, o número de transposições realizadas para se obter a igualdade entre dois caracteres é observado. Assim, para dois registros a e b, uma transposição ocorrerá quando dois valores que ocupem a mesma posição forem diferentes, ou seja, $a[i] \neq b[i]$. A função estabelece ainda que a distância entre os itens permutados não ultrapasse a metade do comprimento do menor registro.

Jaro Winkler. É uma extensão do *Jaro* e foi proposto por Winkler [1990]. Essa função também investiga a similaridade, porém observa de forma mais precisa a igualdade dos quatro primeiros caracteres, ou seja, existe um número $(0 \leq p \leq 4)$, que representa o número de elementos em comum nas i primeiras instâncias dos registros a serem comparados. Por exemplo, para o par de valores "$peter$" e "$petra$", existe um prefixo comum "pet", determinando assim $p = 3$.

***Levenshtein* ou *Edit Distance*.** É uma das primeiras funções utilizadas na comparação de registros. Considerando dois registros X e Y, é calculado o número mínimo de modificações necessárias para transformar X em Y, ou vice versa. De acordo com Navarro [2001], essas alterações restringem-se às funções de inserção, remoção ou substituição de caracteres. Sendo assim, para transformar por exemplo, o valor "$kitten$" em "$sitting$" são necessárias no mínimo três mudanças: I) *sitten* - (substituição de k por s); II) *sittin* - (substituição de e por i); e III) *sitting* - (inserção de g no final).

3.2. Chaves de Indexação.

O gargalo de desempenho em um sistema de deduplicação de dados é geralmente determinado pelo número de comparações entre os registros, o que pode ser auxiliado por meio de um bom índice. Segundo Christem [2012b], a indexação pode ser vista como uma filtragem, porque se baseia em criar estruturas de BKs que agrupam valores semelhantes.

***Soundex*.** É uma técnica desenvolvida e patenteada por Odell e Russell [1918], com base na pronúncia da língua inglesa. O *soundex* codifica uma sequência de caracteres mantendo o primeiro caractere e converte os demais para dígitos entre zero e seis. A Tabela 1 apresenta um exemplo dessa codificação.

***Suffix*.** Utiliza uma metodologia que cria *substrings* de uma palavra, chamadas de sufixos. Segundo Christen [2012a], os sufixos de uma *string* são todas as suas *substrings* com um ou mais caracteres. A Tabela 3 apresenta exemplos de sufixos.

3.3. Método de Indexação.

Diversos métodos são propostos para comparação dos registros. Os principais métodos utilizam conceitos de comparação por blocos e por vizinhos.

Tabela 1. *Soundex* [Christen 2012a].

Letra	Dígito
a, e, i, o, u, y, h, w	0
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

Tabela 2. Exemplos de Codificação *Soundex*.

Nome	Código *Soundex*
peter	p360
pete	p300
pedro	p360
stephen	s315
steve	s310
smith	s530
smythe	s530

Tabela 3. Exemplos de Sufixos de uma *String* Apresentados por Christen [2012a].

ID	Nome	Sufixos
r1	katherina	katherina, atherina, therina, herina, erina, rina
r2	catrina	catrina, atrina, trina, rina
r3	catherine	catherine, atherine, therine, herine, erine, rine

Tabela 4. Comparação de Registros com *Standard Blocking* [Christen 2012a].

RecID	GivenName	Surname	Sndx(GiN)	Sndx(SurN)
a1	peter	myler	p360	**m460**
a2	pedro	smith	p360	s530
a3	steve	peters	s315	p362
a4	christine	miller	c623	**m460**
b1	kristina	miller	k623	**m460**
b2	stephen	peter	s315	p360
b3	kylie	smith	k400	s530
b4	pete	myler	p300	**m460**

Standard Blocking. É um algoritmo que tem o intuito de gerar um conjunto de blocos, onde cada bloco agrupa entidades semelhantes. Logo, apenas as entidades que pertencem ao mesmo bloco são comparadas entre si. Esses agrupamentos são definidos de acordo com as BKs, geradas para cada instância. Segundo Fellegi e Sunter [1969], essa é uma das técnicas mais utilizadas durante décadas, abordando métodos para correspondência de dados e para diminuição das redundâncias. A Tabela 4 apresenta um exemplo de comparação dos elementos utilizando a técnica *Standard Blocking*. Considerando a chave de bloco **m460** da coluna *Sndx(SurN)*, por exemplo, as comparações são realizadas com os valores: (a1,a5), (a1,b1), (a1,b4), (a5,b1), (a5,b4) e (b1,b4).

Sorted Neighborhood. É uma alternativa para definição de quais registros comparar em um conjunto de dados. Esse conceito foi apresentado por Hernández e Stolfo [1995] , onde ao invés de definir vários blocos, criados por meio da indexação, os valores são combinados através de uma chave de triagem, que é semelhante a uma BK. Dessa forma, antes de realizar as comparações em um banco de dados, todos os elementos são ordenados de acordo com a BK definida para cada elemento. Após isso, uma janela deslizante de tamanho $w > 1$ percorre todas as instâncias do banco de dados e, enquanto isso, o primeiro valor da janela é comparado com todos os outros.

4. Experimentos e Discussões

A metodologia dos experimentos[1] consiste em realizar uma combinação das técnicas aplicadas na deduplicação de dados, ou seja, para cada tratamento foi utilizado: uma chave de bloco, uma função de similaridade e um método de indexação. Nas experimentações relacionadas a eficácia, foram avaliadas medidas de *Precision*, *Recall* e *F-Measure* [Brizan and Tansel 2006]. A avaliação de eficácia ateve-se a verificar qual o tratamento que, quando aplicado em uma única base de dados, obteve o melhor valor de *F-Measure*, relacionando este ao total de duplicatas detectadas.

Para definir o limiar de similaridade e o tamanho w das janelas utilizadas no algoritmo de vizinhança foram realizados testes em algumas bases de dados. Assim, investigou-se com quais valores a *F-Measure* foi melhorada. Deste modo, após os testes iniciais, estabeleceu-se um limiar de 0.90 para a similaridade e o tamanho w foi definido

[1]Disponível em: https://www.dropbox.com/s/49xpytfl3ldkmms/C%C3%B3digoFonte.zip?dl=0

Tabela 5. Plano de Experimentos Realizados com a Técnica *Standard Blocking*.

Trat.	Indexação	Chave	Função
1	SB	*Soundex*	*JaroWinkler*
2	SB	*Suffix*	*JaroWinkler*
3	SB	Ambos	*JaroWinkler*
4	SB	*Soundex*	*Jaro*
5	SB	*Suffix*	*Jaro*
6	SB	Ambos	*Jaro*
7	SB	*Soundex*	*Edit Distance*
8	SB	*Suffix*	*Edit Distance*
9	SB	Ambos	*Edit Distance*

Tabela 6. Plano de Experimentos Realizados com a Técnica *Sorted Neighborhood*.

Trat.	Indexação	Chave	Função
10	SN	*Soundex*	*JaroWinkler*
11	SN	*Suffix*	*JaroWinkler*
12	SN	Ambos	*JaroWinkler*
13	SN	*Soundex*	*Jaro*
14	SN	*Suffix*	*Jaro*
15	SN	Ambos	*Jaro*
16	SN	*Soundex*	*Edit Distance*
17	SN	*Suffix*	*Edit Distance*
18	SN	Ambos	*Edit Distance*

de acordo com a quantidade de tuplas na base de dados. Logo, w assumiu os valores: I) $w = 50$, para as bases de dados com menos de 10.000 instâncias; II) $w = 150$, para as bases de dados com mais de 10.000 e menos de 50.000 instâncias; e III) $w = 500$, para as bases de dados maiores.

A avaliação da eficiência consiste em calcular o tempo médio de 30 execuções em cada tratamento de cada base de dados. Os experimentos executaram-se em um *notebook* com processador *Pentium Dual-Core 2.30GHz*, memória *RAM* de 2,00GB e Sistema Operacional *Windows* 7 de 32 bits. As Tabelas 5 e 6 apresentam os tratamentos empregados no experimento, esses tratamentos foram executados em todas as bases de dados.

Bases de Dados[2]. O conjunto dos dados utilizados nos experimentos foram extraídos de fontes livres e artificiais, formando um total de seis bases de dados. Assim, três delas procederam de dados livres, uma foi criada pela ferramenta *Data Set Generator Program*[3], uma foi baixada do endereço Dados Prontos[4] e uma foi selecionada da Superintendência de Seguros Privados[5](SUSEP).

O *Data Set Generator Program* é uma ferramenta que produz dados e duplicatas artificialmente de acordo com os parâmetros do usuário. Ela é parte do projeto *Febrl* [6] e é baseada nas ideias apresentadas por [Hernández and Stolfo 1995]. Os dados livres estão disponíveis no Portal Brasileiro de Dados Abertos[7]. O SUSEP contém dados extraídos dos formulários de informações periódicas enviados pelas companhias seguradoras, resseguradoras, entidades abertas de previdência privada e sociedades de capitalização.

Geração das Duplicatas. Com exceção da base de dados completa gerada pela ferramenta Febrl, todas as duplicatas foram produzidas por meio da ferramenta UAG-DUP[8]. UAG-DUP é uma ferramenta para produção de duplicatas utilizando bases de dados reais e parâmetros customizáveis. Ela foi desenvolvida, durante o trabalho, em vista da dificuldade encontrada no *Data Set Generator Program*, que não possibilita a concepção de dados duplicados utilizando uma base de dados específica. A UAG-DUP adota os principais conceitos para produção de duplicatas apresentados por [Ioannou et al. 2013].

[2]Disponível em: https://www.dropbox.com/s/upashltf5pmul6t/BasesDeDadosExperimentos.sql?dl=0
[3]Disponível em: https://cs.anu.edu.au/people/Peter.Christen/Febrl/febrl-0.3/febrldoc-0.3/node70.html
[4]Disponível em: https://gabrielrb.net/2011/10/18/dados-prontos-em-formato-sql-e-csv/
[5]Disponível em: https://www2.susep.gov.br/menuestatistica/SES/principal.aspx
[6]Disponível em: https://sourceforge.net/projects/febrl/
[7]Disponível em: www.dados.gov.br
[8]Disponível em: https://www.dropbox.com/s/56w9cgwyl6wpf49/UAG-DUP.zip?dl=0

Tabela 7. Proporção de Duplicatas e Registros Originais nas Bases de Dados.

Nº	Nome	Nº de Reg.	Tamanho	% de Duplicatas	QTDMAXDPI	NMAXMODPR
1	UBS5	38.875	$4,41MB$	5%	3	2
2	UBS10	40.729	$4,63MB$	10%	5	3
3	BDT5	10.132	$567KB$	5%	3	2
4	BDT10	10.614	$958KB$	10%	5	2
5	RC5	48.059	$8,44MB$	5%	3	3
6	RC10	50.350	$8,86MB$	10%	5	3
7	RP5	10.208	$1,49MB$	5%	3	3
8	RP10	10.697	$1,57MB$	10%	5	2
9	SUSEP10	270.146	$31,2MB$	10%	3	3
10	FebrlData	55.001	$9,20MB$	10%	4	4
11	NFebrl	58.126	$3,15MB$	10%	4	2

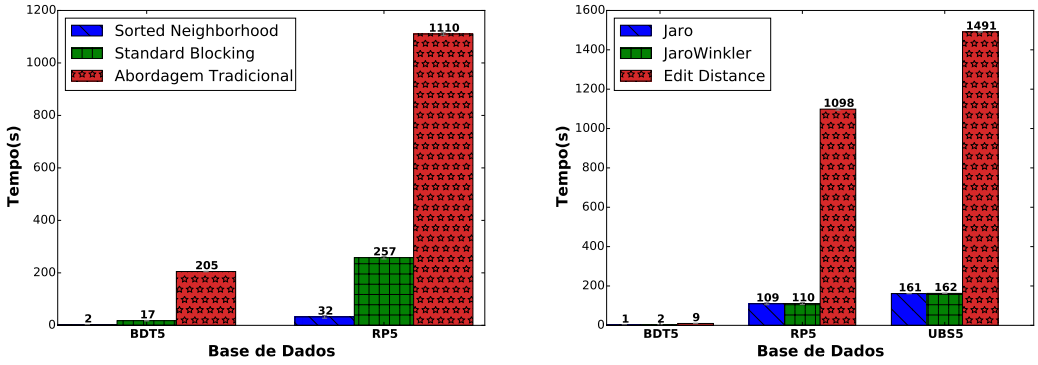
Assim, para cada uma das bases de dados selecionadas criaram-se duas cópias, sendo a primeira com uma porcentagem de 5% de registros duplicados e a outra com 10% de duplicatas. Para a base de dados dos administradores da SUSEP e a dos nomes concebidos pelo *Data Set Generator Program*, foi produzido um percentual de 10% de duplicatas. A Tabela 7 apresenta: I) as informações das bases de dados; II) a quantidade máxima de duplicatas geradas de cada instância (QTDMAXDPI); e III) o número máximo de modificações em cada registro (NMAXMODPR) para geração de uma duplicata.

4.1. Avaliação da Eficiência

A Figura 2a apresenta uma comparação dos tempos de execução dos algoritmos *Standard Blocking* e *Sorted Neighborhood* em relação ao tempo consumido quando não utilizou-se nenhuma das técnicas, ou seja, cada elemento da base de dados foi comparado com todos os outros. Percebe-se, que a ausência de estratégias de indexação contribui para um tempo de execução elevado. Além disso, os tempos de execução possuem variações consideráveis de acordo com a quantidade de tuplas existentes no banco de dados, como observado nas bases $BDT5$ e $RP5$. Sendo assim, a estratégia de vizinhança, por exemplo, é 34 vezes mais rápida do que o método tradicional, como observa-se no conjunto de dados $RP5$. Esse tempo de execução superior é ocasionado pelo número total de pares candidatos que são criados, chegando, por exemplo, a um valor de $51.323.646$ na base de dados $BDT5$. Por sua vez, com o método de vizinhança, são criados 485.160 pares.

A Figura 2b exibe o tempo consumido pelas funções de similaridade para calcular a semelhança entre dois registros. Observa-se, que o tempo para o método *Edit Distance* é bastante elevado, chegando este a ser nove vezes mais lento do que o algoritmo *Jaro Winkler*, como é notado em $UBS5$. Sendo assim, é observado que a técnica *Edit Distance* não apresenta bons resultados quando o tamanho das *strings* que são comparadas são grandes, o que ocorre na base de dados $UBS5$. Por outro lado, quando o tamanho das *strings* é relativamente pequeno, como na base de dados $BT5$, o algoritmo apresenta um tempo de execução semelhante aos demais. Para as funções *Jaro* e *JaroWinkler*, nas bases de dados $BTD5$, $RC5$ e $UBS5$, os tempos de execução são semelhantes e as diferenças são quase imperceptíveis como exibido na Figura 2b.

A Figura 3 apresenta uma comparação para avaliar o desempenho do algoritmo *Standard Blocking*, de acordo com cada BKV definida. Percebe-se, que a eficiência do método está proporcionalmente relacionada a distribuição das BKs, ou seja, o número total de chaves distintas que são criadas influencia no tempo de execução do algoritmo.



(a) (b)

Figura 2. Tempo de Execução *Standard Blocking*, *Sorted Neighborhood* e Abordagem Tradicional (a), (b) Tempo de Execução das Funções de Similaridade.

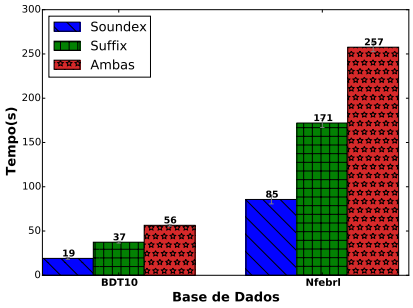


Figura 3. Tempo de Execução do Algoritmo *Standard Blocking*.

Neste exemplo, são comparadas as técnicas de indexação *soundex* e *suffix*, como também, a aplicação de ambos os métodos de indexação ao mesmo tempo. Sendo assim, foi observado que na base de dados $BDT10$, com a técnica *soundex*, criaram-se 2.437 blocos e conceberam-se 87.271 pares candidatos, por outro lado, com a *suffix*, originaram-se 4.361 blocos e formaram-se 130.015 pares candidatos.

Diante disso, é observado que existe um ganho significativo no quesito eficiência quando certas técnicas de indexação e funções de similaridade são aplicadas. Considerando o trabalho de Machado et al. [2016], por exemplo, em que os contatos foram comparados todos com todos, a indexação seria uma estratégia fundamental a ser adotada, pois tanto a memória quanto o processamento disponível nos dispositivos móveis são limitados. Logo, não é uma abordagem viável realizar todas as comparações.

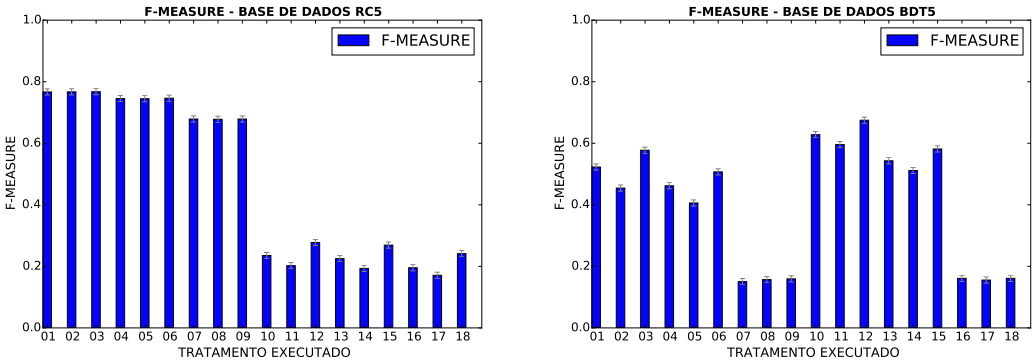
4.2. Avaliação da Eficácia

A Figura 4a apresenta os cálculos de *F-Measure* alcançados na base de dados $RC5$. Neste caso, o algoritmo *Jaro Winkler* produziu resultados melhores do que o *Jaro*, como visto no tratamento $N^{o}03$. Neste exemplo, a base de dados em questão apresenta uma média de 11 caracteres por *string* e a maioria dos seus dados figuram-se como *strings* do tipo: “Tim”, “Vivo”, “GVT”, “SKY”, “Banco BMG” e “Vivo - Telefônica”, o que torna mais fácil de existir uma concordância nas quatro primeiras instâncias dos registros duplicados criados, aumentando assim a similaridade *Jaro Winkler* em contrapartida a *Jaro*. Por outro lado, é

possível observar nesta base de dados que, comparando-se as técnicas de blocagem SB e SN, tratamentos de N^o01 à N^o09 e do N^o10 em diante, respectivamente, o algoritmo de vizinhança não obteve bons resultados. Este fato é ocasionado pela quantidade de tuplas existentes na base de dados. A base de dados $RC5$ possui cerca de 50.000 instâncias, o que pode distanciar o valor duplicado do original. Além disso, dependendo do tamanho w de janela que desliza sobre os dados a detecção das duplicatas fica comprometida, pois a janela pode não englobar todas as duplicatas.

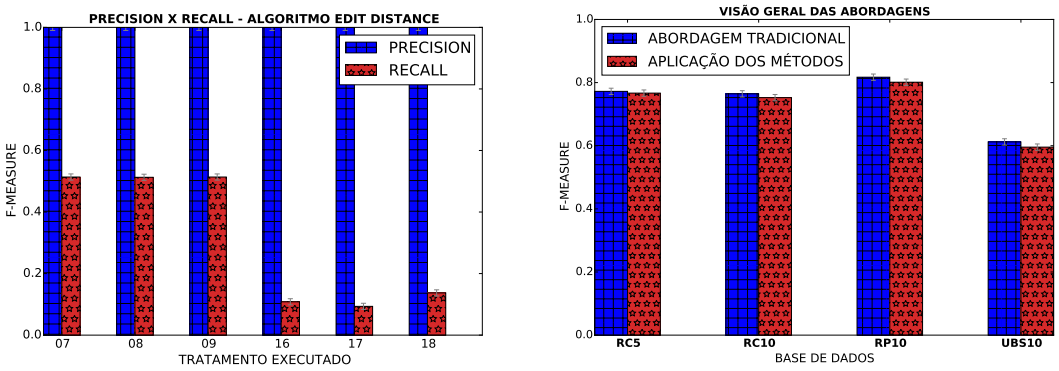
A Figura 4b exibe os cálculos de *F-Measure* obtidos na base de dados $BDT5$. É observado que o método de vizinhança obteve resultados mais significativos em quase todos os tratamentos, execuções N^o10 até a N^o18. E ainda, que a execução N^o12 na qual foi utilizada a função de similaridade *Jaro Winkler*, apresentou a melhor eficácia. Sendo assim, verificando-se a similaridade foi constatado que a base de dados utilizada apresenta *strings* pequenas nos seus atributos, abordando novamente a discussão exposta na Figura 4a. Dessa forma, atributos que contém uma pequena quantidade de caracteres, como os nomes de distritos, por exemplo, intensificam para as duplicatas a igualdade das 4 primeiras ocorrências dos caracteres. Por outro lado, quanto ao método de indexação, para a base de dados em questão, o resultado é atribuído também a quantidade de instâncias da mesma. Neste exemplo, a base de dados tem em média 10.000 tuplas. Sendo assim, os registros duplicados não se distanciam de forma drástica um do outro quando ocorre a indexação, o que não acontece em bases de dados maiores.

A Figura 5a apresenta uma comparação da *Precision* com o *Recall* com foco no algoritmo *Edit Distance* na base de dados $RC5$. Observa-se neste exemplo que, para os tratamentos N^o07, N^o08, N^o09, N^o16, N^o17 e N^o18 a função de similaridade *Edit Distance* apresentou uma precisão máxima, ou seja, não classificou nenhum registro que não era duplicado como um valor duplicado, mas por outro lado, o *recall* é bastante comprometido, onde identificou-se uma quantidade insignificante de duplicatas, o que piora drasticamente a medida de *F-Measure*. Sendo assim, os resultados ruins estão atrelados ao tamanho das *strings* de cada atributo, pois é necessário um maior número de modificações para transformar um valor em outro, o que diminuirá o valor de similaridade dos atributos comparados, comprometendo o *recall* e, consequentemente, a *F-Measure*.



(a) (b)

Figura 4. Eficácia: *Jaro* X *Jaro Winkler* na Base de Dados $RC5$ (a), (b) Eficácia dos Algoritmos SB e SN na Base de Dados $BTD5$.



(a) (b)

Figura 5. *Precision* X *Recall* Algoritmo *Edit Distance* (a), (b) Avaliação Geral das Técnicas de Blocagem, Funções de Similaridade e Método de Indexação.

A Figura 5b apresenta as medidas de *F-Measure* alcançadas em uma comparação onde foi utilizado no primeiro caso, a combinação dos algorítimos e, no segundo, a ausência dos métodos de indexação, ou seja, comparando-se todos os registros da base de dados. Dessa forma, aplicaram-se nas bases de dados $RC5$, $RC10$, $RP10$ e $UBS10$ o método de indexação *Standard Blocking*, ambas as chaves de bloco e a função de similaridade *JaroWinkler*. Assim, pode ser observado que com a aplicação da combinação dos métodos os resultados obtidos são significantes para o processo de deduplicação de dados, tanto para bases de dados com poucas instâncias quanto para bases de dados maiores.

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou um estudo sobre a avaliação da eficiência e eficácia da combinação de técnicas para deduplicação de dados. Diferente dos trabalhos anteriores, a metodologia apresentada buscou comparar todas as técnicas no mesmo ambiente experimental. Diversos métodos e técnicas do estado da arte foram implementados e vários experimentos envolvendo bases de dados reais e artificiais foram executados.

Comparando-se os métodos de indexação, observou-se que para as base de dados com um conjunto pequeno de instâncias, como a $BTD5$, o método de vizinhança mostrou-se eficaz e muito mais eficiente do que o método *Standard Blocking*. Para as base de dados maiores, como a $RC5$, a utilização da técnica de blocos produziu uma eficácia melhor. Assim, considerando o cenário atual em que a maioria dos dados apresentam-se em um contexto de *Big Data*, as estratégias de indexação apresentadas são úteis e podem ser adotadas para realizar um processamento eficiente e eficaz dos dados.

Para as funções de similaridade, a *Edit Distance* mostrou-se a mais ineficaz, pois: I) para as bases de dados com atributos que possuem muitos caracteres, o tempo para o cálculo de similaridade é alto; e II) os valores de *F-Measure* não foram significativos, apesar da função possuir a melhor precisão entre as demais. Para as funções *Jaro* e *Jaro Winkler*, os tempos de cálculo da similaridade são equivalentes e, no quesito eficácia, a *Jaro Winkler* apresentou resultados melhores para atributos com poucos caracteres ($BTD10$ e $RC5$). A função *Jaro* apresentou resultados melhores para as outras bases de dados, as quais contém uma quantidade de caracteres maiores ($RP5$, $UBS5$ e $UBS10$).

Para as chaves *Soundex* e *Suffix*, obtiveram-se em ambas resultados significantes, não possuindo uma diferença de *F-Measure* perceptível. Quando utilizadas as duas de maneira combinada, os resultados obtidos não foram melhores e o tempo de execução se mostrou ineficaz. Por fim, como trabalhos futuros pretende-se investigar novas técnicas de BKs, realizar testes estatísticos para fundamentar as conclusões e analisar os casos de falha das técnicas apresentadas.

Referências

Borges, E. N., Galante, R. M., and Gonçalves, M. A. (2008). Uma abordagem efetiva e eficiente para deduplicação de metadados bibliográficos de objetos digitais. In *Proceedings of the 23rd Brazilian Symposium on Databases*, pages 76–90, Campinas, Brasil.

Brizan, D. G. and Tansel, A. U. (2006). A survey of entity resolution and record linkage methodologies. *Communications of the IIMA*, pages 41–50.

Canalle, G. K., Lóscio, B. F., and Salgado, A. C. (2016). Uma estratégia para seleção de atributos relevantes no processo de resolução de entidades. In *Anais do 31º Simpósio Brasileiro de Banco de Dados*, pages 259–264, Salvador, Bahia, Brasil.

Chen, J., Jin, C., Zhang, R., and Zhou, A. (2012). A learning method for entity matching. In *Proceedings of 10th International Workshop on Quality in Databases*, China.

Christen, P. (2012a). *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, Berlin.

Christen, P. (2012b). A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering*, pages 1537–1555.

Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, pages 1183–1210.

Hajishirzi, H., Yih, W.-t., and Kolcz, A. (2010). Adaptive near-duplicate detection via similarity learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 419–426, Geneva, Switzerland.

Hernández, M. A. and Stolfo, S. J. (1995). The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD*, pages 127–138, New York, USA.

Ioannou, E., Rassadko, N., and Velegrakis, Y. (2013). On generating benchmark data for entity matching. *Journal on Data Semantics*, pages 37–56.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, pages 414–420.

Machado, R. F., Pinheiro, R. F., Machado, K. S., and Borges, E. N. (2016). Contacts deduplication in mobile devices using textual similarity and machine learning. In *Proceedings of the 22nd Brazilian Symposium on Information Systems*, page 22, Florianópolis, Santa Catarina.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys*, pages 31–88.

Odell, M. and Russell, R. (1918). The soundex coding system. *US Patents*, 1261167.

Vesdapunt, N., Bellare, K., and Dalvi, N. (2014). Crowdsourcing algorithms for entity resolution. *Proceedings of the VLDB Endowment*, pages 1071–1082.

Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research*, pages 354–359, Anaheim, California.

paper:171425

Uma Proposta de Perfil de Conjuntos de Dados na Web com Enriquecimento Semântico

Natacha Targino[1], Damires Souza[2], Ana Carolina Salgado[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife – Pernambuco – Brasil

[2]Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB)
João Pessoa – Paraíba - Brasil

{ntrsb, acs}@cin.ufpe.br, damires@ifpb.edu.br

Abstract. *The lack of metadata to describe datasets published on the Web makes their location and access by search engines or applications more difficult. Providing a dataset profile facilitates communication between publishers and consumers and also the integrated use of datasets. This paper proposes an approach that describes datasets on the Web by the generation of a semantically enriched descriptive and structural metadata profile. The enrichment occurs by means of the knowledge domain identification of the dataset at hand and a vocabulary recommendation in order to semantically reference the data. This work presents some accomplished experiments that indicate the relevance of this enrichment.*

Resumo. *A ausência de metadados para a descrição de conjuntos de dados publicados na Web dificulta sua localização e acesso por parte de mecanismos de busca ou aplicações. Prover um perfil do conjunto de dados facilita a comunicação entre publicadores e consumidores e o uso integrado dos conjuntos de dados. Este trabalho propõe uma abordagem que descreve conjuntos de dados na Web por meio da geração de um perfil de metadados descritivos e estruturais enriquecidos semanticamente. O enriquecimento ocorre por meio da identificação do domínio de conhecimento do conjunto de dados e da recomendação de vocabulários para referenciar semanticamente os dados. O trabalho apresenta alguns experimentos realizados que indicam a relevância deste enriquecimento.*

1. Introdução

A variedade de conjuntos de dados disponibilizados na Web possibilita um cenário ilimitado de informações, e combinações dessas informações podem trazer descobertas importantes. Os conjuntos de dados são publicados em diferentes distribuições (e.g., CSV[1]) que facilitam seu compartilhamento e reuso, se estiverem em formato aberto, por grupos de consumidores de dados, sejam eles humanos ou aplicações [Lóscio et al. 2017]. Entretanto,

[1] https://www.w3.org/TR/tabular-data-primer/

nem sempre os publicadores de dados e os consumidores de dados se conhecem. Portanto, é necessário fornecer algumas informações sobre os conjuntos de dados e distribuições que possam contribuir para a sua compreensão e reutilização. Isso normalmente é realizado por meio de metadados [Clarke e Harley 2014].

Os conjuntos de dados publicados geralmente não contêm descrição sobre conteúdo, pois ainda não é uma prática comum os publicadores de dados fornecerem metadados que representem corretamente o conteúdo de seus próprios conjuntos de dados [Abele 2016]. Os metadados podem ajudar na compreensão e processamento dos dados por meio da disponibilização de informações descritivas sobre o conteúdo, estrutura, qualidade e outras características dos conjuntos de dados. Em portais de dados abertos, é comum encontrar alguns metadados, entretanto, nem sempre são apresentados de forma estruturada com informações suficientes para o seu entendimento e processamento [Oliveira et al., 2016].

Como uma boa prática, o *World Wide Web Consortium* (W3C)[2] recomenda o enriquecimento de dados publicados. O enriquecimento de dados compreende um conjunto de processos que podem ser utilizados para aumentar, refinar ou melhorar dados brutos ou processados anteriormente [Lóscio et al. 2017]. Isso pode ser realizado em nível de metadados também, o que ajuda na atribuição de significado, melhora suas descrições e complementa informações que podem promover a compreensão e processamento dos dados por usuários e aplicações (consumidores).

Nesse contexto, este artigo apresenta uma abordagem que descreve conjuntos de dados abertos na Web por meio de um perfil composto de metadados descritivos e estruturais que são enriquecidos semanticamente. O enriquecimento semântico dos metadados descritivos é realizado por meio da identificação do domínio de conhecimento ao qual o conjunto de dados pertence (e.g., "saúde", "música"). Para o enriquecimento dos metadados estruturais, este trabalho provê a recomendação de vocabulários de domínio de acordo com as propriedades encontradas no conjunto de dados em questão. Como resultado, um perfil do conjunto de dados é gerado. Para avaliar a abordagem, foi desenvolvida uma aplicação e, por meio dela, alguns experimentos foram realizados com o intuito de verificar a relevância do enriquecimento tanto em termos do domínio identificado quanto do vocabulário recomendado.

Este artigo está organizado da seguinte forma: a Seção 2 introduz alguns conceitos; a Seção 3 propõe a abordagem; a Seção 4 apresenta resultados obtidos por meio da implementação da abordagem e dos experimentos realizados; a Seção 5 discute os trabalhos relacionados, e a Seção 6 expõe as conclusões e aponta trabalhos futuros.

2. Conceitos Fundamentais

Segundo a *Open Knowledge Foundation Brasil*[3], os dados estão "abertos" quando qualquer pessoa pode livremente usá-los, reutilizá-los e redistribuí-los, estando estes sujeitos, no máximo, à exigência de creditar a sua autoria e a compartilhar pela mesma licença. Dados

[2] https://www.w3.org/

[3] http://br.okfn.org/2016/04/13/uma-revolucao-de-dados-para-quem/

Abertos devem ser disseminados publicamente em formatos abertos (e.g., JSON[4]) na Web, de acordo com alguns critérios e aspectos que possibilitem sua reutilização como, por exemplo, a disponibilização de metadados [Lóscio et al. 2017]. Assim, é possível o desenvolvimento de aplicativos que consumam esses dados.

Com a grande quantidade de dados produzidos pelas mais diversas fontes como, por exemplo, organizações governamentais, surgiram iniciativas para a publicação e consumo de dados em formato aberto por meio de catálogos, que fornecem uma interface entre os publicadores e consumidores dos dados [Oliveira et al. 2016]. De acordo com Maali et al. (2014), um catálogo de dados é uma coleção curada de metadados sobre conjuntos de dados. Um conjunto de dados pode ser definido como uma coleção de dados publicados ou curados por um agente, que está disponível para acesso ou *download*.

Para facilitar o efetivo uso dos conjuntos de dados publicados, metadados descritivos (descrevem características gerais) e metadados estruturais (descrevem a estrutura interna) devem ser providos de modo a facilitar seu entendimento. Os metadados devem ser disponibilizados utilizando padrões e vocabulários recomendados, e podem ser enriquecidos sempre que se julgar necessário para a geração de descrições mais significativas. Segundo [Clarke e Harley 2014], o enriquecimento semântico pode ser realizado por meio de uma categoria adicional de metadados que melhore ainda mais a utilidade, descoberta e interoperabilidade de conteúdo.

O W3C propõe um esquema de metadados descritivos a serem empregados aos conjuntos de dados publicados. Dentre eles, *temas* ou *categorias* referem-se a um conjunto de domínios de conhecimento ao qual um conjunto de dados está vinculado [Lóscio et al. 2017]. Exemplos de domínios de conhecimento são "Música", "Dados Bibliográficos". O domínio está associado, por sua vez, aos principais literais (e.g., palavras-chave) identificados em um conjunto de dados [Abele 2016].

Para viabilizar a estruturação dos metadados de conjuntos de dados, alguns autores propuseram a criação de perfis. Abele (2016) define o perfil de um conjunto de dados como o grupo de informações descritivas e estatísticas a seu respeito. Segundo [Ellefi et al. 2014], a criação de um perfil ajuda na identificação de conjuntos de dados, podendo ser definido como um conjunto de características, tanto semânticas quanto estatísticas, que permitem sua melhor descrição.

3. Abordagem Proposta

Considerando a literatura sobre perfis de conjuntos de dados [Abele 2016; Ellefi et al. 2014; Assaf et al. 2015] e, de acordo com as indicações de boas práticas para publicação de dados na Web do W3C [Lóscio et al. 2017], este trabalho define o conceito de Perfil de Conjunto de Dados como segue.

Definição. Perfil de Conjunto de Dados (PCD). Um Perfil de Conjunto de Dados pode ser compreendido como uma anotação específica que contém metadados descritivos e estruturais referentes a um determinado conjunto de dados publicado na Web.

[4] http://www.json.org/

O PCD é organizado a partir de vocabulários recomendados. São eles: (i) DCAT[5]: descreve conjuntos de dados em catálogos de dados; (ii) VOID[6]: normalmente utilizado para expressar metadados de conjuntos de dados RDF[7] como, por exemplo, a descrição dos *links* entre os conjuntos de dados; (iii) SKOS[8]: utilizado para compartilhar e vincular sistemas de organização do conhecimento utilizando a Web Semântica; (iv) Schema[9]: vocabulário desenvolvido para ajudar mecanismos de busca a encontrar páginas.

Nesta proposta, os metadados descritivos que compõem o PCD são o *título*, *palavras-chave*, *domínio* (tema) e *vocabulário de domínio*. Os metadados estruturais indicam as propriedades identificadas no conjunto de dados e são organizados por meio de um esquema específico. Os metadados do PCD são apresentados na Tabela 1, de acordo com os vocabulários (prefixos) usados para sua formação.

Tabela 1: Metadados que compõem a estrutura do PCD

Metadado	Significado
dct:title	Título do conjunto de dados.
dcat:keyword	Palavras-chave identificadas no conjunto de dados.
dcat:theme	Domínio ou tema ao qual o conjunto de dados pertence. Este metadado é composto de duas propriedades: "rdfs:label", que identifica o nome do domínio, e o "void:uriSpace" que indica a uri desse domínio.
void:vocabulary	Vocabulário de domínio recomendado para referenciar as propriedades do conjunto de dados.
skos:inSchema	Representa o esquema das propriedades do conjunto de dados. Este metadado contém o "void:properties" que contabiliza o total de propriedades e um "void:property" para cada propriedade do documento. Cada metadado "void:property" possui o "schema:name", que indica seu nome , e o "dct:type", que mostra seu tipo.

O processo de geração do PCD com enriquecimento semântico é realizado a partir de algumas estratégias: (i) indexação dos conjuntos de dados; (ii) recomendação de vocabulário para metadados estruturais; e (iii) identificação do domínio. Cada uma dessas estratégias será descrita nas seções a seguir.

3.1. Indexação de Conjuntos de Dados

Para esta etapa são coletados arquivos referentes a conjuntos de dados. A indexação desses conjuntos de dados possibilita a realização das etapas fundamentais para a geração do PCD. Para a indexação, é utilizada a ferramenta Apache Solr[10]. Com esta ferramenta, calcula-se o peso dos termos por meio da função TF-IDF, que possibilita encontrar os termos mais

[5] https://www.w3.org/TR/vocab-dcat/

[6] https://www.w3.org/TR/void/

[7] https://www.w3.org/RDF/

[8] https://www.w3.org/TR/skos-reference/

[9] http://schema.org/

[10] http://lucene.apache.org/solr/

frequentes do arquivo. Conforme apresentado em alguns trabalhos como em Ouksili (2014) e Abele (2016), o TF-IDF pode ser utilizado para identificar os termos mais importantes de um documento e é muito utilizado nos engenhos de busca (ex: Google). Para a obtenção de termos melhorados, esses arquivos passam por um pré-processamento, onde é realizado o processo de (i) *stemming*, em que a palavra é substituída pelo seu radical para associação parcial entre variações de uma mesma palavra; e (ii) retirada de *stopwords*, que são palavras muito frequentes, mas que não possuem semântica associada [Shahi 2015]. Durante a indexação, são obtidas as seguintes informações para cada conjunto de dados: *id*: identificador único, *text*: conteúdo textual e *properties*: lista das propriedades identificadas.

3.2. Recomendação de Vocabulários

Após a indexação dos conjuntos de dados, é realizada a recomendação de vocabulários de domínio. Para isso, são selecionadas as propriedades que estão no campo "properties" do arquivo indexado e, para cada uma delas, são retornadas informações de vocabulários de domínio candidatos, como propriedades e classes. Cada propriedade recebe até quatro recomendações de termos de vocabulários que podem ser utilizados para representá-las.

Como fonte de vocabulários foi utilizado o repositório de vocabulários abertos denominado *Linked Open Vocabularies*[11] (LOV), sendo este um dos maiores catálogos aberto existentes. Por meio dele, e é possível obter vocabulários que reusam outros e suas descrições semânticas. O LOV vem sendo utilizado em alguns trabalhos, como apresentado por Ellefi et al. (2015). Após alguns testes com consultas em vários domínios, ele foi considerado suficiente para uma primeira versão da abordagem. Também é realizada a identificação do vocabulário que recebeu mais recomendações para um conjunto de dados específico, o qual é inserido no perfil gerado como o vocabulário de domínio recomendado. O Algoritmo 1 apresenta a estratégia definida.

Inicialmente, a partir do id do conjunto de dados são identificadas suas propriedades (linha 1). Cada propriedade identificada é inserida em consultas SPARQL[12] para a recuperação de informações contidas no SPARQL *endpoint* do LOV (linha 4). O objetivo é encontrar os vocabulários que possuam em suas classes as propriedades que sejam equivalentes às propriedades do conjunto de dados. Para este processo foi definida uma consulta escrita em linguagem SPARQL, que retorna a URI de até quatro propriedades de vocabulários mais populares que o *endpoint* tenha acesso. Considerando que um conjunto de dados geralmente depende de vários vocabulários para descrever seus recursos, a partir desses resultados, é possível fornecer ao usuário um resultado mais completo. O resultado de cada consulta é adicionado em uma lista, assim como os vocabulários recomendados para cada propriedade (linha 5). Estes vocabulários são inseridos em uma lista, cuja ordenação é realizada de acordo com o número de ocorrências dos vocabulários entre os resultados de todas as propriedades do documento (linha 7). São selecionados vocabulários que possuírem maior número de ocorrências entre os resultados e que estejam ativos (linha 8 – 12). Os vocabulários ativos são verificados por meio de uma consulta SPARQL no

[11] http://lov.okfn.org/dataset/lov/

[12] https://www.w3.org/TR/rdf-sparql-query/

mesmo *endpoint* indicado, que retorna um valor booleano referente à disponibilidade do vocabulário (se ele está ativo). São retornados os resultados detalhados de cada propriedade e os vocabulários melhor ranqueados na lista (linha 14).

Algoritmo 1 Recomendação de Vocabulários

Entrada: Id do conjunto de dados indexado
Saída: Lista de vocabulários recomendados para cada propriedade e os vocabulários melhor ranqueados

```
1:  propriedades =  getPropriedades(id_Conjunto_Dados);
2:  If (propriedades não nulo) Then
3:     While temProx(propriedades) Then
4:        Resultados_Conjunto_Dados = add(Executar(Consulta_Endpoint(proximo.propriedades)));
5:        Lista_Vocabulários = add(getVocabularios(Resultados_Conjunto_Dados));
6:     end While;
7:     Lista_Vocabulários.ordenar();
8:    While (temProx(Lista_Vocabulários)) Then
9:          If(Executar(Consulta_Endpoint_Disponibilidade(próximo.Lista_Vocabulários))) Then
10:           Vocabulários = add(Vocabulário);
11:         end If;
12:   end While;
13: end If;
14: return (Resultados_Conjunto_Dados, Vocabulários);
```

3.3. Identificação do Domínio

Nesta etapa, é identificado de forma automática o domínio ao qual o conjunto de dados pertence. Para isso, é utilizada como referência semântica a Ontologia do DBpedia[13]. Esta ontologia foi escolhida por conter informações sobre uma grande quantidade de domínios de conhecimento de forma bem estruturada, apresentando a taxonomia de suas classes. A estratégia de identificação do domínio é mostrada por meio do Algoritmo 2.

Essa etapa também recebe como parâmetro o id do conjunto de dados. São identificadas as palavras-chave e propriedades, a partir dos termos que possuírem os maiores valores na aplicação da função TF-IDF nos campos "text" e "properties", do conjunto de dados indexado (Linha 1 - 3). Esses termos identificados são utilizados em consultas realizadas sobre o SPARQL *endpoint* do DBpedia para a obtenção de informações de domínios de conhecimento (linha 6). Nesta consulta são identificadas as classes ou propriedades da ontologia do DBpedia, que correspondem a cada um dos termos. Em seguida (linha 9), todas as classes ou propriedades identificadas passam por um processamento onde é executada outra consulta SPARQL que percorre sua hierarquia até a identificação das classes do nó raiz, a classe retornada por essa consulta é inserida em uma lista de domínios que são referentes aos resultados provenientes de cada termo consultado. É retornado como domínio a classe da hierarquia de domínios que apresente maior número de ocorrências entre os resultados das consultas (linha 11 – 13). Na seção seguinte, um exemplo será apresentado.

[13] http://wiki.dbpedia.org/

Algoritmo 2 Identificação do Domínio

Entrada: Id do conjunto de dados indexado
Saída: Os domínios melhor ranqueados

```
1:  palavrasChave = getPalavrasChave(id_Conjunto_Dados);
2:  propriedades = getPropriedades(id_Conjunto_Dados);
3:  termos = palavrasChave + propriedades;
4:  If (termos não nulo) Then
5:     While temProx(termos) Then
6:        Resultados_Ontologia = add(Executar(Consulta_Ontologia(próximo. termos));
7:     end While;
8:     While temProx(Resultados_Ontologia) Then
9:      Lista_Domínios = add(Executar(Consulta_Hierarquia_Onto(próximo.Resultados_Ontologia)));
10:    end While;
11:   Lista_Domínios.ordenar();
12:  end If;
13:  return (Lista_Domínios.getPrimeiro());
```

4. Implementação e Experimentos Realizados

Esta seção apresenta alguns resultados da implementação e dos experimentos realizados.

4.1. Implementação

A abordagem proposta foi desenvolvida na linguagem Java como uma aplicação baseada na Web com uma interface amigável, o que facilita a realização das etapas de indexação de conjuntos de dados, recomendação de vocabulários, identificação do domínio e geração do PCD. Nesta versão foram considerados conjuntos de dados que estejam em formato JSON e em inglês, principal idioma de publicação de dados na Web. Para a utilização de outros idiomas, é necessário configurar a etapa de indexação (o *stemming* e a retirada de *stopwords)*. Na recomendação de vocabulários, é necessário o acesso a vocabulários que forneçam o nome de seus recursos na língua desejada. Já na identificação do domínio, foi utilizado o DBpedia que fornece a nomeação dos seus recursos em diversos idiomas, inclusive em português.

Na Figura 1 é exibida a página principal da aplicação, que possibilita a execução das estratégias da abordagem, gerando resultados intermediários e finais. Como exemplo, considere um conjunto de dados sobre música, intitulado "Wanderlust". Primeiramente esse conjunto de dados é indexado. Depois, é realizada a identificação dos vocabulários recomendados. Neste caso, foi recomendado como vocabulário geral o *The Music Ontology* [14]. Durante a execução da identificação do domínio foi identificado como domínio geral a classe do DBpedia *Musical Work*[15]. A partir deste domínio são extraídos o seu nome e a URI, que possibilita ao usuário o acesso a maiores informações.

[14] http://purl.org/ontology/mo/

[15] http://dbpedia.org/ontology/MusicalWork

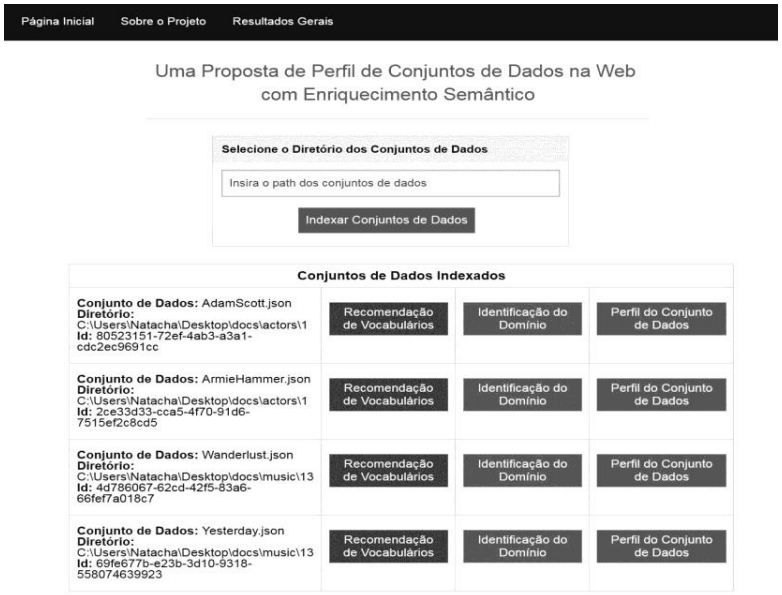


Figura 1. Página com Etapa inicial de Indexação.

Para a identificação do título, caso o conjunto de dados não possua alguma propriedade referente, será retornado seu nome de arquivo. No exemplo, foi encontrada a propriedade *title:Wanderlust*. Já as palavras-chave são identificadas com a aplicação da função TF-IDF no campo *text* do conjunto de dados indexado. A descrição do esquema do conjunto de dados é composta dos nomes e tipos das propriedades identificadas.

Na Figura 2 é apresentado o perfil que é gerado no formato RDF, sendo este o formato recomendado pelo W3C para a geração de metadados. São exibidos os prefixos dos vocabulários utilizados e, em seguida, é possível observar os metadados (definidos na Seção 3) que receberam os valores encontrados durante a execução da aplicação.

```
@prefix skos:   <http://www.w3.org/2004/02/skos/core# >.
@prefix void:   <http://rdfs.org/ns/void# >.
@prefix dct:    <http://purl.org/dc/terms/> .
@prefix dcat:   <http://www.w3.org/ns/dcat#:> .
@prefix  rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix  schema: <http://schema.org/>.

[]  a      dcat:Dataset ;
    dct:title "Wanderlust" ;
    dcat:keyword "singer" , "artist" , "name" ;
    dcat:theme
          [ rdfs:label "Musical Work";
            void:uriSpace "http://dbpedia.org/ontology/MusicalWork"
          ] ;
    void:vocabulary <http://purl.org/ontology/mo/> ;
    skos:inScheme
          [ void:properties "3" ;
            void:property
                  [ schema:name "id" ;
                    dct:type "string"
                  ] ;
            void:property
                  [ schema:name "title" ;
                    dct:type "string"
                  ];
            void:property
                  [ schema:name "name" ;
                    dct:type "string"
                  ]
          ] .
```

Figura 2. Perfil exemplo gerado no formato RDF/Turtle

4.2. Experimentos Realizados

Alguns experimentos foram realizados com o intuito de avaliar a abordagem proposta. Dois objetivos foram identificados: (i) avaliar a recomendação de vocabulários de domínio para o conjunto de dados, e (ii) avaliar o domínio identificado de acordo com o conjunto de dados. Para cada um dos objetivos, foram definidos *gold standards* associados para cada conjunto de dados com os resultados esperados. Foram selecionados 100 conjuntos de dados no formato JSON, escritos em língua inglesa e identificados como pertencentes aos domínios de "biografia de atores" e "música". Os conjuntos de dados são provenientes do Internet Movie Database (IMDB)[16], e da enciclopédia aberta sobre música MusicBrainz[17]. Eles se encontram publicados sem especificações de metadados descritivos e estruturais.

Os resultados obtidos com a geração do PCD foram comparados com os *gold standards* estabelecidos, e então foram calculadas as métricas de Precisão, Cobertura e F-Measure, cujas fórmulas são apresentadas a seguir:

$$\text{Precisão} = \frac{\#\text{ResultadosRelevantes}}{\#\text{ResultadosRetornados}}$$

$$\text{Cobertura} = \frac{\#\text{ResultadosRelevantes}}{\#\text{ResultadosEsperados}}$$

$$F - Measure = \frac{(2 * \text{Pr}ecisão * Cobertura)}{(\text{Pr}ecisão + Cobertura)}$$

Nas fórmulas apresentadas, #ResultadosRelevantes é a quantidade de resultados retornados considerados relevantes de acordo com o *gold stan*dard, #ResultadosEsperados é o total de resultados que poderiam ser retornados definidos pelos *gold standards*, e #ResultadosRetornados é o número total de todos os resultados retornados.

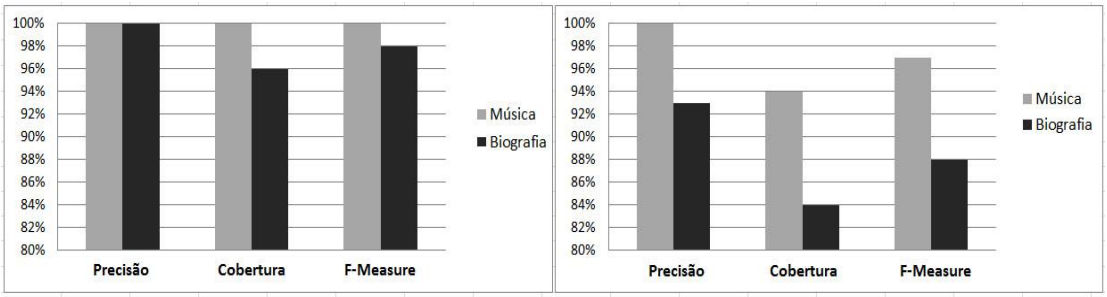
Conforme mostra a Figura 3(a), para a recomendação de vocabulários de domínio, foi observado que os conjuntos de dados dos dois domínios obtiveram resultados semelhantes. Para todos os conjuntos de dados do domínio de música foi recomendado o vocabulário definido como o *gold standard* e, para os conjuntos de dados do domínio de biografia de atores, apenas dois deles não obtiveram em sua recomendação o vocabulário definido como o *gold standard*. Dessa forma, nos conjuntos de dados do domínio de música, foram obtidos valores de precisão, cobertura e F-measure de 100% para cada uma dessas métricas. E para os conjuntos de dados do domínio de biografia, foram obtidos valores de precisão de 100%, cobertura de 96% e F-measure de 98%.

De acordo com a Figura 3(b), na identificação do domínio, os conjuntos de dados sobre música também alcançaram melhores resultados. Apenas três deles não obtiveram em seus resultados o domínio definido como o *gold standard*. Já nos conjuntos de dados sobre a biografia de atores, por possuírem termos mais diversos, em oito deles não foi

[16] http://www.imdb.com/

[17] https://musicbrainz.org/

identificado o domínio definido como *gold standard*. Nos conjuntos de dados do domínio de música, foram obtidos valores de precisão de 100%, uma cobertura de 94% e F-measure de 97%. Já para os conjuntos de dados sobre biografia, foram obtidos valores de precisão de 93%, cobertura de 84% e F-measure de 88%.



(a) Recomendação de Vocabulários (b) Identificação do Domínio

Figura 3 – Resultados Métricas

Com os resultados das métricas calculadas foi observado que a especialidade dos conjuntos de dados reflete diretamente nos resultados. Por exemplo, nos conjuntos de dados sobre música, quase todos obtiveram em seus resultados o ideal estabelecido. Isso pode ser atribuído à semântica desses conjuntos de dados, que é utilizada nas palavras-chave e interferem diretamente nos resultados. Quando o conjunto de dados é de uma área mais específica, suas palavras são bastante especializadas, o que torna mais provável alcançar os resultados esperados.

5. Trabalhos Relacionados

Considerando as estratégias definidas, alguns trabalhos relacionados foram identificados. Em relação à recomendação de vocabulários, Ellefi et al. (2015) propuseram um sistema de recomendação de vocabulários baseado no repositório LOV. Schaible et al. (2013) apresentaram uma abordagem que identifica as classes e propriedades de vocabulários que são mais utilizadas na nuvem *Linking Open Data* - LOD[18], para a representação dos dados em RDF. Em nosso trabalho, é apresentada a recomendação de vocabulários para cada propriedade do conjunto de dados de forma automática, recomendando apenas os vocabulários que estão ativos, de forma diferente dos trabalhos citados. Com essa recomendação é possível referenciar os metadados estruturais, possibilitando a conversão de conjuntos de dados de vários formatos semiestruturados ou estruturados para o formato RDF, promovendo a reutilização de metadados e integração dos dados.

Com relação à identificação do domínio, o trabalho de Ouksili et al. (2014) apresenta uma abordagem que possibilita a identificação de temas de um determinado conjunto de dados RDF. Nesta abordagem é utilizada uma combinação de critérios estruturais e semânticos para o agrupamento (*clustering*) dos grafos, onde cada agrupamento corresponde a um tema. Em nosso trabalho, as palavras-chave de um conjunto de dados são extraídas para identificar o domínio, por meio de um mecanismo que possui

[18] http://lod-cloud.net/

informações sobre uma grande quantidade de domínios de conhecimento de forma bem estruturada.

Em termos de geração de perfis, como exemplo, o trabalho apresentado por Assaf et al. (2015) propõe o Roomba, uma abordagem automática e escalável para extração, validação, correção e geração de perfil de conjuntos de dados. A abordagem proposta foi avaliada diante de um conjunto de portais de dados abertos. Outro trabalho foi apresentado por Abele (2016), onde é proposta uma abordagem para a descrição detalhada dos conjuntos de dados, utilizando metadados para prover informações gerais sobre os mesmos como, por exemplo, descrição, data de atualização e informações de licença. Diferentemente desses trabalhos, propomos a geração de um perfil com metadados descritivos e estruturais enriquecidos para conjuntos de dados que estejam em qualquer formato semiestruturado ou estruturado, e onde também não há necessidade da existência prévia de metadados descritivos e/ou estruturais. Além do perfil de conjunto de dados gerado, o usuário tem acesso à recomendação de vocabulários do domínio para cada propriedade existente no conjunto de dados em questão.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma abordagem para criação de perfil de conjunto de dados na Web com enriquecimento semântico. A abordagem proposta inclui as seguintes etapas: (i) indexação do conjunto de dados; (ii) recomendação de vocabulários de domínio para os metadados estruturais; (iii) identificação do domínio de conhecimento ao qual o conjunto de dados pertence e (iv) geração do perfil do conjunto de dados composto por metadados descritivos e estruturais. Com a implementação da abordagem, foi gerada uma aplicação Web que pode ser utilizada por produtores que desejem disponibilizar junto ao conjunto de dados um perfil com metadados descritivos e estruturais. Também é possível ter acesso a vocabulários para a conversão de seus dados para formato RDF. Consumidores de dados podem gerar de forma automática o perfil do conjunto de dados sem a necessidade de terem conhecimento sobre esses dados.

Experimentos realizados demonstraram que a abordagem é promissora no que diz respeito à geração do perfil do conjunto de dados e, em especial, ao enriquecimento semântico. Esse enriquecimento é obtido por meio da identificação do domínio de conhecimento do conjunto de dados e da recomendação de vocabulários para referenciar semanticamente os dados. Nos experimentos foram selecionados conjuntos de dados provenientes de domínios de dados distintos. Os resultados apresentaram valores de precisão e cobertura significativos, mas também foi observada a necessidade da realização de experimentos mais aprofundados, com usuários especialistas e conjuntos de dados pertencentes a uma maior variedade de domínios.

Como trabalhos futuros, além de novos experimentos, pretende-se identificar um único vocabulário para cada propriedade do conjunto de dados que possa ser incluído nos metadados estruturais do perfil gerado, promovendo maior facilidade para entendimento e reutilização do conjunto de dados. Também é pretendido incluir outros catálogos de vocabulários como fontes adicionais para a etapa de recomendação de vocabulários e permitir o uso de outros formatos de conjunto de dados, além do JSON.

Referências

Abele, A. (2016) "Linked Data Profiling: Identifying the Domain of Datasets Based on Data Content and Metadata", In: 25th International Conference Companion on World Wide Web. Canada, p. 287-291.

Assaf, A., Troncy, R. and Senart, A. (2015) "Roomba: An extensible framework to validate and build dataset profiles", In: 24th International Conference on World Wide Web, Italy, p. 159-162.

Clarke, M. and Harley, P. (2014) "How smart is your content? Using semantic enrichment to improve your user experience and your bottom line", Science Editor, v. 37, n. 2, p. 40–44.

Ellefi, M. B., Bellahsene, Z., Scharffe, F. and Todorov, K. (2014) "Towards semantic dataset profiling", In: International Workshop on Dataset Profiling & Federated Search for Linked Data co-located with the 11th Extended Semantic Web Conference. Greece.

Ellefi, M. B., Bellahsene, Z. and Todorov, K. (2015) "Datavore: a vocabulary recommender tool assisting Linked Data modeling", In: 14th International Semantic Web Conference Posters & Demonstrations Track a Track. United States.

Lóscio, B. F., Burle, C., Calegari, N. (2017) "Data on the web best practices. The World Wide Web Consortium", https://www.w3.org/TR/dwbp/ Último Acesso: 20 de maio de 2017.

Maali, F., Erickson, J., and Archer, P. (2014). "Data catalog vocabulary (DCAT). W3C recommendation, The World Wide Web Consortium", https://www.w3.org/TR/vocab-dcat/ Último Acesso: 20 de maio de 2017.

Oliveira, M. I. S., Oliveira, L. A., Lima, G. F. B. and Lóscio, B. F. (2016). "Enabling a unified view of open data catalogs", In: 18th International Conference on Enterprise Information Systems (ICEIS). Italy, p. 230-239.

Ouksili, H., Kedad, Z. and Lopes, S. (2014) "Theme Identification in RDF Graphs", In: 4th International Conference on Model and Data Engineering (MEDI). Cyprus, p. 321-329

Schaible, J., Gottron, T., Scheglmann, S. and Scherp, A. (2013) "LOVER: support for modeling data using linked open vocabularies", In: EDBT/ICDT 2013 Joint Conference. Italy, p. 89–92.

Shahi, D. (2015) Apache Solr: a practical approach to enterprise search. Apress, Primeira Edição, p. 82–85. ISBN: 978-1-4842-1071-0

paper:170758

Workload-Aware RDF Partitioning and SPARQL Query Caching for Massive RDF Graphs stored in NoSQL Databases

Luiz Henrique Zambom Santana , Ronaldo dos Santos Mello

[1] Federal University of Santa Catarina (UFSC)
Florianópolis - SC - Brazil

luiz.santana@posgrad.ufsc.br, r.mello@ufsc.br

Abstract. *Governments, corporations, startups, open data initiatives and other organizations are increasingly considering RDF and SPARQL in a broad range of information management scenarios. To reduce SPARQL querying times has been the main issue for virtually all the recent RDF triplestores, yet SPARQL caching techniques have not been broadly considered. In this paper we present Rendezvous, a middleware that addresses workload-adaptive management of large RDF graphs with a caching strategy for SPARQL query results. Our middleware provides a novel RDF data partitioning approach based on a fragmentation strategy that maps RDF data into multiple NoSQL databases. The focus of this paper is also on Rendezvous caching, which can reduce average response time by up to an order of magnitude. Our experimental evaluation shows that the approach is promising, outperforming a recent key/value-based caching baseline.*

1. Introduction

RDF is a standardized data model that - along with other technologies like OWL, RDFS, and SPARQL - grounds the vision of Semantic Web as an initiative to foment interlinked machine-processable information [Berners-Lee et al. 2001]. In the last decade, RDF has been increasingly used in a wide range of data management scenarios (*e.g.*, data integration, search-engine optimization, data representation, information extraction) as a resource for better understanding of complex real-world entities and their relationship. However, the current scale of data intensive applications (*e.g.*, Smart Cities, Sensor Networks, eHealth, IoT) - all of them very attractive for the Semantic Web vision -, prevents the efficient usage of existing RDF storage systems operating on a single node. In fact, such a kind of system is becoming quite a performance bottleneck giving the actual generation of massive RDF data which goes beyond its processing capacities. It raises the need for innovations in the frontier of Big Data and Semantic Web research fields.

This paper presents *Rendezvous*, a middleware that includes a novel RDF data partitioning approach with a fragmentation strategy that maps pieces of an RDF graph into NoSQL databases with different data models. We consider a workload-aware partitioning approach and into account the ideas from Estocada [Bugiotti et al. 2015] to develop a multiformat RDF storage based on its query workload to decide which NoSQL data model is the best fit for each incoming RDF fragment. The main contributions of this work are: *(i)* a mapping of RDF data to the columnar, document NoSQL and key/value data models [Sadalage and Fowler 2012]; *(ii)* a complex caching mechanism to store query results

both in each server and remotely in a key/value database; *(iii)* a workload-aware partitioner based on the current graph structure and, mainly, in the typical application workload; and, *(iv)* an experimental evaluation that compares our approach against a baseline (ScalaRDF [Hu et al. 2016]) by considering Apache Cassandra, MongoDB and Redis. Our high point is to process queries over large RDF graphs stored on multiple NoSQL servers with zero or a subtle amount data joining cost. An experimental evaluation shows that our middleware scales well, being able to process huge RDF datasets efficiently.

The rest of the paper is organized as follows. We give a brief background overview and discuss related works in Section 2. In Section 3 we detail the *Rendezvous* approach. We report our experimental evaluation in Section 4 and conclude the paper in Section 5.

2. Background and Related Work

The most important pillar of this work is the Semantic Web, as envisioned by Tim Berners-Lee in 2001 [Berners-Lee et al. 2001]. The Semantic Web offers, as practical value, the development of applications that can handle complex human queries based not only on simple matches of raw data but also on its meaning. When Semantic Web was presented, the exponential increase of information quantity could not be foreseen by most of the specialists, but the need for data integration was already argued as one of its fundamental purposes. Thus, in the recent years, the effort of developing the Semantic Web was harvested mainly in the form of well-established standards for expressing shared meaning, defined by WWW Consortium (W3C), like Resource Description Framework (RDF) and the Simple Protocol and RDF Query Language (SPARQL).

RDF is expressed by triples that define a relationship between two resources. RDF triples can be modeled as graphs, where the resources, called *subject* and *object*, are vertexes, and the relationship, called *predicate*, is a directed edge from the subject to the object. For instance, we can define a predicate *:owns* between two resources: *:person* (subject) and *:car* (object). SPARQL is a query language for searching and retrieving RDF information. A query statement in SPARQL consists of triple patterns, conjunctions, disjunctions, and optional patterns. The *triple pattern* defines the RDF subject, predicate and object to be searched, *conjunctions and disjunctions* express the intended relations between the searched resources, and the *optional patterns* combine two graph models. Moreover, sets of triple patterns define *Basic Graph Patterns (BGP)*, being each BGP a function that transforms the RDF dataset into mapping sets. Finally, these mapping sets are the answer to an SPARQL query in the form of RDF triples.

Traditionally, the SPARQL queries can be categorized into *star*, *chain* and *complex* queries [Gallego et al. 2011]. These shapes depend on the location of variables in triple patterns which can influence the query performance. The diameter of an SPARQL BGP is defined as the largest connected sequence of triple patterns, disregarding the edge direction. The *star* pattern has a diameter of one and is characterized by subject-subject joins within triple patterns as the join variable is located on the subject. *Chain* patterns are very common in graph querying (*e.g.*, friend-of-a-friend), being formed by object-subject joins, *i.e.*, the join variable is in the subject location in one pattern and on object location in the other one. *Complex* patterns are combinations of several star patterns connected by typically a single pattern. Other query structures are compositions of these major patterns.

The use of RDF to describe semantic data has seen a dramatic increase over the

last years, making RDF data almost ubiquitous (see LODStats[1]). As a consequence, recent surveys have highlighted the joint usage of RDF and NoSQL by Big Data applications [Ma et al. 2016]. NoSQL databases, as defined, in a consensual way, by Sadalage and Fowler [Sadalage and Fowler 2012], means database systems that use more than one storage mechanism, including new types not compatible with the traditional relational databases . NoSQL databases are usually organized into the following categories w.r.t. their data models in *key-value*, *document*, *columnar* and *graph*.

In this paper, we focus on the document, columnar, key/value NoSQL data models. The document data model is suitable to store semistructured data in one of the formats considered by current computational systems (*e.g.*, XML and JSON). *MongoDB* is the most popular NoSQL document database, being already tested for storing RDF as JSON documents [Mulay and Kumar 2012]. The columnar data model aims at storing data that do not respect a rigid schema but belong to the same domain of data, *i.e.*, data instances that usually hold a standard set of properties (columns), but may have a different number of columns. Apache Cassandra is the most popular NoSQL columnar database, and it is also used for RDF storage in the CumulusRDF approach [Ma et al. 2016]. Finally, the Key/value databases are also helping the RDF solutions to scale, for instance, *ScalaRDF* [Hu et al. 2016] is a triple store that stands out for persisting data on the key/-value database Redis as a distributed memory storage to speed up query performance.

There are many works proposed on polyglot NoSQL databases and scalable RDF data management [Ma et al. 2016], denoting that these are very hot topics. Among polyglot NoSQL databases the *Estocada* [Bugiotti et al. 2015], stands out as an architecture for handling highly heterogeneous datasets that provide a middleware capable of hosting multiple data sets as a set of potentially overlapping fragments, distributing the various fragments of each dataset across different stores, including SQL and NoSQL databases. *Rendezvous* goes beyond *Estocada* by focusing the dataset type in the RDF format and the storage exclusively in NoSQL storages. We also propose both a fragment expansion and a partition scheme for avoiding joins between data coming from different NoSQL nodes that can reduce the response time, especially in queries touching a large amount of data. ScalaRDF [Hu et al. 2016] is a important representative of scalable RDF data management, by presenting a distributed in-memory RDF triple store that uses Redis in a fault-tolerant store and query mechanism. *Rendezvous* polyglot capabilities for data storage - we use document, columnar and key/value databases -, along with the *n-hop* fragmentation scheme and the workload-awareness and complex caching solution, makes our approach more suitable to dynamic query workload and offers interactive querying response time over large RDF graphs.

3. Rendezvous

Rendezvous is a middleware for partitioning and storing RDF data in multiple NoSQL database nodes. Although its focus is on storing RDF triples, its inspiration comes from *Estocada* (see Section 2), which argues that a mixed-model layer, relying on a set of diverse and heterogeneous data stores, can provide performance advantages for the applications using this layer. Figure 1 presents an overview of *Rendezvous* architecture. A *RDF-based application* issues storing or querying requests to *Rendezvous*, that is nor-

[1]http://stats.lod2.eu/

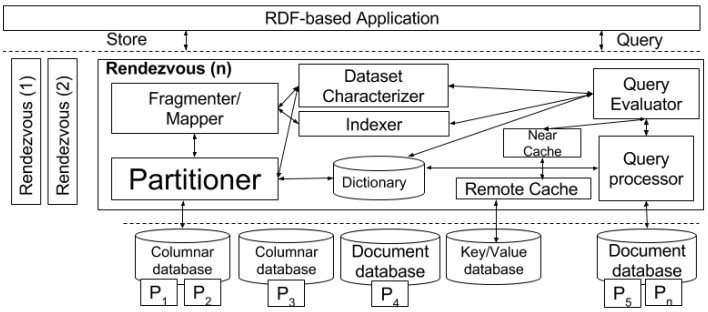


Figure 1. Rendezvous Architecture

mally deployed into multiple dedicated physical node. Thus, one could integrate several applications on top of *Rendezvous* using RDF as a common data model. Another idea that we borrowed from *Estocada* is the development of a fragment-based storage which is entirely transparent to the client applications. The data flow in *Rendezvous* is most of the time in the format of fragments. As defined in the following, a fragment is essentially a part of the RDF graph to be stored and retrieved into/from NoSQL databases. A **RDF Fragment** is a set $F_{RDFi} = \{t_{RDF}\}$ of triples whose content may overlap with other fragment F_{RDFj}, where t_{RDF} is a RDF triple $t_{RDF} = (s, p, o)$ where $t_{RDF}.s$ is the subject, $t_{RDF}.p$ is the predicate and $t_{RDF}.o$ is the object.

When an *RDF-based Application* issues a store request for a triple to the *Fragmenter/Mapper* component, *Rendezvous* expands this triple to a fragment F_{RDFi} and maps F_{RDFi} to the target NoSQL database(s). This process (see Section 3.1) is performed by the *Dataset Characterizer*, the main component of our middleware. During a triple storage, it decides on translating F_{RDFi} to a columnar or document data (or both) according to the usual query workload, and indexes it in the *Indexer*. Once F_{RDFi} is created, the *Partitioner* register this fragment into the *Dictionary* repository - designed as a *in-memory hashmap* - and stores it in the NoSQL databases (see Section 3.1). When an *RDF-based Application* issues a query request, the *Query Evaluator* component decomposes this query into star-shaped and chain-shaped subqueries and reports to the *Dataset Characterizer* about these queries (see Section 3.2). In the following, the *Query Evaluator* verifies, with the aid of the *Dictionary*, the partitions in which the triples for the query are potentially located. Based on this information, the *Query Processor* translates the SPARQL query to columnar and/or document database queries. Finally, again with the aid of the *Dictionary*, the component *Query Evaluator* translates back the query results to RDF triples and returns to the *RDF-based Application*.

The primary purpose of *Rendezvous* is to store large RDF graphs. In such scenario, the number of RDF triples can easily surpass the performance capacity (*e.g.*, disk, memory, CPU) of a single server. When it occurs, *Rendezvous* distributes the RDF fragments among potentially many NoSQL nodes. Notice that a fragment is our smallest grain of distribution, *i.e.*, during the partitioning process we deal with fragments instead of triples. In *Rendezvous*, as defined in the following, an RDF partition is a set of fragments stored in the same physical NoSQL node, and a fragment can be replicated in multiple partitions. A **RDF Partition** P_m of an RDF graph G, such that G $\subseteq P_1 \cup P_2 \cup ...P_n$, is a set of RDF fragments $P_m = \{F_{RDFi}\}$, being not required that $P_m \cap P_t = \emptyset$, for m $\neq$ t. Nevertheless,

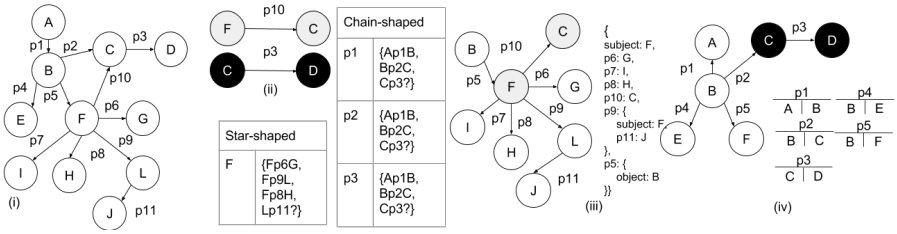


Figure 2. Fragmentation process

a query can eventually access data in multiple partitions, forcing *Rendezvous* to join the data from different NoSQL nodes. Since a join operation is very costly, we try to avoid join processes by replicating fragments that are potentially part of a join. As defined in the following, if the *typical workload* for a fragment spans more than one partition, our partition scheme replicates the boundary fragments of the partition. Given SP = $\{ P_1, P_2, ..., P_n\}$ the set of RDF partitions, the **Partition Boundary** B_{P_i} of a partition $P_i \subset SP$ is the set of RDF fragments $B_{P_i} = Fb_{P_1} \cup Fb_{P_2} ... \cup Fb_{P_n}$, where $Fb_{P_k} \subset P_k$ for any k. Each $Fb_{P_i} \in B_{P_i}$ has one or more RDF triples $t_i F_{P_i} = (s_i, p_i, o_i)$ where $o_i = s_j$ been s_j subject of any other triple $t_j F_{P_j}$ of partition P_j where $t_j F_{P_j} = (s_j, p_j, o_j)$.

On RDF indexing, a traditional approach is to build indexes for the full set of permutations of each triple component (subject, predicate, object). Although this method has been designed to accelerate joins by some orders of magnitude, the overhead with the large index space limits its scalability and makes it heavyweight. Hence, we decided to develop a hashmap index with subject and object keys following the patterns *S-PO* and *O-PS* [Weiss et al. 2008]. In *Rendezvous*, the component *Indexer* is responsible to manage these indexes, being accessed in two situations: *(i)* during the fragment creation, explained in the Section 3.1 and *(ii)* to process queries that inform only the object or the subject, but not the predicate.

3.1. Storing: Fragmentation, Mapping and Partition

The data mapping is the most prominent *Rendezvous* task during a *storing* process. As stated before, our proposal is based on RDF fragmentation, so we first define our fragmentation strategy and the supported types of fragments. An RDF fragment is created when a new RDF triple to be stored (called *core triple*) is expanded with all of its neighbors according to a *n-hop replication horizon*. The parameter *n* is the number of predicates from the *core triple* that our fragment expansion process considers. This parameter is controlled by the *Dataset Characterizer* component, being defined as the 75th percentile of the total number of joins of the typical workload over the *core triple* elements. Figure 2 (i) shows an RDF graph, Figure 2 (ii) represents two core triples and Figure 2 (iii and iv) is the 1-hop fragment for both core triples.

The main reasoning for our fragmentation strategy in several RDF fragments is to maximize the NoSQL query capability and to minimize the cost of joining data in the *Rendezvous* node. As presented in Figure 2, if two core triples (*F p10 C* and *C p3 D*) have to be stored in the RDF graph of Figure 2 (i), we invoke the component *Dataset Characterizer*, responsible for keeping track of the typical query workload. It manages two *in-memory hashmaps*: one for the *star-shaped* queries, where the key is a subject or a object (for subject-subject and object-object joins, respectively - see Section

2), and another for the *chain-shaped* queries, where the key is the predicate. The *Dataset Characterizer* also decides the size of the fragment (*n*-hop) for the star-shaped case (*n* is the longest chain among the queries in the hashmap), and for the chain-shaped case (*n* is the size of the biggest query in the hashmap). In the Figure 2, the n-hop for star-shaped fragments with F as the subject is 2, and for the predicates, p1, p2, and p3 is also 2. If the subject of the new triple is defined in the star-shaped hashmap, the triple is converted into a document fragment, or an existing document fragment is updated in the NoSQL document database (Figure 2 (iii)). Otherwise, if the predicate of the new triple is defined in the chain-shaped hashmap, we create and store a columnar fragment in the columnar NoSQL database, or we update the subject and/or object of this triple in an existing column family (Figure 2 (iv)).

The same fragment can be mapped to both document and columnar fragments if the subject or object of the core triple of this fragment is in the star-shaped hashmap and its predicate is in the chain-shaped at the same time. If an RDF fragment is translated to a *document fragment* we have a mapping to a JSON document[2], which is the standard format for NoSQL document databases. If an RDF fragment is translated to a *columnar fragment*, we have a mapping to a column family which is the typical logical structuring for NoSQL column databases. The definition of these two types of a fragment, as well as the mapping of RDF fragments to them, are given in the following.

Document Fragment is a tuple $f_{doc} = (k_d, A)$ where $f_{doc}.k_d$ is the JSON document key and $f_{doc}.A$ is a set of attributes (key-value pairs) $f_{doc}.A = \{(k_\alpha : v)\}$, being k_α the attribute key and v a value whose domain can be atomic, a list, a set, or a tuple. The **RDF-to-Document Fragment Mapping** of F_{RDFi} to f_{doc} proceeds as follows: (i) given the core triple $t_{core} \in F_{RDFi}$: $f_{doc}.k_d \leftarrow t_{core}.s$; (ii) for each $t_{core}.p \in F_{RDFi}$ ($t_{core}.p$ is a t_{core} outgoing edge $\wedge$ F_{RDFi} IS a 1-hop fragment): $a_j.k_\alpha \leftarrow t_{core}.p$ and $a_j.v \leftarrow t_{core}.o$, being $a_j \in f_{doc}.A$ and $t_{core}.o$ reached from $t_{core}.p$; (iii) for each $t_{core}.p \in F_{RDFi}$ ($t_{core}.p$ is a t_{core} outgoing edge $\wedge$ F_{RDFi} IS NOT a 1-hop fragment): $a_j.k_\alpha \leftarrow t_{core}.p$ and $a_j.v$ is an inner document generated from the mapping of t_j to a document fragment recursively, being t_j an RDF triple where $t_j.s$ is the object reached from $t_{core}.p$. In short, the core triple t_{core} in the RDF fragment F_{RDFi} is mapped to document whose key is $t_{core}.s$, and each outgoing predicate from the subject becomes a document attribute with a key $t_{core}.p$. If F_{RDFi} is 1-hop, the attribute value of each outgoing predicate is the object $t_{core}.o$ reached from it. Otherwise, the predicate value is an inner document that maintains the target object as the inner document key, and its outgoing predicates as attributes. If any of these outgoing predicates is, in turn, an n-hop, n>1, the generation of other inner documents proceeds recursively. One example is shown in Figure 2 (iii).

Columnar Fragment is a tuple $f_{cf} = (k_cf, C)$ where $f_{cf}.k_cf$ is the name (key) of the column family and $f_{cf}.C$ is a set of columns (key-value pairs) $f_{cf}.C = \{(n_c : v)\}$, being n_c the column name (or column key) and v an atomic value. The **RDF-to-Columnar Fragment Mapping** of F_{RDFi} to columnar fragments proceeds as follows: for each predicate $t_k.p$ of a triple $t_k \in F_{RDFi}$: generate a columnar fragment $f_{cfi} = (k_cfi, \{c1, c2\})$ such that $f_{cf}.k_cfi \leftarrow t_k.p$, $f_{cf}.C.c1 \leftarrow t_k.s$ and $f_{cf}.C.c2 \leftarrow t_k.o$. A columnar fragment maintains a triple predicate of an RDF fragment. For each predicate we define a column family - named with the predicate label - and two columns to hold

[2]https://www.w3.org/TR/json-ld/

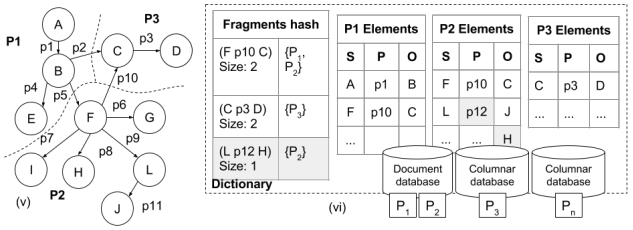


Figure 3. Fragment partitioning

the triple subject and triple object names. One example is shown in Figure 2 (iv). The decision for such a mapping strategy allows that, given a predicate, we query the columnar database using the subject and/or object as a filter, which is usually the scenario for chain-shaped queries.

If the number of RDF triples exceeds the performance capacity (*e.g.*, disk, memory, CPU) of a single server, *Rendezvous* distributes the RDF fragments among potentially many NoSQL nodes. In general, an RDF graph can be partitioned vertically, horizontally, and hash-based [Ma et al. 2016]. In a hash-based partitioning, a triple is placed into a partition based on the result of a hashing algorithm. The benefit of this approach is to distribute data evenly among the nodes, but several cross partition joins can be necessary to respond to a query request. To obtain better performance, the *Rendezvous* fragmentation process leads to horizontal partitioning [Mulay and Kumar 2012] for storage in a document database to deal with star-shaped queries, and vertical partitioning [Abadi et al. 2009] for a columnar database to tackle chain-shaped queries.

In *Rendezvous* the most basic element of partition is an RDF fragment. Given the RDF graph of Figure 2 (iii) the fragments (iv) were stored respectively in the document partition P_2 and the columnar partition P_3, and each fragment location is stored in the *Dictionary* repository, as presented in Figure 3. This repository maintains a hashmap with a key based on the core triple and its size and the partitions where this fragment is stored as the value. We also use three hash sets for each partition to keep track of the RDF elements (subjects, predicates, and objects) stored in this partition, so during a query request, we can avoid accessing unnecessarily partitions that cannot answer this query. If a *Rendezvous* node manages more than one partition of a NoSQL type, in face of a core triple, we have to decide which is the best partition to store its fragments. For doing so, the information provided by the *Dataset Characterizer* is used to find the typical workload for the triples present in the fragment generated by the core triple. With this information, we can query the partition sets in the *Dictionary* to verify in which partition this fragment can be more useful, in the sense that the joins outside the fragment can be answered within a single partition. For instance, for a fragment with core triple *L p12 H* the best fit would be P_2, since all the typical workload queries ($\{$*Fp6G, Fp9L, Fp8H, Lp11?*$\}$) can be answered in this single partition.

Another point to highlight is that our workload-awareness is limited since the queries issued against *Rendezvous* are dynamic. Thus, in order to increase the fraction of queries that can be processed by accessing only one partition, we have two options: *(i)* to increase the n-hop replication horizon, which would ultimately result in a high amount of replicated data, or *(ii)* to choose a reasonable replication extent, *e.g.*, n = 2, and

add systematic replication fragments in the boundary of each partition. We advocate the second option since, as we show in Section 4, it generates a small storage overhead. In Figure 3, the boundary replication (with size n=2) is presented between partitions the P_1, P_2, and P_3. For instance, the fragment with core triple F p10 C and size 2 is in partitions P_1 and P_2, and the elements F, p10, and C are placed in more than one partition.

3.2. Querying Decomposition and Caching

Another important task accomplished by *Rendezvous* is the query decomposition. The input SPARQL queries are analyzed by the *Query Evaluator*. It, in turn, classifies a query into *simple*, *star-shaped*, *chain-shaped* or *complex*. A query is called *simple* if it does not involves a join in any triple component. If the query is classified as *complex*, it can be decomposed into *star-shaped* and/or *chain-shaped* subqueries. The *Query Evaluator* then reports to *Dataset Characterizer* in order to keep the workload metrics up-to-date, and accesses the *Dictionary* to get the partitions storing the triples for these queries.

The *star-shaped* queries (object-object or subject-subject joins) are converted to queries over NoSQL document databases. For instance, the object-object star-shaped query Q1 in the following is converted to the access method D1 (MongoDB NoSQL database syntax), and the subject-subject star-shaped Q2 is converted to the access method D2. The `$exists` function of MongoDB filters the JSON documents that have all the predicates of each query, moreover in D2 we also filter by the subject M.

Q1: `SELECT ?x WHERE {x? p5 y? . y? p2 z? .}`
Q2: `SELECT ?x WHERE {x? p9 y? . M p10 y? .}`
D1: `db.partition1.find({p5:{$exists:true}, p2:{$exists:true}})`
D2: `db.partition1.find({p9:{$exists:true}, subject:M})`

The *chain-shaped* queries are converted to queries over NoSQL columnar databases. For example, given the query Q3 in the following, with object-subject and subject-object joins, *Rendezvous* translates it to the set of queries C1 according to the Cassandra NoSQL columnar database query language syntax.

Q3: `SELECT ?x WHERE {x? p1 y?. y? p2 z?. z? p3 w?.}`
C1:
```
SELECT S1,O1 FROM p1
  SELECT S2,O2 FROM p2 WHERE O=S1
  SELECT S3,O3 FROM p3 WHERE O=S2 AND S=D (C1)
```

The processing of joins occurs when a query as a whole cannot be executed on a single partition, and it needs to be decomposed into a set of subqueries, being each subquery evaluated separately and joined at the *Rendezvous* node. For instance, if we consider the graph of Figure 3, the query Q4 in the following is not able to be completed only querying the partitions P1 or P2 alone. In this case, the *Query Decomposer* divides it into subqueries SQ5 and SQ6, issues it to the partitions P1 and P2, respectively, and joins the result sets by matching the predicate *p5* (the connection between P1 and P2).

Q4: `SELECT ?x WHERE {x? p2 y?.y? p3 z?.x? p5 w?.w? p9 k?.L p11 k?.}`
SQ5: `SELECT ?x WHERE {x? p2 y?. y? p3 z?. x? p5 w?.}`
SQ6: `SELECT ?x WHERE {x? p5 w?. w? p9 k?. L p11 k?.}`

As explained before, a complex query is a combination of the star-shaped and chain-shaped patterns, potentially connected by simple queries. The query Q5 in the fol-

lowing presents an example, where *x? p1 y? . y? p2 z? . z? p3 w?* is a chain-shaped pattern, *z? p5 ?k* is a simple query, and *k? p6 G . k? p7 I . k? p8 H* is a star-shaped. In this case, our decomposition process works as follows: *(i)* it first normalizes the query by sorting the patterns by subject and object; *(i)* if we can identify a subset with two or more patterns with the same subject or object, we consider this a star-shaped subquery, like the P1 query below. Then we identify chains in the remaining of the query patterns, i.e.; *(iii)* for each pattern, we navigate from object to subject creating chains. Then we pick the longest chain and consider this a chain-shaped sub-query, like query P2 below. We repeat the step *(iii)* until there are no more chains, or there are only simple patterns, like the query P3 below. Each star-shaped subquery and chain-shaped subquery is processed separately, and the join of the results - along with the simple patterns - is performed at the *Rendezvous* node. In case of ambiguity, *i.e.*, a pattern that is present in more than one query type, we consider the following priority: *(1)* subject-based star-shapes; *(2)* object-based star-shapes; *(3)* the longest chain shapes; and *(4)* simple patterns.

Q5: `SELECT ?x WHERE { x? p1 y? . y? p2 z? . z? p3 w? . z? p5 ?k . k? p6 G . k? p7 I . k? p8 H }`
P1: {`k? p6 G . k? p7 I . k? p8 H }`
P2: {`x? p1 y? . y? p2 z? . z? p3 w?`}
P3: { `z? p5 ?k`}

Rendezvous also provides caching management during the query decomposition. Basically, the *Cache* component verifies, during a query processing, if a fragment (or even a query result) is maintained in the *Rendezvous* cache. The cache is organized as key/value fragments, as defined in the following. **Key/Value Fragment** is a tuple $f_{kv} = (k_kv, V)$ where $f_{kv}.k_kv$ is the name of the key with the form $t_{core}.s : t_{core}.p : t_{core}.o$ (the concatenation of the t_{core} components), and the value $f_{kv}.V$ is a set $f_{kv}.V = \{(t_{1-hop})\}$, being each $t_{1-hop} \in f_{kv}.V$ a triple with a 1-hop distance of t_{core}. *Rendezvous* holds two types of cache: the *Near Cache*, designed as an in-memory *TreeMap*[3] located on each *Rendezvous* server, and a *Remote Cache* maintained as a remote key-value NoSQL database (see Figure 1). When a query is issued against a *Rendezvous* server, it firstly checks its *near cache* to get all the fragments that are already available in the server. Then, the server accesses the *remote cache* to get all the missing fragments, and finally queries the document and columnar databases. The *TreeMap* used in the *near cache* provides an efficient way of storing key/value pairs in sorted order, which is specially interesting to the chain-shaped queries. On using a key-value NoSQL database in the *remote cache*, it is possible to search for keys with wild-cards and patterns (*e.g.*, *Redis*). If the database does not provide this feature, we create all the permutations of indexes for an RDF triple.

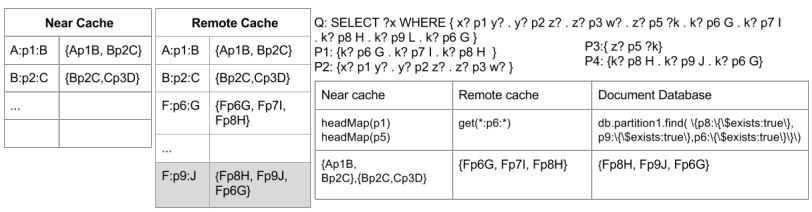


Figure 4. Types of *Rendezvous* Caching

[3]https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html

Figure 4 represents the state of the caches when the query *Q* is issued to a *Rendezvous* server containing the graph of Figure 2 *(i)*. First of all, the server decomposes *Q* into *P1*, *P2*, *P3*, and *P4*. Then, it checks, for the simple and chain-shaped subqueries in the near cache (in this case, *P3* and *P2*, respectively), if any of the triple predicates is the root of a tree containing this chain (in Figure 4, it is represented by the Java commands *headMap(p1)* and *headMap(p5)*). Concurrently, it gets the star-shaped fragments of subquery *P1* from the *remote cache* (*e.g.*, *get(*:p6:*)*). In Figure 4, *P1* returns the necessary triples, but no triples are found in the caches for *P4*. Thus, the server queries the NoSQL databases, as explained in Section 3.2. The triples returned from the queries are stored in both the *near* and *remote caches*. In Figure 3, it is represented by the triples *Fp8H, Fp9J, Fp6G* retrieved from subquery *P4*.

When the cache is almost full, *Rendezvous* automatically evicts some keys. *Rendezvous* currently implements a cache eviction policy for *Most Recently Used (MRU)* as well as *Most Frequently Used (MFU)*. Moreover, the *near cache* is designed to not have false positives, *i.e.*, we prefer to double check with the *remote cache* instead of returning inconsistencies to the client. Due to it, we had developed a communication channel between the *Rendezvous* servers that is able to communicate each other about any update in an existing triple. In Figure 4, if the triple *t=(A, p1, B)* is updated to *t'=(A, p1, B')* in one of the servers S_i, *Rendezvous* removes triple *t* from S_i *remote cache* and send a signal to all the other servers to remove all the fragments that contains *t* from their *near caches*.

4. Experimental Evaluation

This section presents an evaluation of the proposed approach through an experimental evaluation. The considered dataset comes from the *Lehigh University Benchmark (LUBM)* [Guo et al. 2005]. LUBM features an ontology for the University domain, synthetic RDF data scalable to any size, and 14 extensional queries representing a variety of properties. In our experiments, we generate a dataset with 4000 universities. The dataset size is around 100 GB and contains around 500 million triples. Regarding query complexity, we have twelve queries with joins, all of them have at least one subject-subject join, and six of them also have at least one subject-object join. We ran experiments for data loading and querying to test the performance and scalability of *Rendezvous*. *Rendezvous* was developed using Apache Jena version 3.2.0 with Java 1.8, and we use Redis 3.2, MongoDB 3.4.3, and Apache Cassandra 3.10 as the key/value, document and columnar NoSQL databases, respectively, on considering their maturity as representatives of these three families of NoSQL storages. All the nodes are Amazon m3.xlarge spot instances[4] with 7.5 GB of memory and 1 x 32 SSD capacity. For all the experiments, the nodes represent the number of MongoDB + Cassandra servers, always with half of each database. Moreover, we created a cluster of Redis in 3 nodes with the same configuration. We also created one partition for each server, and the *Rendezvous* servers were installed alone in each node. All the queries were issued from a server in the same network, so the latency between the client and *Rendezvous* was inexpressive.

The total dataset size, the loading time, and the average querying time are shown in Figure 5 (a) to (c), respectively. In Figure 5 (a) and Figure 5 (b), we notice that the dataset and the loading time grow exponentially with the number of nodes and the n-

[4]https://aws.amazon.com/ec2/instance-types/

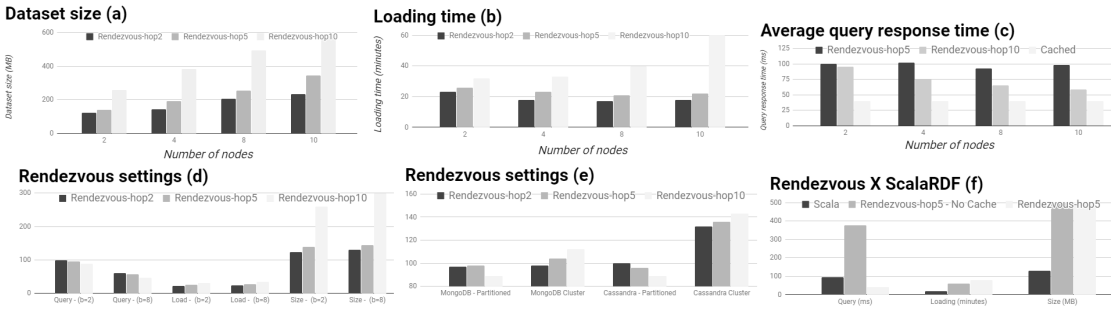


Figure 5. Summary of the Experimental Evaluation

hop, ramping up from around 102 GB, loaded around 20 minutes in the 2 nodes with 2-hop configuration, to more than 500 GB loaded in more than 60 minutes in the 10 nodes with 10-hop configuration. These results can make *Rendezvous* very costly in cloud environments that charge per storage usage. We are investigating compression techniques to mitigate this problem. In Figure 5 (c), we studied both the not cached configuration and cached configuration. In the not cached configuration, the best response time was achieved with 10 nodes and a 10-hop. In such a configuration, we did not register any join outside of the fragment - with this significant fragment size all the queries could be solved within a unique NoSQL access - and each server CPU and Memory load were very small. The cached configuration presents the same response time (around 40 ms in average) regardless the hop size. The results show that the fragmentation and partition solution of *Rendezvous* is scalable and if the tradeoff for the dataset size is acceptable, the average response time can be subtle, and that cache is a good solution for scalability. In Figure 5 (d) and (e) we compared different settings of *Rendezvous*. In Figure 5 (e), we show that the systematic replication of fragments in the boundary of each partition ("b" parameter) increases the speed on the query response, without a big impact in the total size of the dataset and the data load time. This is because the size of the boundaries' triples is not very significant in such big dataset. This result is motivating new studies on the optimal boundary replication size to accelerate the query response. In Figure 5 (d) we compared our partitioning solution to the NoSQL database partition solutions. We analyzed here *Rendezvous* accessing each NoSQL database server separately (as a partition), as well as accessing the servers as a cluster (delegating the data partition to the NoSQL database). The results show that, especially for Cassandra, the graph awareness of the proposed schema plus the replication boundary lead to better performance. For MongoDB, we can conclude that the most important factor is the size of the fragment (n-hop) since a bigger fragment will typically lead to a smaller number of database accesses. Finally,in Figure 5 (f) we compared the performance of *Rendezvous* 5-hop - cached and not cached - with the recent related work *ScalaRDF*[5]. Our cached solution is 30 percent faster on average. This results is mainly due our Near Cache component - which is not present in ScalaRDF - and avoid network latency between the Rendezvous server and Redis. The downsides of this comparison are the loading time - is almost twice slower - and the dataset size - almost five times bigger.

[5]The code for ScalaRDF was found in https://github.com/xinghuayu007/ScalaRDF/

5. Conclusion

This paper presents *Rendezvous*, a novel workload-aware RDF partitioning approach for the persistence of RDF data into NoSQL Databases. We based it on a middleware that can, according to the typical shape of the main SPARQL queries, define RDF fragments and store them into the document, columnar and key/value NoSQL databases. The partition is used when the dataset is bigger than each server capabilities. In this case, we considered a replication boundary to avoid cross-server joins and speed up the query response time. Our experiments reveal that a bigger replication boundary can accelerate the queries without a negative impact regarding storage space and load time. Besides, *Rendezvous* outperformed a recent disk-based baseline, denoting that our proposal is promising. In general, *Rendezvous* is a contribution to the problem of efficient mapping of the RDF data model to NoSQL data models. Even so, we have some future works in mind. First of all, we are considering implementing algorithm for triples compression. The lack of this feature makes *Rendezvous* uses exponentially more storage space as the n-hop horizon grows. We also intend to consider update and deletion operations, other NoSQL types in the *Rendezvous* architecture as well as cluster capabilities in the *Rendezvous* server. With these improvements, we aim at comparing it again with the related work.

References

Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K. (2009). Sw-store: a vertically partitioned dbms for semantic web data management. *The VLDB JournalThe International Journal on Very Large Data Bases*, 18(2):385–406.

Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5):28–37.

Bugiotti, F., Bursztyn, D., Diego, U. C. S., and Ileana, I. (2015). Invisible Glue : Scalable Self-Tuning Multi-Stores. *Cidr 2015*.

Gallego, M. A., Fernández, J. D., Martínez-Prieto, M. A., and de la Fuente, P. (2011). An empirical study of real-world sparql queries. In *USEWOD workshop*.

Guo, Y., Pan, Z., and Heflin, J. (2005). Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, S. and Agents on the WWW*, 3(2):158–182.

Hu, C., Wang, X., Yang, R., and Wo, T. (2016). Scalardf: a distributed, elastic and scalable in-memory rdf triple store.

Ma, Z., Capretz, M. A., and Yan, L. (2016). Storing massive resource description framework (rdf) data: a survey. *The Knowledge Engineering Review*, 31(4):391–413.

Mulay, K. and Kumar, P. S. (2012). Spovc: a scalable rdf store using horizontal partitioning and column oriented dbms. In *Proceedings of the 4th International Workshop on Semantic Web Information Management*, page 8. ACM.

Sadalage, P. J. and Fowler, M. (2012). *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education.

Weiss, C., Karras, P., and Bernstein, A. (2008). Hexastore: sextuple indexing for semantic web data management. *Proceedings of the VLDB Endowment*, 1(1):1008–1019.

32th Brazilian Symposium on Databases

short

October 2nd to 5th, 2017
Uberlândia – MG – Brazil

SBBD PROCEEDINGS

SHORT PAPERS

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Uberlândia – UFU

Program Chairs

Bernadette Farias Lóscio, UFPE
Damires Yluska Souza Fernandes, IFPB

32nd Brazilian Symposium on Databases

October 2nd to 5th, 2017
Uberlândia – MG – Brazil

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Uberlândia – UFU

SBBD Steering Commitee

Agma Juci Machado Traina, USP
Bernadette Lóscio, UFPE
Caetano Traina Jr., USP
Carmem Hara, UFPR
Javam Machado, UFC
Mirella M. Moro, UFMG
Vanessa Braganholo, UFF

SBBD 2017 Commitee

Steering Committee Chair
Javam Machado, UFC

Local Organization Chairs
Maria Camila N. Barioni, UFU and Humberto L. Razente, UFU

Program Committee Chair
Carmem S. Hara, UFPR

Short papers Chairs
Bernadette Lóscio, UFPE and Damires Souza, IFPB

Demos and Applications Session Chair
Daniel de Oliveira, UFF

Short Courses Chair
Vaninha Vieira, UFBA

Workshop on Thesis and Dissertations in Databases Chair
Carina Dorneles, UFSC

Tutorials Chair
Ana Carolina Salgado, UFPE

Thesis and Dissertation Contest Chair
Vânia Vidal, UFC

Workshops Chair
Fernanda Baião (UNIRIO)

Local Organization Committee

Maria Camila N. Barioni, UFU
Humberto L. Razente, UFU
José Gustavo de Souza Paiva, UFU
Marcelo Zanchetta do Nascimento, UFU
Elaine Ribeiro de Faria Paiva, UFU
João Henrique de Souza Pereira, UFU

Short Papers Program Committee

Alessandreia Oliveira, UFJF
Altigran Soares da Silva, UFAM
Ana Carolina Almeida, UERJ
Anderson Ferreira, UFOP
Angelo Brayner, UFC
Carina F. Dorneles, UFSC
Carlos Eduardo Pires, UFCG
Carmem Hara, UFPR
Celso Hirata, ITA
Clodoveu Davis, UFMG
Cristiano Cervi, UPF
Damires Souza, IFPB
Daniel de Oliveira, UFFS
Daniel Kaster, UEL
Daniel Notari, UCS
Daniela Barreiro Claro, UFBA
Deise Saccol, UFSM
Denio Duarte, UFFS
Duncan Ruiz, PUCRS
Eduardo de Almeida, UFPR

Eduardo Ogasawara, CEFET/RJ
Elaine Sousa, USP
Eveline Sacramento, FUNCEME
Fabio Porto, LNCC
Fernanda Baião, UNIRIO
Flávio R. C. Sousa, UFC
Helena Ribeiro, UCS
Humberto Razente, UFU
João B. Rocha-Junior, UEFS
Jonice Oliveira, DCC/IM/UFRJ
José Palazzo Moreira de Oliveira, UFRGS
José Antonio Macêdo, UFC
José de Aguiar Moraes Filho, UNIFOR
José Maria Monteiro, UFC
Karin Becker, UFRGS
Kelly Braghetto, IME/USP
Luiz Celso Gomes Jr, UTFPR
Marco Antonio Casanova, PUC-Rio
Maria Camila Nardini Barioni, UFU
Maria Claudia Cavalcanti, IME
Marilde Santos, UFSCar
Maristela Holanda, UnB
Mirella M. Moro, UFMG
Moisés Carvalho, UFAM
Pedro Eugenio Rocha Pedreira, Facebook Inc.
Raquel Stasiu, PUCPR/Universidade Tecnológica Federal do Paraná
Raqueline Penteado, UEM
Rebeca Schroeder, UDESC
Renato Fileto, UFSC
Robson Cordeiro, ICMC - USP
Robson Fidalgo, UFPE
Ronaldo Mello, UFSC
Sergio Lifschitz, PUC-Rio
Sergio Mergen, UFSM
Thiago Silva, UTFPR
Ticiana Coelho da Silva, UFC
Valéria C. Times, UFPE
Vania Bogorny, UFSC
Vaninha Vieira, UFBA
Wagner Meira Jr., UFMG
Wellington Martins, UFG

External Reviewers

Carlos Teles, CEFET/RJ
Christian Bones, USP
Demetrio Mestre, UFCG
Flavio Carvalho, CEFET-RJ

Leonardo Moreira, UFC
Marcelo Iury S . Oliveira, UFRPE

Table of Contents (Short Papers)

A Força dos Relacionamentos Pode Medir a Qualidade de Comunidades? 204
Mariana O. Silva (UFMG), Michele A. Brandão (UFMG) e Mirella M. Moro (UFMG)

A Predictive Load Balancing Service for Cloud-Replicated Databases 210
Carlos Marinho (UFC), Emanuel Coutinho (UFC), José Serafim da Costa Filho (UFC), Leonardo Moreira (UFC), Flávio R. C. Sousa (UFC) e Javam Machado (UFC)

Análise da Formação e Evolução de Times de Desenvolvimento no Hibernate-ORM 216
Talita S. Orfanó (UFMG), Michele A. Brandão (UFMG), Larissa E. Maia (UFMG) e Mirella M. Moro (UFMG)

Análise de Dados Científicos: uma Análise Comparativa de Dados de Simulações Computacionais 222
Thaylon Guedes Santos (UFF), Vítor Silva (UFRJ), Jose Camata (UFRJ), Marta Mattoso (UFRJ) e Daniel de Oliveira (UFF)

Análise do Impacto do Gerador de Conjuntos de Dados em Experimentos de Deduplicação de Dados 228
Levy de Souza Silva (UFMG) e Mirella M. Moro (UFMG)

Aspectos Temporais para Medir a Força da Colaboração no GitHub 234
Natércia A. Batista (UFMG), Michele A. Brandão (UFMG), Ana Paula C. da Silva (UFMG) e Mirella M. Moro (UFMG)

Detecção de Anomalias no Transporte Rodoviário Urbano 240
Ana Beatriz Cruz (CEFET/RJ), João Ferreira (CEFET/RJ), Bernardo Monteiro (CEFET/RJ), Rafaelli Coutinho (CEFET/RJ), Fabio Andre Porto (LNCC) e Eduardo Ogasawara (CEFET/RJ)

Don't Keep My UDFs Hostage - Exporting UDFs For Debugging Purposes ... 246
Pedro Holanda (CWI- The Netherlands), Mark Raasveldt (CWI – The Netherlands) e Martin Kersten (CWI – The Netherlands)

Empirical Evaluation of Strategies to Process Range Queries of Numeric Sequences in Batch-mode 252
Luiz F. A. Brito (UFU) e Marcelo K. Albertini (UFU)

Estratégia Distribuída para Análise de Assuntos Abordados no Twitter Via Evolução de Clusters 258
Priscila Rocha Ferreira Rodrigues (UFC), Ticiana L. Coelho da Silva (UFC), José Maria da Silva Monteiro Filho (UFC) e José Antonio F. de Macêdo (UFC)

Mineração de Perfis Sociais em Redes Temporais 264
Jeancarlo C. Leão (UFMG), Michele A. Brandão (UFMG), Pedro O. S. Vaz de Melo (UFMG) e Alberto H. F. Laender (UFMG)

Mineração de Regras de Associação Multirrelação em Grafos: Direcionando o Processo de Busca 270
Felipe Alves Oliveira (IME), Raquel L. Costa (IME), Ronaldo Goldschmidt (IME) e Maria Claudia Cavalcanti (IME)

Online Performance Modeling for NoSQL Databases using Extreme Learning Machines 276
Victor A. E. Farias (UFC), Pedro R. A. Pinheiro (UFC), Flávio R. C. Sousa (UFC), João P. P. Gomes (UFC) e Javam C. Machado(UFC)

Redes Sociais Científicas: análise topológica da influência dos pesquisadores ... 282
Vitor Horta (UFJF), Victor Stroele (UFJF), Fernanda Campos (UFJF), José Maria David (UFJF) e Regina Braga (UFJF)

Spark Scalability Analysis in a Scientific Workflow 288
Renan Santos Souza (UFRJ e IBM Brazil), Vítor Silva (UFRJ), Pedro Miranda (UFRJ), Alexandre Lima (UFRJ), Patrick Valduriez (INRIA-France) e Marta Mattoso (UFRJ)

Spatial Join on Positional Uncertain Data 294
Welder Oliveira (UFG), Sávio de Oliveira (UFG), Vagner Sacramento (UFG), Helton Santos (UFG) e Kleber Cardoso (UFG)

Uma Abordagem para Processamento Distribuído de Junção por Similaridade sobre Múltiplos Atributos 300
Diego Junior do Carmo Oliveira (UFG), Felipe Ferreira Borges (UFG) e Leonardo Andrade Ribeiro (UFG)

Uma Análise Comparativa entre Sistemas Gerenciadores de Bancos de Dados NoSQL no contexto de Internet das Coisas 306
Allexandre Sampaio Santos Soares (IFBA) e Pablo Freire Matos (IFBA)

Uso de instâncias de dados e carga de trabalho para mineração de restrições de integridade 312
Eduardo Henrique Monteiro Pena (UTFPR) e Eduardo Cunha de Almeida (UTFPR)

paper:174600

A Força dos Relacionamentos pode Medir a Qualidade de Comunidades?

Mariana O. Silva, Michele A. Brandão, Mirella M. Moro

Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

{mariana.santos,micheleabrandao,mirella}@dcc.ufmg.br

Abstract. *In social networks, community detection provides valuable data about relationships between individuals. There are various metrics to validate the quality of communities, but there is no consensus on the performance of these metrics. In this paper, we evaluate whether strength metrics can also be used to measure the quality of algorithms that detect communities. The results are positive to confirm such hypothesis.*

Resumo. *Em redes sociais, a detecção de comunidades fornece valiosos dados sobre as relações entre indivíduos. Há diversas métricas para validar a qualidade de comunidades, mas não há um consenso sobre o desempenho dessas métricas. Neste artigo, avaliamos se métricas para força dos relacionamentos podem também ser usadas para medir a qualidade de algoritmos que detectam comunidades. Os resultados são positivos para confirmar tal hipótese.*

1. Introdução

Estudos para a análise de interações entre pessoas ou organizações, bem como a detecção de padrões nessas interações, permitem prever o comportamento de uma rede e analisar diferentes aspectos da mesma. No contexto acadêmico, uma rede social de co-autoria pode ser representada por colaborações científicas que possuem padrões e características relacionados às interações dos indivíduos envolvidos. Tais indivíduos são representados por pesquisadores e os vínculos relacionais, as colaborações entre eles.

Em tais redes, existem grupos de pessoas que possuem relacionamentos mais fortes e compartilham interesses semelhantes [Brandão and Moro 2017] e são chamados de comunidades (agrupamentos ou *clusters*). A detecção de tais comunidades tornou-se um problema fundamental [Yang et al. 2016], podendo ajudar a obter valiosos dados sobre a existência de grupos que colaboram mais densamente, a identificação de relacionamentos mais intensos entre determinados autores ou ainda autores com maior grau de colaboração [Procópio et al. 2011]. Uma vez que este problema não possui uma solução exata, muitas heurísticas foram propostas para encontrar *clusters*. No entanto, não há garantia formal de que os resultados obtidos através deles sejam os melhores possíveis [Almeida et al. 2012].

Um dos aspectos mais importantes do processo de identificação de comunidades é a avaliação da qualidade dos algoritmos que as detectam. É fundamental não só para medir a eficiência dos algoritmos de agrupamento, mas também para prover uma visão sobre a dinâmica de relacionamentos em uma determinada rede. Muitas métricas para validar a qualidade de comunidades já foram propostas na literatura, mas não há um consenso sobre como elas se comparam e quão bons são seus desempenhos [Almeida et al. 2012].

Nesse contexto, este trabalho realiza uma análise comparativa entre métricas para a força dos relacionamentos em redes de co-autoria e métricas usadas para validar comunidades. Especificamente, utilizamos tais métricas para avaliar a qualidade de comunidades detectadas por três algoritmos comumente aplicados em grafos não direcionados [Brandão and Moro 2017]: *Louvain Method* (LM) [Blondel et al. 2008], *Clique Percolation Method* (CPM) [Palla et al. 2005] e *Markov Cluster Algorithm* (MCL) [Van Dongen 2000].

Após apresentar os trabalhos relacionados (Seção 2), descrevemos a metodologia para realização deste trabalho (Seção 3). Em seguida, apresentamos os resultados obtidos (Seção 4). Finalmente, discutimos as principais conclusões (Seção 5).

2. Trabalhos Relacionados

O processo de agrupamento (*clustering*) tem sido aplicado em diversos campos, incluindo Engenharias, Ciência da Computação, Ciências Médicas e Economia [Xu and Wunsch 2005]. Na área de ciência da computação, em especial, essas técnicas de agrupamento são utilizadas para detectar comunidades em redes sociais. Este processo é um problema fundamental, uma vez que permite a análise das interações e relacionamentos entre os participantes das redes. Especialmente em redes sociais acadêmicas, a detecção de comunidades auxilia na descoberta de padrões que podem aumentar a produtividade dos pesquisadores bem como entender a formação de grupos.

Identificar comunidades em redes sociais de co-autoria é geralmente uma tarefa difícil, sendo necessário utilizar ferramentas para detectar e entender o comportamento das mesmas. De acordo com Mishra et al. [2007], modularidade, mutualidade, acessibilidade e cadeias de Markov, são exemplos de estratégias para detectar comunidades em redes sociais. Diante da variedade de técnicas, aplicamos três comumente utilizadas em grafos não direcionados, o *Louvain Method* que é baseado em modularidade [Blondel et al. 2008], *Clique Percolation Method* que considera o conceito de mutualidade [Palla et al. 2005] e *Markov Cluster Algorithm* que utiliza cadeias de Markov [Van Dongen 2000].

Para validar a qualidade das comunidades detectadas pelos algoritmos de agrupamento escolhidos, utilizamos as métricas *neighborhood_overlap* e *co-authorship frequency*. No entanto, de acordo com Brandão e Moro [2017], considerar apenas métricas para a força dos relacionamentos de uma rede não é suficiente para definir a qualidade de algoritmos de clusterização. Portanto, aplicamos também métricas que são comumente utilizadas para avaliar a qualidade de comunidades. Nosso estudo investiga se métricas para a força dos relacionamentos podem medir a qualidade das comunidades de forma adequada.

3. Metodologia

A metodologia deste trabalho possui cinco passos: inicialmente, criamos as redes sociais de co-autoria para analisar a detecção de comunidades; em seguida, aplicamos duas métricas para a força dos relacionamentos dos pesquisadores; então, utilizamos três algoritmos de agrupamento para detectar comunidades; finalmente, validamos a qualidade dos grupos através de três índices e analisamos os resultados das diferentes métricas.

Redes Sociais de Co-autoria. Utilizamos um conjunto de dados coletado da DBLP[1] com aproximadamente 15 milhões de registros para construir as redes sociais. Estes

[1]DBLP *Digital Bibliography & Library Project*): `http://dblp.uni-trier.de/`

Tabela 1: Descrição das redes sociais criadas.

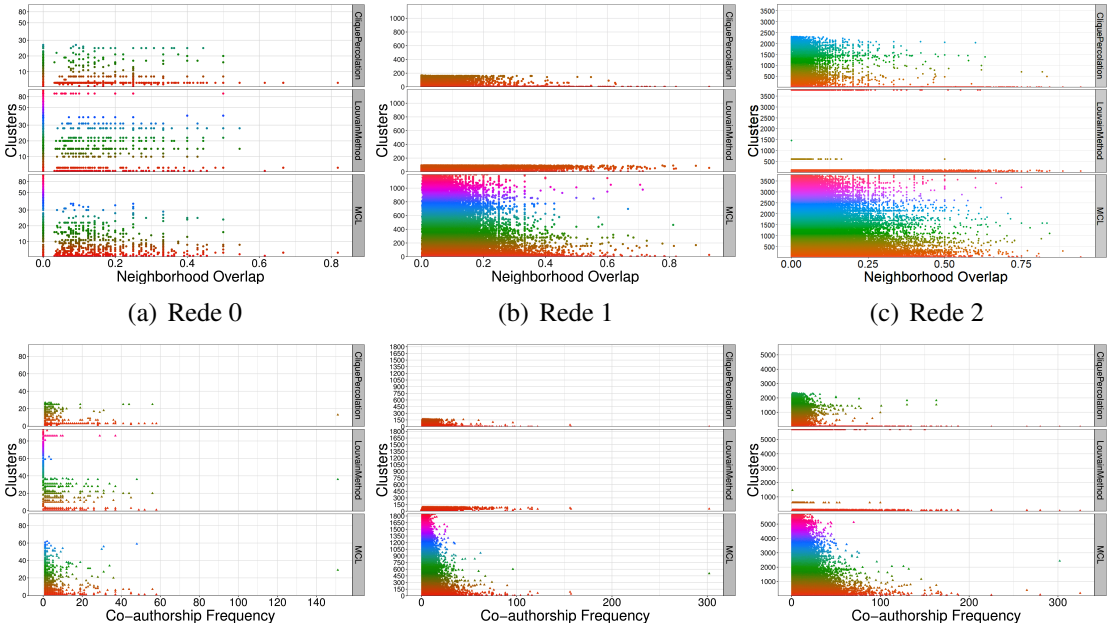
Rede	# autores	# pares (# dist)	MedPubA	Modularidade	Coeficiente de clusterização médio
0	394	3898 (738)	9,89	0,686	0,377
1	68.397	249.352 (110.357)	3,65	0,86	0,672
2	314.444	1.297.929 (540.571)	4,13	0,691	0,524

dados produziram uma rede social extremamente grande, tornando a análise de rede muito trabalhosa. Diante deste problema, utilizamos a técnica de amostragem não probabilística conhecida como *snowball sampling* [Goodman 1961]. A partir de uma rede origem formada pelos bolsistas vigentes (Abril de 2017) de produtividade do CNPq[2] da área de Ciência da Computação, foram feitas mais duas coletas para aumentar a amostra e criar três redes reais de tamanhos diferentes. Ao final, são três redes criadas a partir da DBLP: (0) formada apenas pelos bolsistas vigentes do CNPq (que fazem parte da DBLP); (1) formada pela rede 0 e seus vizinhos; e (2) formada pela rede 1 e seus vizinhos. Dessa forma, foi possível analisar como o tamanho das redes influencia na formação e validação de comunidades. A Tabela 1 apresenta os números de pesquisadores e de publicações, número médio de publicações por autor, número de pares de co-autores (e distintos), modularidade da rede e o coeficiente de clusterização médio de cada rede.

Configuração da Análise. Para a análise comparativa, as redes sociais de co-autoria foram modeladas como um grafo ponderado não dirigido $G^w = (V, E^w)$, onde V é o conjunto de nós e E^w o conjunto de arestas ponderadas. Os nós representam todos os pesquisadores na rede, enquanto as arestas representam a co-autoria em publicações. Além disso, o peso da aresta representa o número absoluto de publicações entre eles, chamado de *co-authorship frequency* (frequência de co-autoria). Para analisar as comunidades, aplicamos os três algoritmos de agrupamento: *Louvain Method* (LM), *Clique Percolation Method* (CPM) e *Markov Cluster Algorithm* (MCL). O método de Louvain é um dos algoritmos de *clustering* mais utilizados e baseia-se na otimização da modularidade de uma partição de rede. O segundo método é o Clique Percolation que é capaz de detectar comunidades com sobreposição, ou seja, os nós podem pertencer a mais de uma comunidade. Finalmente, o Markov Cluster detecta *clusters* alternando dois processos de Markov: expansão e inflação.

Para medir a força dos relacionamentos, utilizamos as métricas *co-authorship frequency* e *neighborhood_overlap* (sobreposição de vizinhança). Dados dois pesquisadores v_i e v_j, *neighborhood_overlap* é definida pela equação: $\frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j) - \{v_i, v_j\}|}$, onde $N(v_i)$ representa os co-autores do pesquisador v_i, e $N(v_j)$ os co-autores de v_j. Essa propriedade topológica calcula a colaboração entre dois nós em relação ao seus vizinhos. Seguindo [Brandão and Moro 2015], consideramos que um relacionamento é fraco quando o valor da métrica *neighborhood_overlap* está no intervalo $[0; 0.2]$ e forte, caso contrário. Da mesma forma, um laço é fraco quando a frequência de co-autoria está no intervalo $[1; 5]$ e forte, caso contrário. Para verificar se métricas para a força dos relacionamentos podem ser usadas para avaliar a qualidade do agrupamento, comparamos seus resultados com três métricas de validação interna: *C-index* (C), *Dunn Index* (Dunn) e *Davies Bouldin Index* (DB). Tais métricas utilizam noções de similaridade intra-comunidade, em contraste com as noções de separação inter-comunidades [Zaki and Meira Jr 2014]. Pela definição de

[2]Bolsistas de produtividade do CNPq: `http://cnpq.br/bolsistas-vigentes/`



(a) Rede 0 (b) Rede 1 (c) Rede 2

(d) Rede 0 (e) Rede 1 (f) Rede 2

Figura 1: Resultados dos algoritmos de agrupamento (cada cor representa um *cluster*.

clusters [Blondel et al. 2008; Palla et al. 2005; Van Dongen 2000], os vínculos intra-comunidade devem ser fortes e os vínculos inter-comunidades devem ser fracos. Logo, uma comunidade deve ter a maioria dos pares de pesquisadores classificados como fortes e a maioria dos laços que conectam diferentes grupos devem ser fracos.

4. Resultados

Esta seção compara os resultados das métricas para a força dos relacionamentos com as métricas de validação interna considerando as comunidades detectadas pelos LM, CPM e MCL. A Figura 1 apresenta a comparação das comunidades detectadas por cada algoritmo em relação às duas métricas de força dos relacionamentos, *neighborhood_overlap* e *co-authorship frequency*. Nota-se que o CPM encontra um número menor de comunidades em relação aos demais métodos, na maioria dos casos. Por outro lado, o MCL detecta o maior número de grupos. Ademais, é importante enfatizar que algumas relações fortes foram excluídas no método CPM, pelo fato dessas arestas pertencerem a um 2-clique. Isso ocorre porque na aplicação do método, foi escolhido k=3. Essa limitação pode ter restringido o número de comunidades identificados pelo algoritmo, fazendo com que ele tenha sido o método com menor número de comunidades formadas.

Analisando as principais vantagens de cada algoritmo, observamos que o LM não foi capaz de detectar pequenas comunidades em grandes redes. Por outro lado, o CPM permite a sobreposição de comunidades, sendo uma situação bem rotineira em redes sociais reais (pesquisadores publicam com pesquisadores de outras comunidades). No entanto, pela necessidade da escolha do parâmetro *k*, o método CPM não teve desempenho melhor que o MCL, visto que o MCL não detectou relacionamentos entre pesquisadores de diferentes comunidades nas três redes. Ou seja, uma vez que o objetivo de um algoritmo de agrupamento é maximizar o número de arestas intra-comunidade e minimizar as inter-comunidade, este foi o algoritmo que forneceu o melhor resultado.

Tabela 2: Resultados das métricas para a força dos relacionamentos.

	Neighborhood Overlap			Co-authorship Frequency		
Rede	**1º**	**2º**	**3º**	**1º**	**2º**	**3º**
0	MCL	LM	CPM	MCL	LM	CPM
1	MCL	CPM	LM	MCL	CPM	LM
2	MCL	CPM	LM	MCL	CPM	LM

Tabela 3: Resultados dos índices de validação interna.

		Neighborhood Overlap			Co-authorship Frequency		
Índice	**Rede**	**1º**	**2º**	**3º**	**1º**	**2º**	**3º**
C	**0**	MCL	LM	CPM	MCL	LM	CPM
C	**1**	-	-	-	-	-	-
C	**2**	-	-	-	-	-	-
Dunn	**0**	LM	CPM	MCL	LM	CPM	MCL
Dunn	**1**	MCL/LM	CPM	-	MCL/LM	CPM	-
Dunn	**2**	-	-	-	-	-	-
DB	**0**	LM/MCL	CPM	-	LM/MCL	CPM	-
DB	**1**	CPM	MCL	LM	CPM	MCL	LM
DB	**2**	-	-	-	-	-	-

Tabela 4: Resultados dos diferentes tipos de métricas.

Rede	**NO**	**C-index**	**Dunn Index**	**DB Index**	**CF**	**C-index**	**Dunn Index**	**DB Index**
0	MCL	MCL	MCL	LM/MCL	MCL	MCL	LM	LM/MCL
1	MCL	-	MCL/LM	CPM	MCL	-	MCL/LM	CPM
2	MCL	-	-	-	MCL	-	-	-

Em relação ao tamanho das redes, podemos notar que ao aumentar o número de nós conectados, a frequência de co-autoria aumentou menos do que o esperado. Da *Rede 1* para a *Rede 2* (Figuras 1(e) e 1(f), respectivamente), não houve um crescimento significativo no nível de colaboração. Por outro lado, foi nítido o crescimento no valor da sobreposição de vizinhança, à medida que o tamanho das redes aumenta. Além disso, nas Figuras 1(a), 1(b) e 1(c), nota-se que há uma alta concentração de arestas somente até o valor 0.4 em todas as redes. Considerando a frequência de co-autoria, as Figuras 1(d), 1(e) e 1(f) mostram uma alta concentração de arestas com frequência de co-autoria menor do que 40 na *Rede 0* e menor do que 100 nas outras duas redes.

As Tabelas 2, 3 e 4 apresentam os resultados das métricas para a força dos relacionamentos, dos índices de validação interna e a comparação entre as diferentes métricas, respectivamente. Nas duas primeiras tabelas, os resultados foram ordenados de acordo com o desempenho dos métodos de agrupamento (primeiro, segundo e terceiro melhor algoritmo). Na Tabela 2, comparando os resultados das duas métricas, observa-se que os algoritmos apresentaram os mesmos desempenhos independente da métrica analisada. Além disso, o algoritmo que realizou o melhor agrupamento foi o MCL em todos os casos. Em redes maiores, o segundo melhor método foi o CPM, seguido pelo LM. Já na menor rede (*Rede 0*), o LM apresenta o segundo melhor resultado. Isso pode ser explicado pelo fato de que algoritmos de otimização de modularidade, geralmente, apresentam dificuldades em detectar pequenas comunidades em grandes redes.

Na Tabela 3, a ordem de desempenho dos algoritmos qualificados pelas medidas de validação interna é similar aos resultados apresentados na Tabela 2 . Já a Tabela 4 apresenta uma comparação entre as métricas para força dos relacionamentos e as métricas para validação interna. Analisando apenas a *Rede 0*, nota-se que na maioria dos casos as

medidas de validação interna indicam o mesmo método com melhor resultado: o algoritmo MCL. Finalmente, os três algoritmos detectam comunidades com relacionamentos fracos e fortes. No entanto, o MCL mostrou-se capaz de detectar grupos com relacionamentos mais fortes do que fracos. Similarmente, analisando os resultados das métricas de validação interna, o algoritmo MCL apresenta comunidades mais coesas na maioria dos casos.

5. Conclusão

Neste trabalho, aplicamos três algoritmos de agrupamento em três redes sociais de co-autoria criadas a partir de dados coletados da DBLP. Os resultados mostram que o MCL é o melhor algoritmo de agrupamento a ser aplicado em redes sociais de co-autoria quando comparado ao LM e CPM, pois a força dos relacionamentos *inter-cluster* tendem a ser mais fortes nessa técnica do que nas demais. Ademais, os resultados mostram que as métricas para a força de relacionamento, principalmente a *neighborhood_overlap*, podem também auxiliar na avaliação da qualidade de algoritmos que detectam comunidades. Pesquisas futuras incluem realizar a mesma comparação gerando redes sociais de co-autoria sintéticas. Além disso, planejamos aplicar mais métricas para força dos relacionamentos e avaliar mais algoritmos de detecção de comunidades.

Agradecimentos. Trabalho parcialmente financiado por CAPES, CNPq e FAPEMIG.

Referências

Almeida, H., Guedes Neto, D., Meira Jr., W., and Zaki, M. J. (2012). Towards a better quality metric for graph cluster evaluation. *JIDM*, 3(3):378.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *J. Stat. Mech.*, 2008(10):P10008.

Brandão, M. A. and Moro, M. M. (2015). Analyzing the strength of co-authorship ties with neighborhood overlap. In *DEXA*, pages 527–542, Valencia, Espanha.

Brandão, M. A. and Moro, M. M. (2017). A comparative analysis of the strength of co-authorship ties in clusters. In *AMW*, Montevideo, Uruguai.

Goodman, L. A. (1961). Snowball sampling. *Ann. Math. Statist.*, 32(1):148–170.

Mishra et al., N. (2007). Clustering social networks. In *WAW*, pages 56–67, S.Diego, USA.

Palla et al., G. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818.

Procópio, P., Laender, A. H., and Moro, M. M. (2011). Análise da rede de coautoria do simpósio brasileiro de bancos de dados. In *SBBD Short Papers*, Florianópolis, Brasil.

Van Dongen, S. M. (2000). *Graph clustering by flow simulation*. PhD thesis, University of Utrecht.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678.

Yang, Z., Algesheimer, R., and Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6(30750).

Zaki, M. J. and Meira Jr, W. (2014). *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press.

paper:174094

A Predictive Load Balancing Service for Cloud-Replicated Databases

Carlos S. S. Marinho[1,2], Emanuel F. Coutinho[1], José S. Costa Filho[2], Leonardo O. Moreira[1,2], Flávio R. C. Sousa[2], Javam C. Machado[2]

[1]Instituto Universidade Virtual (UFC Virtual)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

[2]Laboratório de Sistemas e Banco de Dados (LSBD)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

sergio.marinho@lsbd.ufc.br, emanuel@virtual.ufc.br,

{serafim.costa,leonardo.moreira,flavio.sousa,javam.machado}@lsbd.ufc.br

Abstract. *Cloud computing emerges as an alternative to promote quality of service for data-driven applications. Database Management Systems must be available to support the deployment of cloud applications resorting to databases. Many solutions use database replication as a strategy to increase availability and decentralize the workload of database transactions between replicas. By the distribution of database transactions between replicas, load balancing techniques improve the computational resources utilization. However, several solutions use the current state of the database service in making decisions for the distribution of transactions. This paper proposes a predictive load balancing service for replicated cloud databases.*

1. Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [Mell and Grance 2011]. The cloud computing infrastructure is typically composed of a large number of physical machines, connected through a network. On each physical machine there is a variable number of virtual machines (VMs), according to the hardware capacity available on the physical machine. In these VMs, services are executed and usually must attend to some Service Level Agreement (SLA) contracted by the users. SLA can be defined as an obligation where the cloud provider gives its users some guarantees of Quality of Service (QoS) levels such as: performance and availability [Moreira et al. 2012].

Many cloud applications are data-driven, and because of this, Database Management Systems (DBMSs) are potential candidates for cloud deployment [Moreira et al. 2014]. Other reasons are: (i) in general, the DBMS installations are complex and involve a large amount of data, causing high hardware and software costs [Moreira et al. 2012]; (ii) most of the time spent processing in data-driven applications is related to processing in the DBMS [Sousa et al. 2012]. Several solutions are being proposed to use cloud databases with aspects of quality of service [Sousa and Machado 2012] [Moreira et al. 2014]. Some solutions that utilize database replication use load balancing

strategies to distribute workloads among the replicas. However, the identified existing solutions do not use predictive aspects to observe the future impact of the workload in relation to the use of existing cloud resources and SLA compliance.

The hypothesis of this research is as follows: *The use of prediction in load balancing solutions for replicated databases can maximize resource utilization in computational clouds while taking into account aspects of SLA*. From this hypothesis, the main objective is to design and implement a predictive load balancing service for replicated cloud databases. In order to achieve the main objective, the following specific objectives have been established: (i) to analyze load balancing techniques in replicated databases that can be used in computational clouds; (ii) to study techniques used to forecast workload in databases; (iii) to design a load balancing service that uses workload forecasting techniques for replicated cloud databases; and (iv) to evaluate whether the forecast techniques can be used to observe performance behavior of replicated cloud databases.

2. Related Work

The following filtering criteria were used to collect some related work: (i) works used in replicated cloud databases; (ii) works that adopted, in some way, load balancing between replicas; and (iii) works that used the relational model as a model of persistence. Table 1 lists the related works and highlights their characteristics. Sousa and Machado (2012) developed RepliC, an approach for total replication of relational database in the cloud. This work considers the aspects of quality of service, elasticity and multi-tenant model of shared DBMS. In RepliC there is the adjustment of the number of replicas according to the workload to comply the SLA. To divide the workload between the replicas, RepliC uses load balancing implemented as a circular queue (round-robin), distributing the transactions evenly across the existing replicas.

Moon et al. (2013) proposed SWAT, a middleware for load balancing on replicated cloud databases. SWAT uses the same multi-tenant model adopted by RepliC that does the DBMS sharing to host the replicas in the VMs. The load balancing strategy implemented by SWAT directs the workload to the replica that has greater computational resources availability at a given time. Although it seeks to better management and efficient use of existing resources in the cloud, SWAT does not solve problems related to SLA violations when resources are scarce, because it does not employ provisioning or elasticity. Pippal et al. (2015) developed a strategy based on partial replication, using the read-one-write-all model. This model proposes that the write requests are forwarded to all replicas and the read requests are only for the server that has the less available computational resources. In this work, CPU usage was considered for the requests distribution decision and experiments were made using the MySQL Server. Farther, it is necessary for the Database Administrator to configure which tables should be replicated together.

Table 1. Main related works and their characteristics

Work	Replication Strategy	Persistence Model	Forecast Techniques
Sousa and Machado (2012)	Total	Relational	No
Moon et al. (2013)	Total	Relational	No
Pippal et al. (2015)	Partial	Relational	No
Our Approach	**Total**	**Relational**	**Yes**

3. Our Approach

The predictive load balancing service for replicated cloud databases was designed as a service to be deployed in the QoSDBC [Sousa et al. 2012] system architecture. QoSDBC was designed to provide a data-persistence solution in relational databases with the multi-tenant model of shared DBMS, covering aspects of data distribution and quality of service in computational clouds. In the multi-tenant model of shared DBMS, different application databases can share the same DBMS. Barker et al. (2012) discuss some multi-tenant models for databases reinforcing that the shared multi-tenant DBMS model expresses the best relationship between provider resource usage, performance and security. QoSDBC was thought to be a generic system architecture and it have the possibility of being instantiated for several quality strategies for SLA compliance in cloud databases. Works proposed by [Moreira et al. 2014] and [Sousa and Machado 2012] used QoSDBC to provide service quality in cloud databases, using, respectively, migration and replication strategies. An overview of the QoSDBC system architecture can be seen in Figure 1.

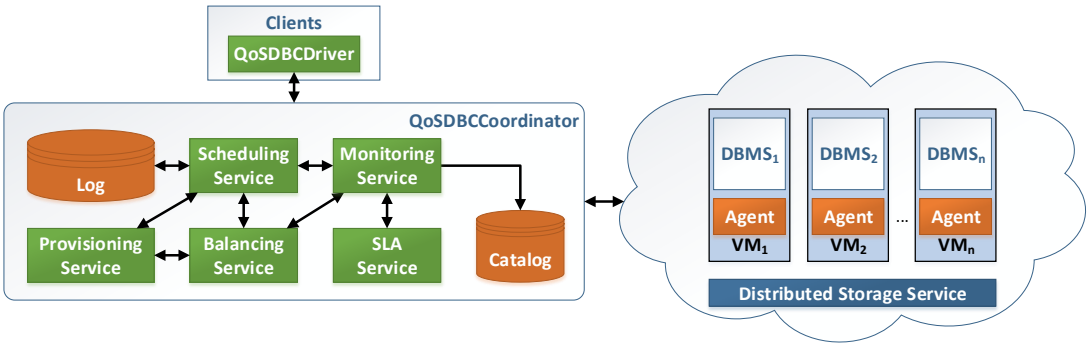


Figure 1. QoSDBC system architecture [Sousa et al. 2012]

The *QoSDBCDriver* is the component that provides access to the system, replacing the driver of a specific DBMS. It offers the same interface of communication with DBMS without the need of modifications in the DBMS. The *QoSDBCCoordinator* consists of a set of services that handle database management. The *Agent* is the component added in each VM, being responsible for collecting, monitoring and interacting with the VM and the DBMSs. The *Monitoring Service* manages the information collected by the *Agent*. This *Monitoring Service* aggregates information about processing, main and secondary memory of each VM, state of the DBMSs, workload summary, statistics etc. This information can be used to define the allocation of databases replicas in the DBMSs or to provision resources in order to guarantee the quality of service. All of this information is stored in the *Catalog* and continuously updated to fit the other services.

The *SLA Service* manages the information of the service level agreement defined between the user and the provider who has the QoSDBC installation. This information is related to SLA definitions, such as performance and consistency. This service provides parameters for the *Monitoring Service* to check the values set and adjust the other services. The *Provisioning Service* uses information from other services to provision VM resources. The *Scheduling Service* lists and selects the provisioned resources. For this, the scheduler manages the replicas, guaranteeing access to the DBMS during and after

the replication process. The *Log* stores all transactions their details that have been performed on the replicas of the system. Finally, the *Balancing Service* implements the strategy of distributing transactions between the replicas of the system. To do this, the *Scheduling Service* uses the *Balancing Service* to decide which replica to execute a transaction. Further details on the components and workflow of QoSDBC can be obtained in [Sousa et al. 2012].

The *Balancing Service* proposed in this paper uses a predictive strategy taking into account the monitoring data that is managed by the *Monitoring Service*. The prediction model adopted in this work was *AutoRegressive Integrated Moving Average* (ARIMA), since the studies carried out by [Santos et al. 2013] and [Moreira et al. 2014] showed that this forecast model has good prediction results of database workload in short prediction windows. Working with short-prediction windows helps adjust a highly dynamic environment, especially to avoid SLA violations. The *Balancing Service* requires that three parameters be configured: (i) the frequency that the prediction model should be trained and executed for each replica of the database schemas; (ii) the size of the prediction window that is returned for each replica of the database schemas; and (iii) the size of the monitoring data that must be entered into the ARIMA training step for each replica of the database schemas.

4. Preliminary Evaluation

The objective of the proposed evaluation is to verify whether the forecast techniques can be used to observe performance behavior of replicated cloud databases. For this, we will run RepliC [Sousa and Machado 2012] in a scenario with replication of cloud databases. The justifications to use RepliC are: (i) our service and RepliC's load balancing service can be implemented in the same system architecture, facilitating future comparisons between the works; (ii) both works use the same persistence model; and (iii) both works use the same replication strategy.

The OLTPBenchmark [Difallah et al. 2013] was used to create and generate database workloads in the experimental environment. OLTPBenchmark is a framework for evaluating the performance of different relational DBMSs against OLTP workload configurations. The framework has several benchmarks and with different data schemas, such as TPC-C, Twitter, YCSB, Wikipedia etc. OLTPBenchmark allows setting the time rate for submitting requests, setting the percentage of each type of transaction per experiment time, obtaining throughput information, average response time, and information on the use of operational system resources [Moreira et al. 2012]. Amazon EC2 was adopted as an environment for management of the VMs to execute the experiments. All the VMs used in the experiments have the Ubuntu Server 16.04 LTS. MySQL Server 5.7 with InnoDB engine and 128MB buffer was adopted as DBMS to manage the all databases. VMs of type *t2.small* were used to deploy databases. To host the *QoSDBCCoordinator*, a VM of type *c4.2xlarge* was used. VMs of type *t2.small* were used to run the OLTPBenchmark with *QoSDBCDriver* in order to simulate a workload for each database. All VMs were created in the same availability zone (*us-west-2b*).

The experiment scenario aims to observe the ARIMA performance when applied to a workload performed by RepliC. In this experiment, we used a *t2.small* VM to manage the databases. This VM has two databases of TPC-C, two databases of YCSB and

two databases of Wikipedia. Initially, all databases had 500MB of size. To simulate client connections, a total of 8 *t2.small* VMs were created, where each VM has 6 OLTP-Benchmark's instances. Each OLTPBenchmark's instance submits 50 connections to each database. Therefore, in total 300 connections were created for the DBMS in each *t2.small*. In total, 48 databases were created and evenly distributed in 8 *t2.small* VMs. The SLA metric used in the experiment was the average response time of the transactions and the SLA adopted for the YCSB was 60ms. The YCSB rate, per connection, was defined following the sequence: 250, 500, 750, 1000, 1000, 1000. Each transition in the sequence occurred every 10 minutes. The TPC-C and Wikipedia rates, per connection, were fixed in 10 and 100, respectively. For the use of ARIMA in the workload, one minute was inferred from every four minutes read from the RepliC monitoring system. Figure 2 shows the average response time of the workload performed in RepliC and the forecasted points inferred by ARIMA.

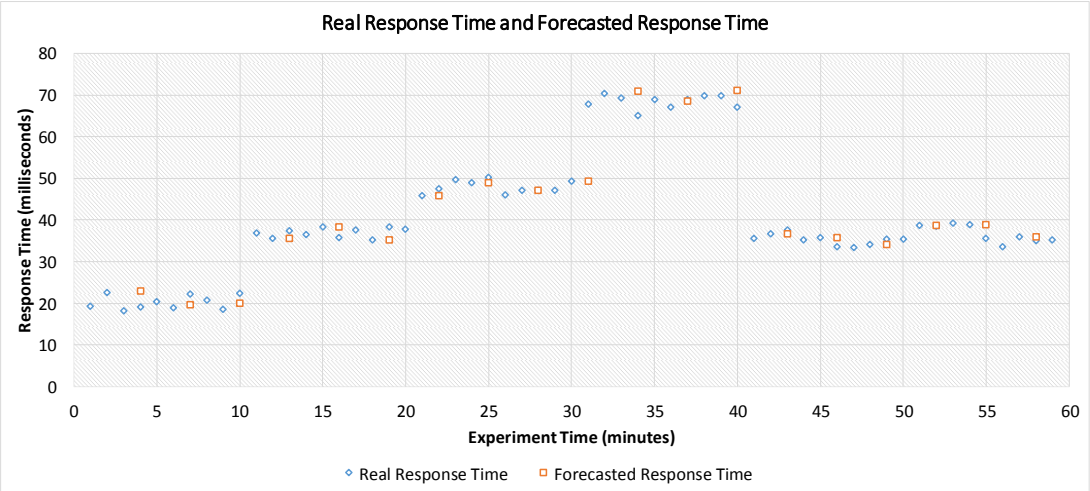


Figure 2. Real Response Time and Forecasted Response Time

After the time instant 30 minutes, RepliC detected the SLA violation for the YCSB. With this, a new *t2.small* VM was provisioned to provide a replica for the YCSB. After the availability of a new replica for the YCSB, there was a workload distribution between the replicas and the average response time decreased. From the graph shown in Figure 2 it can be seen that the forecasted points are close to the real points. Taking the points in the same timeline and plotting a time series with the real points and another one with the forecasted points, a comparison can be made to show present the accuracy level of the forecast with ARIMA. By using the *root-mean-square error* (RMSE) and *Mean Absolute Percentage Error* (MAPE) functions [Santos et al. 2013], a value can be reached which reveals the proximity of the real series with the forecasted series. Lower values of RMSE and MAPE indicate superior prediction accuracy. Calculating the values by means of the functions give the following results: RMSE = 4.925 and MAPE = 0.071. The values obtained by the RMSE and MAPE functions reinforce that ARIMA can be used in workloads for replicated databases in the cloud.

5. Conclusion

This paper presented a proposal for a predictive load balancing service for databases replicated in computational clouds. The proposed service is designed to be deployed on QoSDBC system architecture. In the design of the proposed service, the adoption of the ARIMA forecast model was considered because this model had good results for short prediction windows. Preliminary experiment was conducted and showed that ARIMA can be used to forecast workloads of replicated cloud databases. As future works we intend: (i) implement the prediction service designed in the QoSDBC system architecture; (ii) conduct deeper experiments, with a greater variety of workloads, and in several scenarios for replicating cloud databases; (iii) evaluate the performance of our approach with the use of other prediction models; and (iv) upgrade the QoSDBC system architecture to support the use of other multi-tenant and persistence models.

Acknowledgment

This research is a partial result of the project nº 455214/2014-0 supported by CNPq.

References

Barker, S., Chi, Y., Moon, H. J., Hacigümüş, H., and Shenoy, P. (2012). "cut me some slack": Latency-aware live migration for databases. In *EDBT '12*, pages 432–443, New York, NY, USA. ACM.

Difallah, D. E., Pavlo, A., Curino, C., and Cudré-Mauroux, P. (2013). Oltp-bench: An extensible testbed for benchmarking relational databases. *PVLDB*, 7(4):277–288.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. *National Institute of Standards and Technology (NIST)*.

Moon, H. J., Hacümüş, H., Chi, Y., and Hsiung, W.-P. (2013). Swat: A lightweight load balancing method for multitenant databases. In *EDBT '13*, pages 65–76, New York, NY, USA. ACM.

Moreira, L. O., Farias, V. A. E., Sousa, F. R. C., Santos, G. A. C., Maia, J. G. R., and Machado, J. C. (2014). Towards improvements on the quality of service for multi-tenant rdbms in the cloud. In *ICDE Workshops*, pages 162–169, Chicago, IL, USA.

Moreira, L. O., Sousa, F. R. C., and Machado, J. C. (2012). Analisando o desempenho de banco de dados multi-inquilino em nuvem. In *SBBD '12*, pages 161–168.

Pippal, S., Singh, S., Sachan, R. K., and Kushwaha, D. S. (2015). High availability of databases for cloud. In *INDIACom*, pages 1716–1722.

Santos, G. A. C., Maia, J. G. R., Moreira, L. O., Sousa, F. R. C., and Machado, J. C. (2013). Scale-space filtering for workload analysis and forecast. In *CLOUD '13*, pages 677–684. IEEE.

Sousa, F. R. C. and Machado, J. C. (2012). Towards elastic multi-tenant database replication with quality of service. In *UCC '12*, pages 168–175. IEEE.

Sousa, F. R. C., Moreira, L. O., Santos, G. A. C., and Machado, J. C. (2012). Quality of service for database in the cloud. In *CLOSER '12*, pages 595–601.

paper:174646

Análise da Formação e Evolução de Times de Desenvolvimento no Hibernate-ORM

Talita S. Orfanó, Michele A. Brandão, Larissa E. Maia, Mirella M. Moro

Universidade Federal de Minas Gerais - Belo Horizonte - MG

talitaorfano@ufmg.br, {micheleabrandao,larissaemanuelle,mirella}@dcc.ufmg.br

Abstract. *Various factors influence the formation and evolution of relationships in social networks. Here, we have developed a platform called AuthView to characterize the behavior of developers in Hibernate-ORM and how their relationships evolve. The results show that (1) developers tend to start contributing to a project from specific files; (2) small developers tend to share knowledge of a project module always with other major developers; and (3) there is no formation of small internal development teams.*

Resumo. *Diversos fatores influenciam a formação e evolução das relações em redes sociais. Aqui, desenvolvemos uma ferramenta chamada AuthView para caracterizar o comportamento de desenvolvedores no Hibernate-ORM e como suas relações evoluem. Os resultados mostram que (1) desenvolvedores tendem a iniciar contribuição no projeto a partir de arquivos pontuais; (2) pequenos desenvolvedores tendem a compartilhar o conhecimento de um módulo do projeto sempre com outros grandes desenvolvedores; e (3) não há a formação de pequenos times internos de desenvolvimento.*

1. Introdução

O termo *social coding* refere-se ao desenvolvimento de software baseado no compartilhamento de conhecimento entre desenvolvedores. Com a disseminação dessa prática e a popularização dos projetos *Open Source*, as redes sociais de colaboração, como o GitHub[1], representam produtividade, praticidade e aproximam desenvolvedores de diferentes localidades [Tsay et al. 2014]. O GitHub é considerado um dos maiores repositórios de projetos comerciais e pessoais para controle de versão e colaboração.

Nesse contexto, este trabalho propõe um estudo de um repositório do GitHub para compreender as características dos relacionamentos entre os desenvolvedores e a formação dos times. Tal objetivo é dividido em três questões de pesquisa: *(Q1)* Como os desenvolvedores começam a contribuir no projeto? *(Q2)* Como os membros compartilham entre si o conhecimento do sistema no decorrer do tempo? *(Q3)* Há a formação de times menores no projeto? Para responder a essas questões, os arquivos (.java) e *commits* dos desenvolvedores responsáveis pelo repositório do Hibernate-ORM [2] foram analisados. Buscou-se compreender a estrutura desses times a nível de classes, analisando desde o surgimento, a formação e evolução dos relacionamentos entre os desenvolvedores. Para realizar tais análises, desenvolvemos uma ferramenta chamada *AuthView* (*Auth*

[1]GitHub: https://github.com/

[2]Hibernate-ORM: https://github.com/hibernate/hibernate-orm

de *Authorship* por considerar autoria de software feita pelos desenvolvedores e *View* por possibilitar a visualização da rede de colaboração). Tal ferramenta permite a análise das contribuições de cada desenvolvedor por meio da coleta dos arquivos de cada *commit* das versões examinadas do repositório.

Entender a formação e evolução de times de desenvolvimento é importante para detectar como as características dos desenvolvedores estão relacionadas aos softwares e seus respectivos componentes. Trabalhos recentes apontam uma significativa relação entre os fatores humanos e a qualidade do software: a falta de responsabilidade e domínio sobre o código podem reduzir sua qualidade [Greiler et al. 2015].

O trabalho está organizado da seguinte forma: a Seção 2 descreve os trabalhos relacionados. Em seguida, a Seção 3 detalha a metodologia e a Seção 4 apresenta a ferramenta *AuthView* e suas principais funcionalidades. Os resultados são mostrados na Seção 5. Finalmente, a Seção 6 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Características sociais revelam a complexidade dos relacionamentos entre desenvolvedores. Nesse contexto, Tsay et al. [2014] investigam características sociais que influenciam contribuições em projetos *open source* no GitHub. Além disso, Casalnuovo et al. [2015] estudam como conexões sociais anteriores afetam a produtividade e o surgimento de times. Dabbish et al. [2012] mostra que usuários utilizam informações no GitHub para inferir sobre as habilidades e o conhecimento de desenvolvedores, bem como decidir se os mesmos podem iniciar colaboração no projeto.

Ademais, compreender o quanto desenvolvedores conhecem de um software facilita para os gestores manter as pessoas responsáveis por tarefas com as quais possuem conhecimento e encontrar especialistas para tarefas específicas [Rahman & Devanbu 2011]. Além disso, analisar repositórios no GitHub revela a existência de perfis de comportamentos de desenvolvedores em diferentes linguagens de programação [Rocha et al. 2016].

Em uma abordagem próxima a adotada neste trabalho, Alves et al. [2016] e Batista et al. [2017] analisam a força dos relacionamentos no GitHub a nível de repositório, direcionando o estudo em repositórios desenvolvidos majoritariamente em JavaScript e Ruby. Tais estudos mostram que a maioria dos desenvolvedores são ativos em poucos repositórios. Isso revela a necessidade de uma pesquisa que analise níveis mais profundos de abstração em um determinado repositório.

Assim, este trabalho visa suprir a necessidade levantada por [Alves et al. 2016; Batista et al. 2017]. Este estudo inicia uma análise na formação de times de desenvolvimento, buscando compreender em uma granularidade de arquivos Java. Ademais, analisamos como os desenvolvedores interagem entre si, considerando suas contribuições no decorrer do tempo de vida do projeto.

3. Metodologia

Este artigo analisa a formação e evolução de times de desenvolvimento no GitHub por meio do estudo do Hibernate-ORM, que é um *framework open source* desenvolvido em Java. Escolhemos tal projeto devido a quantidade de desenvolvedores que contribuíram, a quantidade de arquivos fonte existentes que compõem o sistema, a estabilidade no ciclo

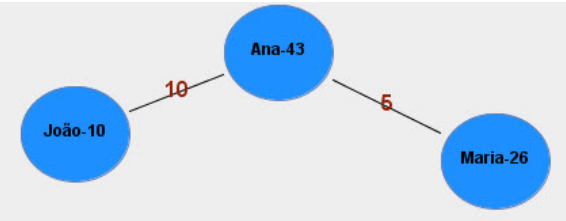


Figura 1: Representação da interação entre os desenvolvedores e suas contribuições.

evolutivo do projeto, a presença de diversas versões do projeto no repositório do GitHub e a sua relevância no cenário de tecnologia da informação.

Para compreender a distribuição dos membros de um time, oito versões (*tags*) do Hibernate-ORM foram analisadas, totalizando 2.195 *commits*, 29 desenvolvedores e mais de 8 mil arquivos. Investigou-se como os desenvolvedores interagem entre si em cada versão. A interação entre os membros acontece quando dois desenvolvedores alteram um mesmo arquivo em uma mesma versão. Por exemplo, dois desenvolvedores A e B que realizaram *commits* em cinco arquivos em comum possuem uma relação mais forte do que C e D que fizeram *commits* em apenas dois arquivos em comum em uma versão.

4. *AuthView* - uma Ferramenta para Analisar Arquivos de Commits

A ferramenta *AuthView* possui quatro funcionalidades principais: a coleta de dados, o refinamento e armazenamento dos dados, a visualização da distribuição dos desenvolvedores em cada versão e a análise comparativa entre versões do repositório. A ferramenta foi criada para permitir futuras análises. Por isso, ela possibilita a análise de qualquer repositório do GitHub, sem restrições de linguagem de programação ou tamanho.

Coleta de dados. Durante essa etapa são extraídas informações sobre todos os arquivos e os correspondentes desenvolvedores. Dessa forma, foram coletados dados de todos os *commits* de cada versão (*tag*) selecionada. Alguns critérios foram adotados e padronizados durante todas as coletas. Foram excluídos da análise os *commits* relativos à exclusão de arquivos, visto que os desenvolvedores não modificaram efetivamente o arquivo deletado. Por isso, foram considerados apenas *commits* relativos à criação e alteração de arquivos. Todos os arquivos recuperados nessa coleta possuem extensão .java. Para melhor analisar a evolução das relações entre desenvolvedores entre versões, consideramos o *commit* realizado pela primeira vez, ou seja, os *commits* não são cumulativos entre as *tags*. Assim, um *commit* realizado na *tag* 1.1 é desconsiderado ao analisar a *tag* 1.2. Esse refinamento possibilita uma visão da evolução da interação entre os desenvolvedores no decorrer da evolução do projeto.

Armazenamento dos dados. Após coletar e refinar os dados provenientes do GitHub, fez-se necessário seu armazenamento para análises posteriores. Dentre as informações armazenadas estão o nome e a URL do repositório, o nome e o caminho dos arquivos, as versões analisadas, os *commits* de cada versão, os desenvolvedores responsáveis pelos *commits* e os arquivos modificados ou criados em cada versão.

Visualização da distribuição de desenvolvedores. Nesta etapa, a ferramenta possibilita a análise de uma determinada versão do repositório. As informações para análise são o

Tabela 1: Top 10 novos desenvolvedores no Hibernate-ORM. Para garantir a privacidade dos desenvolvedores, não apresentamos o nome real.

Autor	# interações(desenvolvedores)	# contribuições(arquivos)
Desenvolvedor1	3	10
Desenvolvedor2	3	4
Desenvolvedor3	-	16
Desenvolvedor4	1	17
Desenvolvedor5	-	26
Desenvolvedor6	-	1
Desenvolvedor7	1	5
Desenvolvedor8	1	6
Desenvolvedor9	2	3
Desenvolvedor10	2	4

número total de *commits* realizados na versão (*tag*), o total de arquivos (.java) criados ou atualizados e o número de desenvolvedores que realizaram contribuições em arquivos na versão em questão. Para compreender a relação entre os desenvolvedores, desenvolvemos uma funcionalidade que propicia a visualização da relação entre os diferentes desenvolvedores que contribuíram nos arquivos atualizados na versão em análise. Na visualização apresentada na Figura 1, os nós são os desenvolvedores e as arestas constituem a relação entre dois desenvolvedores. Por meio desse recurso, é possível observar a quantidade de arquivos que cada autor contribuiu e a força de interação entre os desenvolvedores. Em cada vértice tem-se a quantidade de arquivos que os dois desenvolvedores contribuíram em comum nessa versão. A Figura 1 também mostra que a interação entre Ana e João é maior do que entre Ana e Maria, visto que eles contribuíram em 10 arquivos em comum. Por outro lado, João e Maria não tiveram nenhuma interação durante essa versão do projeto, pois não houve nenhuma interseção entre suas contribuições.

Análise comparativa. Esta funcionalidade da *AuthView* assemelha-se com a apresentada na Figura 1. Nesta última etapa, é possível realizar uma análise comparativa entre duas versões de um mesmo repositório. Dentre as informações apresentadas pela ferramenta, tem-se o número de desenvolvedores que entraram e saíram do projeto nas versões comparadas. O número de desenvolvedores que entraram corresponde ao total de desenvolvedores que não contribuíram na versão 1, mas contribuíram na versão 2. O número de desenvolvedores que saíram, representam o número de desenvolvedores que contribuíram na versão 1, mas não realizaram contribuições na segunda versão analisada. Além disso, também é possível comparar a visualização da distribuição dos desenvolvedores nas duas versões e os dados sumarizados de *commits*, arquivos e desenvolvedores em cada versão.

5. Resultados

Nesta seção, apresentamos os resultados para as três questões de pesquisa. Este trabalho considera oito versões do Hibernate-ORM a partir da versão 3.6.0.Beta1, que é a primeira presente no repositório deste projeto no GitHub, até a versão 3.6.1.Final.

***(Q1)* Como os desenvolvedores começam a contribuir no projeto?** Para responder esta pergunta, foi realizada uma análise sobre a interação entre 29 desenvolvedores. Dentre

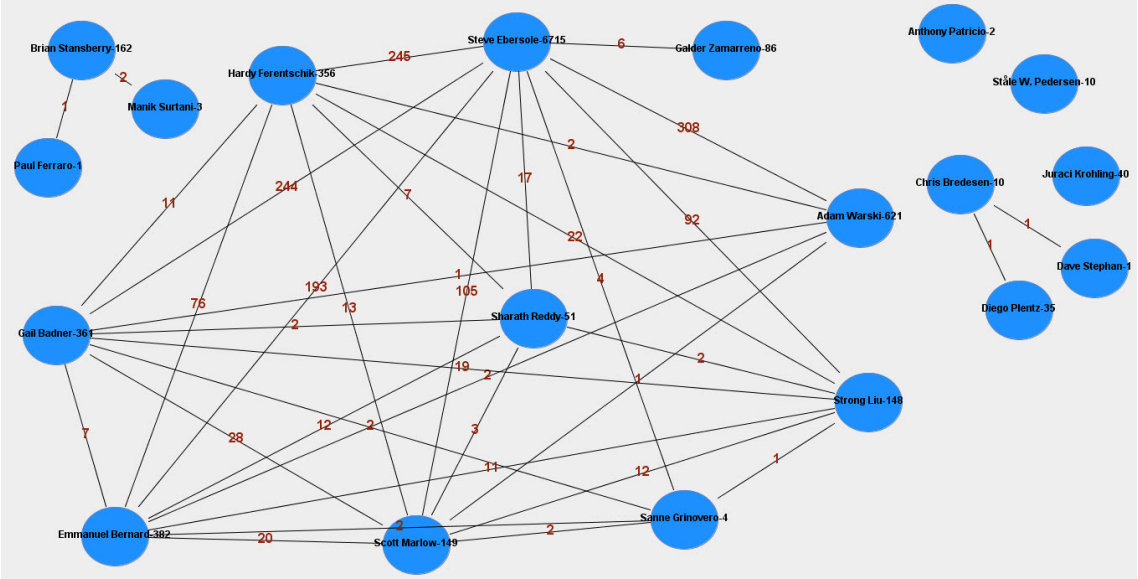


Figura 2: Visualização dos desenvolvedores na versão 3.6.0.Beta1 do Hibernate-ORM.

esse total, apenas 19 desenvolvedores realizam contribuições na primeira versão analisada do projeto. A identificação dos novos autores é feita por meio da análise comparativa entre os autores que contribuíram na versão *i* e aqueles que contribuíram na versão *i+1*. Para responder a essa questão, avaliamos o comportamento dos 10 desenvolvedores que ingressaram no projeto após a versão inicial. A Tabela 1 mostra que os novos desenvolvedores tendem a interagir com poucos desenvolvedores e a realizar poucas contribuições (medida pela quantidade de arquivos modificados e/ou criados) ao ingressar no projeto.

***(Q2)* Como os membros compartilham entre si o conhecimento do sistema no decorrer do tempo?** Para responder esta questão, foi analisado como a interação entre os desenvolvedores mantém-se nas versões analisadas. No entanto, nas oito versões analisadas, apenas três versões revelam interações entre os desenvolvedores. As versões sem interação entre os desenvolvedores são versões pequenas, possivelmente originadas para corrigir problemas pontuais no projeto sem o desenvolvimento de novas funcionalidades. Isso pode ser melhor compreendido pela quantidade reduzida de arquivos em que cada autor contribuiu nesses casos. Nas versões em que foi possível constatar interações entre os desenvolvedores, percebeu-se notoriamente uma distinção entre o comportamento dos pequenos e grandes desenvolvedores (também chamados de *heroes*). Os pequenos desenvolvedores, isto é, aqueles que realizam poucas contribuições no projeto, se relacionam em geral com um ou dois desenvolvedores, enquanto os *heroes*, aqueles que realizam muitas contribuições no projeto, possuem número elevado de interações entre desenvolvedores, que variam entre pequenos desenvolvedores e outros *heroes*.

***(Q3)* Há a formação de times menores no projeto?** Para responder essa questão, observou-se o comportamento de todos os desenvolvedores nas oito versões. Nesta etapa, a funcionalidade de visualização da rede de desenvolvedores permite observar a existência de alguns pequenos grupos de aproximadamente três desenvolvedores que possuem interações em comum. Entretanto, tais pequenos grupos não podem ser considerados times de desenvolvimento, pois o número de arquivos que esses desenvolvedores interagiram é

pequeno. Em contrapartida, observou-se uma grande interação entre a maioria dos desenvolvedores, conforme mostra a Figura 2. Isso mostra que não há pequenos times dentro do projeto, mas um único e grande time de desenvolvimento bem conectados.

6. Conclusões

Este trabalho analisou a formação e evolução dos times de desenvolvimento do GitHub, utilizando como caso de estudo o repositório Hibernate-ORM. A *AuthView* foi criada para auxiliar na coleta, armazenamento, análise e comparação dos dados. A análise buscou responder três questões de pesquisa. A primeira questão revela que os desenvolvedores tendem a começar a contribuir no projeto a partir de arquivos pontuais e não se relacionam de forma significativa com o grande time de desenvolvimento do projeto. A segunda questão apresenta como o conhecimento entre os desenvolvedores é distribuído no decorrer do tempo, percebe-se que os grandes desenvolvedores concentram a maior parcela do conhecimento do software, mas distribui esse conhecimento entre si. Por outro lado, um pequeno autor tende a compartilhar o conhecimento de um módulo do projeto sempre com um outro *heroe*. A terceira questão de pesquisa estuda como os desenvolvedores estão organizados no Hibernate-ORM. Percebeu-se que não há a formação de pequenos times internos de desenvolvimento, mas existe um grande time de desenvolvedores responsáveis pelo projeto conectados entre si. Como trabalhos futuros, pretendemos analisar mais versões do Hibernate-ORM, considerando *major* e *minor releases*, avaliar repositórios de diferentes linguagens de programação, bem como analisar as contribuições e o comportamento dos grandes desenvolvedores em outros repositórios do GitHub.

Agradecimentos. Trabalho parcialmente financiado por CAPES, CNPq e FAPEMIG.

Referências

Alves et al., G. B. (2016). The strength of social coding collaboration on github. In *Anais do SBBD*, pages 247–252, Salvador, Brasil.

Batista et al., A. N. (2017). Collaboration strength metrics and analyses on github. In *IEEE/WIC/ACM WI*, pages 1277–1286, Leipzig, Alemanhã.

Casalnuovo et al., C. (2015). Developer onboarding in github: the role of prior social links and language experience. In *ESEC/FSE*, pages 817–828, Bergamo, Itália.

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. (2012). Social coding in github: transparency and collaboration in an open software repository. In *CSCW*, pages 1277–1286, Seattle, USA.

Greiler, M., Herzig, K., and Czerwonka, J. (2015). Code ownership and software quality: a replication study. In *MSR*, pages 2–12, Florença, Itália.

Rahman, F. and Devanbu, P. (2011). Ownership, experience and defects: a fine-grained study of authorship. In *ICSE*, pages 491–500, Honolulu, USA.

Rocha, L. M. A., Silva, T. H. P., and Moro, M. M. (2016). Análise da contribuição para código entre repositórios do github. In *Anais do SBBD*, pages 103–108, Salvador, Brasil.

Tsay, J., Dabbish, L., and Herbsleb, J. (2014). Influence of social and technical factors for evaluating contribution in github. In *ICSE*, pages 356–366, Hyderabad, Índia.

paper:174142

Análise de Dados Científicos: uma Análise Comparativa de Dados de Simulações Computacionais *

Thaylon Guedes[1], **Vítor Silva**[2], **José Camata**[2,3], **Marta Mattoso**[2], **Daniel de Oliveira**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (IC/UFF)

[2]COPPE - Universidade Federal do Rio de Janeiro

[3]Centro de Computação de Alto Desempenho e Departamento de Engenharia Civil

{thaylongs, danielcmo}@ic.uff.br, {silva, marta}@cos.ufrj.br, camata@nacad.ufrj.br

Resumo. *Os avanços nas simulações computacionais têm permitido o processamento de volumes de dados cada vez maiores. Para representar as estruturas de dados complexas inerentes de tais simulações, elas são armazenadas em arquivos de formatos heterogêneos. Carregar tais dados em um SGBD, como o SciDB, para apoiar as análises deles se torna uma tarefa complexa, ou mesmo inviável, devido ao seu volume e/ou estrutura. Para evitar esse carregamento, existem abordagens que realizam consultas adaptativas e/ou que indexam os arquivos. Escolher a mais adequada pode não ser trivial. Neste artigo realizamos uma análise comparativa em termos de desempenho das abordagens de consulta de dados produzidos por uma simulação em dinâmica de fluídos computacional.*

1. Introdução

Diante dos avanços nas simulações computacionais, o volume de dados produzidos pelos programas de simulação tem aumentado cada vez mais. Esses dados representam resultados parciais e finais, sendo comumente armazenados em arquivos, que assumem diferentes formatos de acordo com o domínio da aplicação (*e.g.*, XDMF e HDF5). Neste artigo, nomeamos os dados representados em sua forma bruta simplesmente como dados científicos. Nesse cenário, para apoiar a análise de dados científicos, os conteúdos dos arquivos precisam ser acessados, extraídos e indexados, a fim de que sejam carregados em um repositório externo ou uma base de dados para permitir a análise. Existem diversas abordagens que lidam com o acesso, a extração e a indexação de dados científicos envolvidos em simulações computacionais, mantendo esses dados em seu formato original [Blanas et al. 2014]. Essas abordagens comumente acessam dados específicos de domínio em arquivos por meio das análises léxica e sintática, depois extraem esses dados de acordo os atributos de interesse do usuário por meio de um catálogo de dados (específico para cada formato de arquivo de acordo com o domínio da simulação) e indexam esses dados usando linguagens específicas de consulta e APIs. A indexação dos dados científicos em seu formato original permite que o volume de dados carregados em um repositório externo seja reduzido, e, consequentemente, a sobrecarga em termos de tempo também deve ser reduzida. Ademais, essas soluções aliviam o trabalho dos usuários quanto ao desenvolvimento de seus próprios códigos para cada tipo de análise, ao propor recursos para o processamento de consultas. Como exemplos, temos o FastBit [1], o FastQuery [2] e o NoDB [3].

*Este artigo foi financiado parcialmente pelo CNPq, CAPES e FAPERJ

[1]https://sdm.lbl.gov/fastbit

[2]http://www-vis.lbl.gov/Events/SC05/HDF5FastQuery

[3]http://dias.epfl.ch/nodb

Entretanto, essas soluções são baseadas na gerência de dados em arquivos isolados (*i.e.* um arquivo não tem relação com outros produzidos) [Silva et al. 2017, Karpathiotakis et al. 2014]. Porém, em simulações computacionais, é produzida uma série de arquivos que possuem associação entre si. Para gerenciar os dados envolvidos ao longo das simulações, registros do histórico dos dados consumidos e produzidos pelos programas de simulação (*i.e.*, dados de proveniência [Freire et al. 2008]) são capturados. Ademais, os elementos de dados (conteúdo dos arquivos) capturados precisam que seus relacionamentos (ou dependências de dados) com elementos de dados obtidos de outros arquivos sejam representados [Silva et al. 2017]. Logo, a análise exploratória de dados científicos em simulações computacionais envolve tanto a gerência dos dados científicos, como dos dados de proveniência. Recentemente, foi proposto o modelo PROV-Df [Silva et al. 2017], que segue o padrão definido pelo W3C PROV [Moreau and Groth 2013], para apoiar a gerência do fluxo de arquivos e de elementos de dados ao longo da execução de simulações, integrando dados de proveniência, de execução, e científicos.

Este artigo considera a captura e a indexação de dados científicos a partir de arquivos, assim como a carga dos índices gerados em uma base de dados de acordo com o modelo PROV-Df. Mais especificamente, este artigo propõe uma análise comparativa de técnicas de indexação de dados científicos usando 3 abordagens: FastBit, NoDB e índice posicional. Como *baseline*, comparamos as técnicas de indexação com a carga de todos os elementos de dados (*i.e.*, valores dos atributos de interesse extraídos) em uma base de dados relacional (usando PostgreSQL). Além desta introdução, este artigo contém outras 4 seções. A seção 2 apresenta abordagens existentes para a indexação de dados científicos. A seção 3 discute o estudo de caso baseado em uma simulação em dinâmica de fluidos computacional, assim como o volume e a natureza dos dados manipulados. A seção 4 apresenta a análise comparativa realizada. Por último, a seção 5 conclui esse trabalho e apresenta os trabalhos futuros.

2. Abordagens Existentes para a Indexação de Dados Científicos

Existem diferentes técnicas baseadas na indexação de dados científicos, sendo que duas propostas de índices têm sido largamente utilizadas: mapa de *bits* (*bitmap*) e posicional. A indexação baseada em um mapa de *bits* consiste na análise do domínio de determinados atributos presentes em dados científicos para gerar índices booleanos por meio da verificação de equações ou inequações algébricas. Por exemplo, assumindo-se que os valores do atributo *X* para um arquivo de dados científicos apresentam um domínio com apenas 5 valores possíveis na estrutura de dados de inteiro, então o mapa de *bits* necessita de cinco colunas para identificar a presença ou não de um valor desse atributo por meio de uma equação algébrica. Existem casos também em que o uso de inequações para a indexação *bitmap* é mais vantajoso, principalmente em cenários em que o domínio de um determinado atributo é muito extenso. O FastBit e o FastQuery são exemplos de sistemas que realizam a indexação de dados em mapas de *bits*. Mais especificamente, o FastBit propõe um conjunto de variações nas técnica de indexação, permitindo o ajuste do tipo de codificação utilizado; do algoritmo de compressão; e do nível de precisão dos valores numéricos para realizar o mapa de *bits*.

Outra técnica de indexação de dados científicos considera a geração de índices posicionais, sendo que estes utilizam informações que facilitam a localização (*i.e.*, posição) dos dados científicos em arquivos. Em comparação à indexação *bitmap*, os índices posicionais podem fornecer uma sobrecarga de dados menor, pois estes não utilizam *bits* redundantes (*i.e.*, sequência de zeros no mapa de *bits*) que crescem de acordo com o domínio dos atributos indexados. Além disso, o índice posicional permite referenciar estruturas

de dados mais complexas, como árvores e malhas. O NoDB [Karpathiotakis et al. 2014] é um exemplo de sistema que emprega índices posicionais e realiza a extração de dados de domínio presentes em arquivos, carregando-os em uma versão modificada do SGBD PostgreSQL, conhecido como PostgresRaw. Apesar de evitar mudanças nas estruturas de dados adotadas pelos arquivos, o NoDB baseia-se no processamento de consultas adaptativas, que utiliza estatísticas e estratégias de *caching* para ter acesso aos dados científicos e realizar as transformações de dados estritamente necessárias. No NoDB apenas os dados científicos necessários pelas consultas são efetivamente indexados. Entretanto, como limitação, o NoDB realiza a carga dos dados científicos acessados no PostgresRaw, o que caracteriza uma sobrecarga de armazenamento dos dados.

Entretanto, os trabalhos discutidos focam apenas em arquivos "isolados", ou seja, eles não permitem análises sobre o fluxo de dados. Para que o fluxo fosse representado nos SGBDs adaptados aos dados científicos, os programas de simulação que geram os dados teriam que ser reescritos, mudando todo o seu acesso às estruturas de dados para o acesso ao SGBD, algo que não seria viável no panorama atual de programas de simulação computacional. A outra alternativa seria manter os dados em seus formatos originais e replicá-los nos sistemas de gerência e análise de dados científicos, algo que não se mostra viável com o NoDB. Dessa forma, conforme mencionado anteriormente, os trabalhos existentes não permitem ainda a gerência dos diversos arquivos envolvidos e dos elementos de dados relacionados pelos programas que compõem a simulação computacional.

3. Estudo de Caso: Dinâmica de Fluidos Computacional

O estudo de caso desse artigo é uma simulação no domínio de dinâmica de fluidos computacional, focada em um problema de análise da deposição de sedimentos em bacias sedimentares. Esse estudo de caso foi utilizado pelo fato de envolver a geração de um grande volume de dados científicos armazenados em formatos específicos do domínio (XDMF e HDF5), sendo que os usuários necessitam do conteúdo desses arquivos para apoiar as suas análises (validação de hipóteses científicas). Especificamente, os experimentos realizados utilizaram o resolvedor libMesh-sedimentation [Silva et al. 2016], que consiste em uma aplicação construída a partir da biblioteca libMesh [4] para simular correntes turbidíticas, tipicamente encontradas em processos geológicos. Nesse caso, os sedimentos são transportados devido à movimentação do fluido, e são representados por meio de equações de Navier-Stokes de fluido incompressível combinado com uma equação de transporte dominada por advecção (concentração de sedimentos) [Guerra et al. 2016].

Nessa simulação computacional, os resultados parciais e finais das malhas são frequentemente armazenadas em arquivos de dados científicos no formato XDMF e HDF5, enquanto outros dados de interesse são mantidos em estruturas de dados em memória (*i.e.*, como variáveis no código do resolvedor). Entretanto, para capturar os dados em memória, o libMesh-sedimentation foi integrado ao ParaView Catalyst [Ayachit et al. 2015] para capturar esses dados científicos da malha simulada pelo resolvedor e armazená-los em arquivos no formato CSV. Esses arquivos no formato CSV apresentam informações como a direção de movimentação do fluido, a concentração de sedimentos em suspensão e depositados, as coordenadas do ponto na malha, entre outras.

A Figura 1 apresenta as transformações de dados envolvidas no libMesh-sedimentation, na cor laranja, assim como as transformações de dados usando o ParaView Catalyst, na cor azul, que envolveram a geração de dados científicos capturados nesse experimento. Na

[4] http://libmesh.github.io

análise comparativa desse artigo, considerou-se o processo de acesso, extração e indexação de dados envolvidos após a geração dos arquivos no formato CSV pelas transformações.

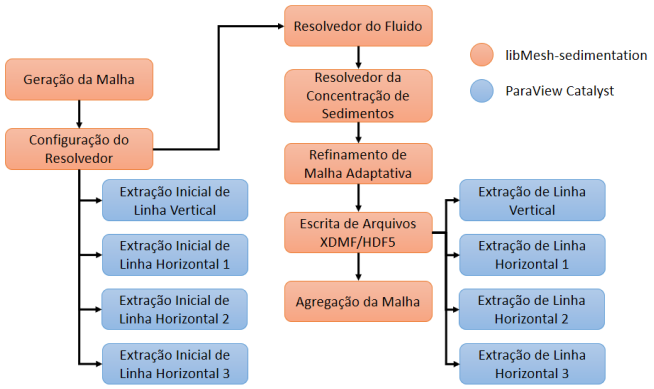


Figura 1. Simulação computacional usando libMesh-sedimentation e ParaView Catalyst

4. Análise Comparativa

Neste trabalho foram analisadas 3 abordagens de indexação: FastBit, PostgresRaw (NoDB) e índice posicional. Esta última abordagem armazena a posição do registro de acordo com atributos presentes nos arquivos de dados científicos, sendo que o seu código foi baseado na solução de indexação do NoDB. A simulação computacional do CFD foi executada no *cluster* Lobo Carneiro[5], usando 480 *cores* e produzindo aproximadamente 47 GB de dados em 97.163 arquivos. Para que seja possível realizar a indexação, os dados científicos de interesse precisaram ser acessados e extraídos dos arquivos gerados. A etapa de extração nesse experimento obteve 4.65 MB de dados presentes nos arquivos XDMF e HDF5 em 235 segundos. Uma análise comparativa dessas abordagens de indexação de dados é realizada na Seção 4.1.

Essa análise comparativa foi realizada em uma máquina local que possui um processador Intel Core i7-7700HQ com 2.80GHz e 8 GB de memória RAM. Sua unidade de armazenamento consiste em um SSD com até 1.570 MB/s de leitura e 540 MB/s de escrita. O Ubuntu 14.04.02 LTS foi utilizado como sistema operacional com o FastBit na versão 2.0.3 e o PostgresRaw compilado de acordo com a última versão disponível no repositório[6]. Em relação ao último *commit* desse repositório (do dia 06/02/2017), modificou-se apenas a quantidade máxima de arquivos de dados científicos a serem indexados. Enquanto o valor padrão desse parâmetro era 50, nossa aplicação considerou a indexação de 808 arquivos.

4.1. Comparação das Diferentes Abordagens de Indexação

Entre as abordagens escolhidas, somente o FastBit dispõe de parâmetros de configuração, sendo os principais o *binning* e o *encoding*. O *binning* determina como o *bitmap* é produzido, e no caso foi utilizado o seu valor padrão. Enquanto o *encoding* manipula os mapas de *bits* para ocupar menos espaço de armazenamento em disco, sem perda de informação, ou otimizar o processamento de consultas. Neste último parâmetro utilizamos como argumento o "*interval-equality*", pois ele é recomendado para o tipo de dados científicos extraídos e indexados neste artigo, os quais apresentaram alta cardinalidade. Logo, nesse caso, favoreceu-se o desempenho no processamento de consultas, ao invés do espaço de armazenamento em disco.

[5]http://www.nacad.ufrj.br/pt/recursos/sgiicex
[6]https://github.com/HBPMedical/PostgresRAW

O primeiro item a ser comparado é o tempo necessário para realizar a indexação dos dados. A Figura 2 apresenta o tempo para indexar dados científicos extraídos de arquivos no formato XDMF e HDF5, utilizando as abordagens PostgresRaw, índice posicional e FastBit. Nesse caso, o PostgresRaw possui o menor tempo de indexação dos dados científicos, pois, ao acessar esses dados, essa ferramenta já tira proveito dos dados em memória para realizar a indexação dos mesmos. Apesar de apresentar o mesmo comportamento (ou seja, geração de índices posicionais) que a abordagem de índice posicional, o PostgresRaw difere ao apresentar uma política de procrastinar a indexação de dados. Assim, o PostgresRaw utiliza uma abordagem adaptativa, em que os índices só são gerados, quando solicitados pelas consultas submetidas. Consequentemente, nas primeiras consultas submetidas para acessar os dados científicos em um arquivo, espera-se que o PostgresRaw apresente um tempo de processamento das consultas maior, pois os índices posicionais ainda serão gerados.

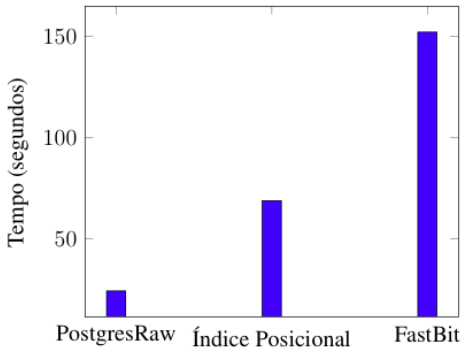


Figura 2. Comparação do tempo de indexação para as três abordagens.

Ainda na Figura 2 pode ser observado que o tempo de execução do FastBit é mais que o dobro do índice posicional. Entretanto, o FastBit responde as consultas com intervalos muito mais rápido do que no índice posicional, uma vez que, no índice posicional, o processamento de consultas requer a varredura de cada linha e coluna, assim como a conversão dos índices para os valores dos atributos (*i.e.*, dados científicos), de acordo com a estrutura de dados mais adequada. Por outro lado, o FastBit apresenta um desempenho melhor para realizar consultas em função da operação binária de *AND* no mapa de *bits* de acordo com as colunas que devem ser analisadas ou investigadas.

No quesito de armazenamento em disco, os algoritmos não mantiveram o comportamento apresentado na Figura 2. Na Tabela 1 é apresentado o espaço de armazenamento em disco necessário para amazenar os índices gerados, em *megabytes*, por cada abordagem de indexação. Como pode ser observado, o PostgresRaw foi o que mais necessitou de espaço em disco. Isso pode ser consequência dos índices verticais e posicionais criados na fase de processamento.

Método	PostgresRaw	Índice Posicional	FastBit
Espaço em disco (MB)	26,30	7,96	18,30

Tabela 1. Espaço em disco utilizado para a indexação de dados

Ao realizar a comparação entre o FastBit e o Índice Posicional, observa-se que a velocidade de acesso é proporcional ao espaço em disco necessário para representar os índices. No FastBit, utiliza-se apenas um *bit* para determinar a existência de um determinado dado científico de interesse em uma linha da malha gerada pela aplicação de dinâmica

de fluidos, enquanto as outras colunas do *bitmap* apresentam seus valores zerados. Esse comportamento é conhecido por redundância de zeros. Consequentemente, é necessário mais espaço de armazenamento em disco para representar tal *bitmap*. Considerando-se o índice posicional, essa abordagem exige mais *bits* para descrever a posição dos dados científicos no arquivo, além do tamanho, em número de *bits*, do espaço ocupado no arquivo para armazenar esse dado.

5. Conclusão e Trabalhos Futuros

Devido ao grande volume de dados manipulados pelos programas em simulações computacionais reais, se torna inviável carregar todos os dados em um SGBD. Logo, tais dados devem ser indexados em seu formato original para que sejam passíveis de consulta. Nos experimentos deste artigo, apresentou-se uma análise dos tempos de execução das técnicas de indexação, sendo que o PostgresRaw apresentou o menor tempo de execução em relação ao FastBit e ao índice posicional. Por outro lado, essa técnica foi a que apresentou o maior custo de armazenamento em disco (26,30 MB), aproximadamente 3,3 vezes maior que o volume de dados do índice posicional (7,96 MB), que obteve o melhor resultado. Como trabalhos futuros, pretende-se executar essa simulação computacional usando um caso real mais complexo. Além disso, pretende-se analisar o desempenho do processamento de consultas usando diferentes abordagens de indexação de dados científicos. Outra proposta de trabalho futuro consiste no uso de soluções de indexação de dados científicos que tirem proveito do paralelismo, como o DiNoDB e o FastQuery.

Referências

Ayachit, U., Bauer, A., Geveci, B., O'Leary, P., Moreland, K., Fabian, N., and Mauldin, J. (2015). Paraview catalyst: Enabling in situ data analysis and visualization. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, ISAV2015, pages 25–29, New York, NY, USA. ACM.

Blanas, S., Wu, K., Byna, S., Dong, B., and Shoshani, A. (2014). Parallel data analysis directly on scientific file formats. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 385–396, New York, NY, USA. ACM.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in Science Engineering*, 10(3):11–21.

Guerra, G. M., Zio, S., Camata, J. J., Dias, J., Elias, R. N., Mattoso, M., B. Paraizo, P. L., G. A. Coutinho, A. L., and Rochinha, F. A. (2016). Uncertainty quantification in numerical simulation of particle-laden flows. *Computational Geosciences*, 20(1):265–281.

Karpathiotakis, M., Branco, M., Alagiannis, I., and Ailamaki, A. (2014). Adaptive query processing on raw data. *Proc. VLDB Endow.*, 7(12):1119–1130.

Moreau, L. and Groth, P. T. (2013). *Provenance: An Introduction to PROV*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers.

Silva, V., Camata, J., de Oliveira, D., Coutinho, A. L., Valduriez, P., and Mattoso, M. (2016). In situ data steering on sedimentation simulation with provenance data. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, SC'16.

Silva, V., Leite, J., Camata, J. J., de Oliveira, D., Coutinho, A. L., Valduriez, P., and Mattoso, M. (2017). Raw data queries during data-intensive parallel workflow execution. *Future Generation Computer Systems*, 75:402 – 422.

paper:174019

Análise do Impacto do Gerador de Conjuntos de Dados em Experimentos de Deduplicação de Dados

Levy de Souza Silva, Mirella M. Moro

Departamento de Ciência da Computação - Universidade Federal de Minas Gerais

{levysouza,mirella}@dcc.ufmg.br

Abstract. *Using tools to create synthetic datasets is the only solution for evaluating data duplication algorithms when real datasets are not available. However, the evaluation results may be affected by the diversity and levels of parameters available in such tools. Our goal is to verify which parameters and levels impact more on the results of deduplication experiments. Hence, we perform factorial projects on datasets created with the most used tool. Results show that two parameters explain the largest variation of results.*

Resumo. *Usar ferramentas para criar dados sintéticos é a solução para avaliar algoritmos de deduplicação quando bases reais não existem. Porém, os resultados da avaliação podem ser afetados pela diversidade e quantidade de parâmetros existentes. Então, nós verificamos quais parâmetros e níveis impactam mais em experimentos de deduplicação de dados. Para tal, executamos projetos fatoriais em dados criados com a ferramenta mais utilizada. Os resultados mostram que dois parâmetros explicam a maior variação dos resultados.*

1. Introdução

Deduplicação de dados é a tarefa de encontrar e remover registros duplicados (representam o mesmo objeto do mundo real) em um único banco de dados; e *record linkage* e *entity resolution* em vários bancos de dados [Christen 2012]. Via de regra, abordagens desse tipo são usadas para melhorar a qualidade e a integridade dos dados. Por exemplo, em recuperação de informação, são importantes para remover documentos duplicados (páginas *web* e citações bibliográficas) retornados por motores de busca, bibliotecas digitais e sistemas de indexação automática de texto [de Carvalho et al. 2011; Hajishirzi et al. 2010]. Com a expansão de comércio eletrônico, outra aplicação em crescimento é a identificação de produtos duplicados em sistemas de lojas *online* [Christen 2012].

Várias soluções têm sido propostas visando encontrar e remover duplicatas, incluindo abordagens de programação genética [de Carvalho et al. 2012]. Independente do mecanismo proposto, um dos problemas da avaliação experimental para tais soluções é a aquisição de *datasets* reais que contenham registros originais e duplicados. Um motivo é que muitos *datasets* do mundo real contêm informações pessoais ou sigilosas, tornando improvável a distribuição pública devido a questões de privacidade e confidencialidade.

Nesse contexto, a alternativa é usar dados gerados artificialmente para validar e avaliar as abordagens. Em resumo, os registros duplicados são criados por meio da inserção de erros e modificações em um registro original. De acordo com Ioannou et al. [2013], essas modificações podem ser sintáticas, estruturais ou semânticas (erros ortográficos, permutação de caracteres, apelidos para o mesmo objeto e abreviações). Esses

datasets têm a vantagem que os erros são previamente conhecidos, tornando os experimentos mais controlados. Entretanto, pode ser difícil aprender de forma eficiente como usar e customizar os parâmetros disponíveis nas ferramentas de modo a construir um melhor *dataset* para as avaliações experimentais.

Nos últimos anos, pelo menos 90 estudos[1] utilizam o *Data Set Generator Program*[2] [Christen 2012, Seção 7.6] para criar conjuntos de dados com registros duplicados. Porém, nenhum desses (ou demais artigos pesquisados) avalia o impacto que as configurações dos parâmetros podem ter nos experimentos de deduplicação. Igualmente, nenhum mensura quais parâmetros e níveis têm maior impacto nos resultados experimentais, aumentando ou diminuindo a significância dos resultados.

Desse modo, os experimentos de deduplicação podem ser comprometidos devido à diversidade de parâmetros e níveis disponíveis, pois os valores de *F-Measure* podem aumentar ou diminuir de acordo com a configuração adotada. Em outras palavras, estudos que utilizam esta ferramenta criando *datasets* para validar suas abordagens podem ter resultados duvidosos. Logo, identificar quais parâmetros e níveis mais influenciam os resultados dos experimentos é uma tarefa primordial para apoiar estudos e pesquisadores que usam essa ferramenta para testar suas abordagens e metodologias. Além disso, com essas informações, os experimentos podem ser realmente mais controlados, e as abordagens podem ser avaliadas em vários cenários de configurações de parâmetros.

Nesse contexto, o objetivo desse trabalho é analisar quais parâmetros do *Data Set Generator Program* têm maior impacto nos resultados dos experimentos de deduplicação de dados. Especificamente, este estudo busca responder às seguintes questões de pesquisa: (1) Qual é a melhor configuração de parâmetros a ser adotada na criação de um *dataset* sintético? (2) Quais parâmetros têm maior influência para obter resultados insignificantes de *F-Measure*? (3) Quais parâmetros não interferem nos resultados dos experimentos? (4) Quais os níveis dos parâmetros que diminuem os resultados de *F-Measure*? e (5) Qual o percentual de impacto que as configurações dos parâmetros têm nos experimentos de deduplicação de dados? Para alcançar o objetivo, nós executamos quatro projetos fatoriais 2^k [Jain 1992] em um conjunto de *datasets* criados com a ferramenta.

2. Trabalhos Relacionados

A Tabela 1 apresenta uma amostra de estudos que usam essa ferramenta conforme coletados do Google Acadêmico e ordenados pelo número de citações. Considerando os parâmetros informados (terceira coluna da tabela), nota-se que não existe um consenso sobre quais parâmetros e configurações são importantes, pois há estudos que reportam poucos [Draisbach et al. 2012] e muitos parâmetros [de Carvalho et al. 2012]. Além disso, existem estudos que também *omitem* parâmetros [Beskales et al. 2009; Draisbach et al. 2012; Steorts et al. 2014]. Desse modo, a realização do nosso trabalho é primordial, pois busca-se descobrir quais configurações de parâmetros são realmente necessárias para criação de um *dataset* e como eles interferem nos resultados dos experimentos e avaliações de deduplicação.

O *Data Set Generator Program* é uma ferramenta que cria registros originais e duplicados de acordo com os parâmetros escolhidos pelo usuário. Essa ferramenta faz parte

[1] Encontrados no Google Acadêmico em maio de 2017

[2] `https://cs.anu.edu.au/people/Peter.Christen/Febrl/febrl-0.3/febrldoc-0.3/node70.html`

Tabela 1: Visão geral dos estudos usando a ferramenta.

Referência	Contribuição do estudo	Configurações dos parâmetros	#citações
[Christen 2008]	Dois métodos de classificação. Um baseado na classificação de vizinhança e o outro melhora a classificação do SVM ao adicionar vetores de peso aos conjuntos de treinamento.	Original Records, Duplicate Records, Max of Change by Field, Max Duplicate by Records	142
[Draisbach et al. 2012]	Apresenta uma estratégia que adapta o tamanho da janela para aumentar a eficiência do processo de detecção de duplicatas sem reduzir a eficácia.	Distribution, Duplicate Records	69
[de Carvalho et al. 2012]	Uma abordagem de programação genética para deduplicação de registros que combina várias provas extraídas do conteúdo dos dados para encontrar uma função de deduplicação que é capaz de identificar se duas entradas são réplicas ou não.	Original Record, Duplicate Records, Max of Change by Field, Max Duplicate by Records, Distribution	62
[Beskales et al. 2009]	Um novo modelo de incerteza que codifica o espaço de possíveis reparações correspondentes a diferentes configurações de parâmetros.	Original Record, Duplicate Records, Distribution	39
[Steorts et al. 2014]	Comparação de métodos de blocagem para Record Linkage (Simple Alternatives to Blocking, Cluster-Based Blocking, LSH-Based Approaches).	Duplicate Records, Max of Change by Field	20
[Sadinle 2017]	Uma estimativa de Bayes que permite que partes incertas da correspondência não sejam resolvidas.	Type of Change, Original Records, Duplicate Records, Max of Change by Field	4

do projeto *Febrl*[3] e é baseada nas ideias apresentadas por Hernández e Stolfo [1995]. A ferramenta permite criar um *dataset* contendo: (1) nomes (baseado em tabelas de frequência para nomes e sobrenomes); (2) endereços (baseado em tabelas de frequência para endereços, códigos postais, números de ruas, estados e territórios); (3) datas (como datas de nascimento); (4) números de telefone; e (5) registros identificadores. Com a ferramenta é possível definir os seguintes parâmetros: *number of original records, number of duplicate records, maximum amount of duplicate by record, maximum amount of changes by field, maximum amount of changes by instances, probability distribution (uniform, poisson or zipf), type of change (type, ocr, phonetic, all), e number of household*s.

3. Experimentos e Discussões

Inicialmente, definimos quatro projetos fatoriais 2^k [Jain 1992][4] para medir o impacto das configurações de parâmetros na criação de um *dataset*. Para cada projeto fatorial, executamos oito experimentos com os *datasets* criados com a ferramenta e escolhemos três parâmetros e dois níveis para análise, isto é, analisamos um projeto fatorial 2^3. Os outros parâmetros foram configurados com valores fixos.

A identificação de registros duplicados é um processo composto de várias etapas incluindo: (1) indexação, os registros recebem um valor de chave de bloco (BKv); (2) blocagem, os registros são agrupados por BKv; e (3) comparação, os registros são comparados por meio de funções de similaridade [Christen 2012]. Desse modo, utilizando a ferramenta, criamos um *dataset* para cada configuração de projeto, sobre os quais aplicamos algoritmos de deduplicação de dados. Os algoritmos usados foram: *Soundex* para o valor da chave de bloco, *Standard Blocking* para blocagem de registros e *Jaro Winkler* para a função de similaridade [Christen 2012]. Nós indexamos os registros pelo atributo *given name* e comparamos os atributos *given name* e *surname* para encontrar as duplicatas. Nós utilizamos as métricas *Precision*, *Recall* e *F-Measure* para averiguar o impacto dos parâmetros e níveis. A Tabela 2 detalha os projetos fatoriais, os parâmetros escolhidos, os níveis dos parâmetro e os valores configurados para os outros parâmetros.

A Tabela 3 agrega as porções de efeito explicadas por cada parâmetro e suas iterações em cada projeto fatorial: (a) *primeiro* a (d) *quarto*. Observe que o valor de

[3]Projeto Febrl `http://sourceforge.net/projects/febrl`

[4]O projeto fatorial serve para analisar o impacto de fatores e níveis em uma variável resposta Y.

Tabela 2: Plano de Experimentos

Projeto Fatorial	Parâmetros Escolhidos	Níveis	Outros Parâmetros
1	Max Duplicate by Record	3 e 8	Original Records = 1.000 Duplicate Records = 100 Distribution = uniform Types of Change = all Households = 1
	Max Change by Field	2 e 7	
	Max Change by Instance	7 e 10	
2	Original Records	1.000 e 10.000	Max Duplicate by Record = 5 Max Changes by Field = 5 Max Changes by Instances = 10 Type of Change = all Households = 1
	Duplicate Records	30% e 70%	
	Probability Distribution	Uniform e Poisson	
3	Probability Distribution	Uniform e Poisson	Original Records = 1.000 Duplicate Records = 100 Max Duplicate by Record = 5 Max Changes by Field = 5 Max Changes by Instances = 10
	Type of Change	Typo e Phonetic	
	Households	3 e 8	
4	Probability Distribution	Uniform e Poisson	Original Records = 1.000 Duplicate Records = 100 Max Duplicate by Record = 5 Max Changes by Instances = 10 Households = 1
	Type of Change	Typo e Phonetic	
	Max Change by Field	2 e 7	

F-Measure é a métrica considerada. Para computar as variações, consideramos um nível para cada parâmetro como especificado na Tabela 3 e executamos oito experimentos.

A Tabela 3(a) apresenta os resultados do primeiro projeto fatorial que considera o valor de *F-Measure* obtido variando-se os parâmetros *Max Duplicate by Record*, *Max Change by Field* e *Max Change by Instance*. Observa-se que 89.65% da variação dos resultados é explicada pelo parâmetro *Max Change by Field*. Assim, a quantidade de erros que são introduzidos nos atributos dos registros duplicados contribui de forma intensa para o valor de *F-Measure* obtido nos resultados. Então, é esperado que quanto maior o número de erros introduzidos, menor a eficácia das funções de similaridade.

A Tabela 3(b) apresenta os resultados do segundo projeto fatorial que considera o valor de *F-Measure* obtido variando-se os parâmetros *Original Records*, *Duplicate Records* e *Probability Distribution*. Observa-se que 76.77% da variação dos resultados é explicada pelo parâmetro *Probability Distribution* e sua iteração com os parâmetros *Original Records* e *Duplicate Records*. Assim, o parâmetro *Probability Distribution* tem uma maior influência nos valores de *F-Measure* dos resultados dos experimentos $(33,12\%)$. Além disso, conclui-se que os resultados também são afetados pela quantidade de registros originais e registros duplicados criados, pois a iteração entre o parâmetro *Probability Distribution* e os demais é significativa (AC = 21.29% e BC = 22.36%).

A Tabela 3(c) apresenta os resultados do terceiro projeto fatorial que considera o valor de *F-Measure* obtido variando-se os parâmetros *Probability Distribution*, *Type of Change* e *Number of Household*. Observa-se que 89.79% da variação dos resultados é explicada pelo parâmetro *Type of Change*. Assim, quando o tipo de modificação é configurado para *Phonetic*, os menores valores de *F-Measure* são obtidos, como observado nas Tabelas 3(c) e 3(d). Essa função simula variações fonéticas, isto é, nomes que soam similares. A função é baseada em um arquivo que contém pares de variações fonéticas de *sub-strings*, como por exemplo, $ph \leftrightarrow f$, or $rie \leftrightarrow ry$ [Christen 2012, Seção 4.3]. Usando esta abordagem obtêm-se uma menor eficácia das funções de similaridade.

Tabela 3: Resultado dos projetos fatoriais.

(a) Primeiro

Exp.	(A) Max Duplicate by Record	(B) Max Change by Field	(C) Max Change by Instance	F-Measure	Porção de efeito explicada por cada parâmetro e suas iterações
1	3	2	7	0.6369	A = 0.62% **B= 89.65%** C = 0.93% AB = 6.70% AC = 0.58% BC = 0.64% ABC = 0.89%
2	8	2	7	0.6170	
3	3	7	7	0.6883	
4	8	7	7	0.7391	
5	3	2	10	0.6174	
6	8	2	10	0.6012	
7	3	7	10	0.7037	
8	8	7	10	0.7204	

(b) Segundo

Exp.	(A) Original Records	(B) Duplicate Records	(C) Probability Distribution	F-Measure	Porção de efeito explicada por cada parâmetro e suas iterações
1	100	30%	Uniform	0.5615	A = 0.11% B = 4.38% **C = 33.12%** AB = 4.38% **AC = 21.29%** **BC = 22.36%** ABC = 14.37%
2	1000	30%	Uniform	0.5877	
3	100	70%	Uniform	0.5786	
4	1000	70%	Uniform	0.5914	
5	100	30%	Poisson	0.6360	
6	1000	30%	Poisson	0.5959	
7	100	70%	Poisson	0.5859	
8	1000	70%	Poisson	0.5922	

(c) Terceiro

Exp.	(A) Probability Distribution	(B) Type of Change	(C) Number of Households	F-Measure	Porção de efeito explicada por cada parâmetro e suas iterações
1	Uniform	Typo	3	0.6456	A = 0.29% **B = 89.79%** C = 0.04% AB = 1.86% AC = 3.28% BC = 4.46% ABC = 0.28%
2	Poisson	Typo	3	0.6623	
3	Uniform	Phonetic	3	0.4975	
4	Poisson	Phonetic	3	0.4118	
5	Uniform	Typo	8	0.6623	
6	**Poisson**	**Typo**	**8**	**0.7485**	
7	**Uniform**	**Phonetic**	**8**	**0.3714**	
8	Poisson	Phonetic	8	0.4122	

(d) Quarto

Exp.	(A) Probability Distribution	(B) Max Change by Field	(C) Type of Change	F-Measure	Porção de efeito explicada por cada parâmetro e suas iterações
1	Uniform	2	Typo	0.7684	A = 1.60% B = 0.88% **C = 85.18%** AB = 0.58% AC = 3.26% BC = 1.44% ABC = 7.07%
2	Poisson	2	Typo	0.6914	
3	Uniform	7	Typo	0.7758	
4	**Poisson**	**7**	**Typo**	**0.8187**	
5	**Uniform**	**2**	**Phonetic**	**0.3741**	
6	Poisson	2	Phonetic	0.5789	
7	Uniform	7	Phonetic	0.4737	
8	Poisson	7	Phonetic	0.4627	

A Tabela 3(d) apresenta os resultados do quarto projeto fatorial que avalia a combinação dos parâmetros mais influentes dos projetos fatoriais um, dois e três. Observa-se que o parâmetro *Type of Change* explica 85.18% da variação dos resultados.

Considerando a primeira questão de pesquisa (i.e., a melhor configuração de parâmetros), observa-se que os maiores valores de *F-Measure* são obtidos usando os níveis da Tabela 3(d). Para a segunda questão de pesquisa (i.e., parâmetros de maior influência), conclui-se que os parâmetros *Probatility Distributiton*, *Max of Change by Field* e *Type of Change* explicam uma maior variação dos resultados, sendo o último o mais influente dos três. Para a terceira questão de pesquisa (i.e., parâmetros que não interferem), observa-se que os parâmetros *Original Records*, *Duplicate Records*, *Max Change by Instance* e *Number of Households* não influenciam muito os resultados.

Por fim, para a quarta (i.e., parâmetros que diminuem os resultados de *F-Measure*) e quinta questões de pesquisa (i.e., percentual de impacto), verifica-se que a definição dos parâmetros influencia o *F-Measure*. De acordo com as Tabelas 3(c) e 3(d), o valor de *F-Measure* variou entre 0.3714 e 0.7485 e entre 0.3741 e 0.8187 respectivamente, ficando claro que os níveis dos parâmetros influenciam nos resultados dos experimentos de deduplicação. Considerando a Tabela 3(d), por exemplo, observa-se que o valor de *F-Measure* obtido teve um aumento de 135.86% com o nível *Typo* ao invés do *Phonetic* para

o parâmetro *Type of Change*. Assim, estudos que não informam a configuração utilizada neste parâmetro podem ter resultados duvidosos (ex., [Steorts et al. 2014]).

4. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma avaliação experimental do impacto dos parâmetros e níveis na criação de um *dataset* usando o *Data Set Generator Program* em experimentos de deduplicação de dados. Foram executados quatro projetos fatoriais 2^k em conjuntos de *datasets* criados com a ferramenta. Além disso, foram discutidas questões de pesquisas relacionadas a identificar quais parâmetros e níveis interferem nos resultados dos experimentos. Nossa principal conclusão é que os parâmetros *type of change* e *max change by field* explicam uma maior variação dos resultados. Como trabalhos futuros, planejamos realizar replicações nos experimentos, calcular o intervalo de confiança dos resultados e executar testes estatísticos para validar os experimentos.

Agradecimentos. Trabalho parcialmente financiado por FAPEMIG e CNPq.

Referências

Beskales, G., Soliman, M. A., Ilyas, I. F., and Ben-David, S. (2009). Modeling and querying possible repairs in duplicate detection. *Proceedings of the VLDB Endowment*, 2(1):598–609.

Christen, P. (2008). Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *SIGKDD*, pages 151–159, Las Vegas, Nevada, USA.

Christen, P. (2012). *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, Berlin.

de Carvalho, A. P., Ferreira, A. A., Laender, A. H., and Gonçalves, M. A. (2011). Incremental unsupervised name disambiguation in cleaned digital libraries. *JIDM*, 2(3):289–304.

de Carvalho, M. G., Laender, A. H., Gonçalves, M. A., and da Silva, A. S. (2012). A genetic programming approach to record deduplication. *TKDE*, 24(3):399–412.

Draisbach, U., Naumann, F., Szott, S., and Wonneberg, O. (2012). Adaptive windows for duplicate detection. In *ICDE*, pages 1073–1083, Arlington, Virginia, USA.

Hajishirzi, H., Yih, W.-t., and Kolcz, A. (2010). Adaptive near-duplicate detection via similarity learning. In *SIGIR*, pages 419–426, Geneva, Switzerland.

Hernández, M. A. and Stolfo, S. J. (1995). The merge/purge problem for large databases. In *SIGMOD*, pages 127–138, San Jose, CA, USA.

Ioannou, E., Rassadko, N., and Velegrakis, Y. (2013). On generating benchmark data for entity matching. *Journal on Data Semantics*, 2(1):37–56.

Jain, R. (1992). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley.

Sadinle, M. (2017). Bayesian estimation of bipartite matchings for record linkage. *Journal of the American Statistical Association*, 112(518):600–612.

Steorts, R. C., Ventura, S. L., Sadinle, M., and Fienberg, S. E. (2014). A comparison of blocking methods for record linkage. In *PSD*, pages 253–268, Ibiza, Spain.

paper:174084

Aspectos Temporais para Medir a Força da Colaboração no GitHub

Natércia A. Batista, Michele A. Brandão, Ana Paula C. da Silva, Mirella M. Moro

Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

{natercia,micheleabrandao,ana.coutosilva,mirella}@dcc.ufmg.br

Abstract. *Time is fundamental to understand interactions between individuals in a social network. Here, we consider the temporal aspect to measure the collaboration strength between GitHub developers. The results show the temporal aspect enables to build a more realistic collaboration graph.*

Resumo. *Tempo é fundamental para entender a interação entre indivíduos em uma rede social. Aqui, consideramos o aspecto temporal para medir a força da colaboração entre desenvolvedores no GitHub. Os resultados revelam que tal aspecto permite construir um grafo de colaboração mais realista.*

1. Introdução

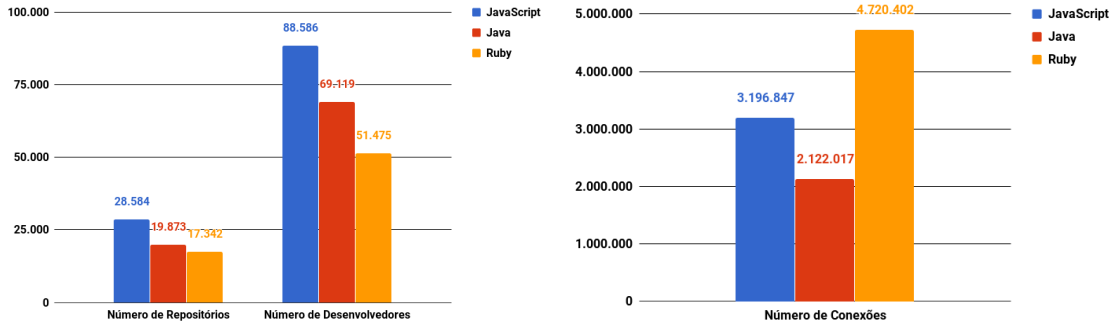
Social Coding é uma abordagem de desenvolvimento de software que propõe um ambiente colaborativo entre os desenvolvedores, encorajando a discussão e compartilhamento de ideias e conhecimento [Dabbish et al. 2012]. Essa metodologia tem alterado a forma de desenvolvimento de software e criado um novo tipo de rede de colaboração. Note que uma colaboração ocorre quando desenvolvedores contribuem em um mesmo repositório.

Trabalhos Relacionados. Estudos abordam diferentes propriedades e investigam comportamentos no GitHub [Tsay et al. 2014] (principal website de *social coding*), incluindo métricas que podem ser calculadas utilizando dados temporais e relacionamentos posteriores [Casalnuovo et al. 2015]. Porém, tais métricas são calculadas como propriedades de um projeto (repositório). Aqui, o foco está na colaboração entre pares de desenvolvedores e na melhoria de propriedades que utilizam dados temporais.

Contribuições. A medida da força dos relacionamentos pode ser utilizada para várias aplicações, como melhorar recomendação de desenvolvedores, avaliar a formação de times ou melhorar algoritmos de análise. Assim, nosso objetivo é encontrar propriedades semânticas da relação entre desenvolvedores e analisar como é possível utilizar essas propriedades para determinar a força do relacionamento entre eles. Para tal, analisamos a correlação entre métricas para a força dos relacionamentos. Essas métricas são propriedades topológicas e semânticas previamente propostas e propriedades semânticas que consideram aspectos temporais. Também discutimos como tais aspectos podem ser utilizados para modelar um grafo que representa a rede de colaboração.

2. Metodologia

Esta seção apresenta a base de dados e o modelo de rede utilizado para as análises, e a definição das métricas topológicas e semânticas para força dos relacionamentos.



(a) Repositórios e desenvolvedores (b) Conexões (links entre desenvolvedores)

Figura 1: Estatísticas para cada uma das linguagens no dataset

Base de Dados. A base provém do GHTorrent [Gousios 2013] e foi modificada e disponibilizada por Batista et al. [2017], que incluiu informações da rede do GitHub para as linguagens JavaScript e Ruby. Com a mesma metodologia de coleta, aqui também incluímos a linguagem Java. Batista et al. [2017] considera a data de início da colaboração entre dois desenvolvedores como a mais recente entre as datas dos primeiros commits de ambos em um repositório, e data de fim de colaboração como o último commit realizado no repositório por *qualquer* colaborador. Aqui, o fim da colaboração foi *alterado* para a menor data entre os últimos commits do *par* de desenvolvedores no repositório, o que permite limitar o relacionamento dentro da janela temporal de contribuição em comum.

Modelagem da Rede. A rede de desenvolvedores é representada por um grafo ponderado $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: $\mathcal{V}$ é o conjunto de nós para os desenvolvedores, e $\mathcal{E}$ é o conjunto de arestas não direcionadas que conectam dois desenvolvedores quando ambos contribuem em um mesmo repositório. O peso das arestas corresponde aos valores que podem ser calculados a partir das métricas propostas por Alves et al. [2016] e Batista et al. [2017] (apresentadas a seguir). A Figura 1 (a) mostra a quantidade de repositórios e desenvolvedores presentes na rede para cada uma das linguagens, e a Figura 1 (b) apresenta a quantidade de conexões existentes em cada uma das redes. Apesar da linguagem Java possuir o segundo maior número de repositórios e desenvolvedores, tal rede não é tão densa quanto as demais. Por outro lado, a rede da linguagem Ruby possui as menores quantidades de repositórios e desenvolvedores, mas possui a maior quantidade de conexões.

Propriedades topológicas e semânticas. A Tabela 1 apresenta as características topológicas e semânticas para avaliar como desenolvedores se relacionam.

Propriedades semânticas com aspecto temporal. A Tabela 1 apresenta três propriedades semânticas que consideram o aspecto temporal dos relacionamentos propostas por Batista et al. [2017]. LPC e GPC calculam o tempo de contribuição entre um par de desenvolvedores em um dado repositório com variações local e global, respectivamente. Após as alterações na definição da data de fim de colaboração entre desenvolvedores, os resultados para tais métricas são alterados. O cálculo de LPC pode ser enviesado, principalmente em repositórios de curta duração como exemplificado na Tabela 2. O par de desenvolvedores AB contribuiu por menos tempo que o par CD, mas possui um valor superior para a métrica. Para entender a força de colaboração baseada no tempo para todos os pares de desenvolvedores, é proposta uma visão global das colaborações de toda a rede. A métrica GPC considera o tempo de contribuição máximo entre todos os pares

Tabela 1: Dados dois nós A e B, $\mathcal{N}(A)$ e $\mathcal{N}(B)$ são os vizinhos de A e B, $w(A)$ e $w(B)$ são o grau ponderado (soma do peso de cada aresta conectada ao nó), $w(A,B)$ o peso da aresta entre A e B, e $\mathcal{R}$ o conjunto de repositórios compartilhados por A e B.

Propriedades topológicas		
Métrica	**Definição**	**Interpretação**
Neighborhood Overlap (NO)	Mede a similaridade dos vizinhos entre dois nós por: $NO_{(A,B)} = \frac{\lvert\mathcal{N}(A)\rvert\cap\lvert\mathcal{N}(B)\rvert}{\lvert\mathcal{N}(A)\rvert\cup\lvert\mathcal{N}(B)\rvert}$	De acordo com [Easley and Kleinberg 2010], NO pode ser utilizada para calcular a força de links. Quanto maior o valor de NO, mais forte é o relacionamento.
Adamic-Adar Coefficient (AA)	Customizada para este contexto e calculada por: $AA_{(A,B)} = \frac{\sum \forall z \in \lvert\mathcal{N}(A)\rvert\cap\lvert\mathcal{N}(B)\rvert}{log\lvert\mathcal{N}(z)\rvert}$.	Vizinhos que não se relacionam com muitos outros recebem maior peso.
Preferential Attachment (PA)	Quanto maior o número de vizinhos de um nó, maior seu valor de PA calculado por: $PA_{(A,B)} = \lvert\mathcal{N}(A)\rvert\lvert\mathcal{N}(B)\rvert$.	De acordo com Barabási and Albert 1999, há uma relação linear entre o número de vizinhos de um nó e a probabilidade de conexão (i.e., "o rico fica mais rico").
Propriedades semânticas (Alves et al. [2015])		
Métrica	**Definição e Interpretação**	
Number of shared repositories (SR)	Refere-se ao número de repositórios compartilhados e é definida pela cardinalidade do conjunto $\mathcal{R}$ (i.e., $SR_{(A,B)} = \lvert\mathcal{R}\rvert$)	
Jointly developers contribution to shared repositories (JCSR)	Definida por: $JCSR_{(A,B)} = \frac{\sum_{\forall r_i \in \mathcal{R}} JCSR_{(A,B,r_i)}}{\lvert\mathcal{R}\rvert}$. Exemplo: considerando dois repositórios r_1 e r_2 em que r_1 é compartilhado apenas pelos desenvolvedores A e B, a contribuição conjunta do par AB em r_1 ($JCSR_{(A,B,r_1)}$) é igual a 1. Para r_2, considerando que é compartilhado pelos desenvolvedores A, B e C, a contribuição conjunta de A e B em r_2 ($JCSR_{(A,B,r_2)}$) é 0.66. Supondo que A e B compartilham apenas r_1 e r_2, a contribuição conjunta para eles é a média dos valores para cada repositório: ($JCSR_{(A,B)}$) = 0.83.	
Jointly developers commits to shared repositories (JCOSR)	Dado que $NC_{(A,r_j)}$ é o número total de commits realizados pelo desenvolvedor A no repositório r_j, $NC_{(B,r_j)}$ é o número total de commits realizados pelo desenvolvedor B no repositório r_j, e $NC_{(r_j)}$ é o número total de commits realizados no repositório r_j independente do desenvolvedor, então: $JCOSR_{(A,B)} = \sum_{\forall r_i \in \mathcal{R}} \frac{(NC_{(A,r_i)}+NC_{(B,r_i)})}{NC_{(r_i)}}$.	
Propriedades semânticas com aspecto temporal (Batista et al. [2017])		
Métrica	**Definição e Interpretação**	
Local Potential Contribution (LPC)	Considerando que $T_{(A,B,r_i)}$ é o intervalo de tempo em que ambos os desenvolvedores contribuíram em um repositório r_i definido pelas datas de início e fim de colaboração (representadas pelo primeiro e último commit em um repositório), esta métrica pode ser calculada por $LPC_{(A,B)} = \frac{\sum_{\forall r_i \in \mathcal{R}} \frac{T_{(A,B,r_i)}}{T_{(r_i)}}}{\lvert\mathcal{R}\rvert}$.	
Global Potential Contribution (GPC)	Para compensar o viés no cálculo da métrica LPC, Batista et al. [2017] estenderam para uma versão global definida como: $GPC_{(A,B)} = \frac{\sum_{\forall r_i \in \mathcal{R}} T_{(D_1,D_2,r_i)}}{\max_{\forall (D_1,D_2)\in \mathcal{D},\ r_i \in \mathcal{R}} T(D_1,D_2,r_i)}$. onde D é o conjunto de desenvolvedores total na rede, e $T_{(A,B,r_i)}$ é o intervalo de tempo em dois desenvolvedores contribuíram num repositório comum r_i.	
Previous Collaboration (PC)	Contabiliza as colaborações prévias entre dado par de desenvolvedores. Assim, no tempo t é definida por: $PC_{(A,B,t)} = \frac{\sum_{\forall r_i \in \mathcal{R}} \frac{1}{ND_{(r_i,t)}}}{\lvert\mathcal{R}\rvert}$, onde $ND_{(r_i,t)}$ é o número total de desenvolvedores que contribuem no repositório r_i no tempo t, antes do desenvolvedor B iniciar a colaboração em r_i. Altos valores da fração $1/ND_{(r_i,t)}$ indicam que A tem mais chances de trabalhar com B e vice-versa. Por exemplo, se há apenas duas pessoas em dado repositório, há apenas uma possibilidade de colaboração; em contrapartida, quanto mais colaboradores no mesmo repositório, mais opções de escolha e assim, menor é a possibilidade de colaboração com um desenvolvedor em particular. Ou seja, existe maior possibilidade de estabelecer uma conexão com um desenvolvedor se a atenção do mesmo não for dividida em muitas opções.	

de desenvolvedores como denominador a fim de normalizar o valor da métrica calculada. Para o exemplo anterior apresentado na Tabela 2, considerando que o maior tempo de contribuição da rede foi de 12 meses, os resultados são divididos por este denominador. Assim, a métrica GPC permite classificar a força potencial de contribuição em relação a todas as contribuições possíveis dentro da rede e por este motivo ela deve ser preferencialmente escolhida para o desenvolvimento das análises.

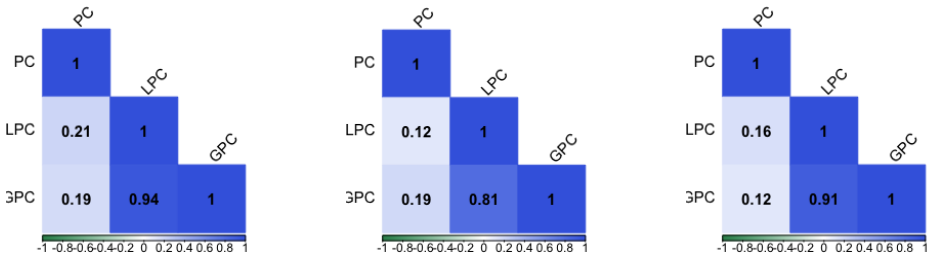
3. Resultados

Nesta seção, apresentamos a análise de correlação entre as propriedades com aspectos temporais e as demais existentes a fim de entender o relacionamento entre elas. Apresen-

Tabela 2: Exemplos de cálculo da LPC e GPC para dois pares de desenvolvedores.

Par de Desenvolvedores	Repositório	Duração Repositório	Tempo de Contribuição	LPC
Dev A - Dev B	R1	3 meses	3 meses	3/3 = 1,00
Dev C - Dev D	R2	12 meses	6 meses	6/12 = 0,50

Par de Desenvolvedores	Tempo de Contribuição	Maior Tempo de Contribuição da Rede	GPC
Dev A - Dev B	3 meses	12 meses	3/12 = 0,25
Dev C - Dev D	6 meses		6/12 = 0,50



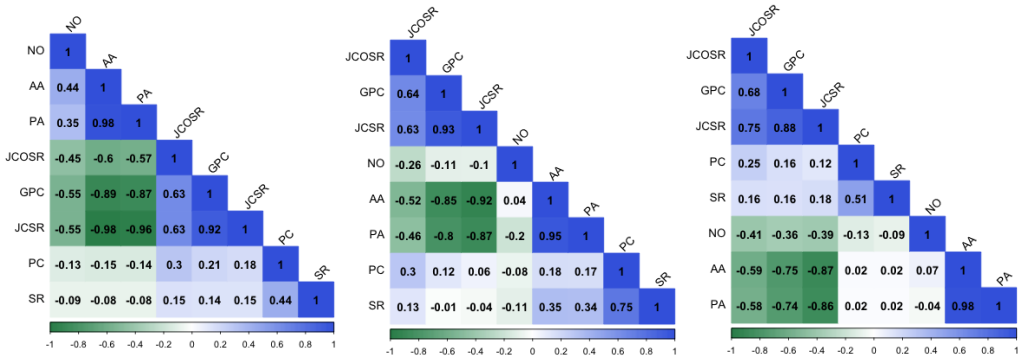
(a) JavaScript (b) Java (c) Ruby

Figura 2: Coeficiente de correlação de Spearman entre as novas propriedades com aspectos temporais: GPC, LPC e PC.

tamos resultados apenas para o coeficiente de correlação de Spearman, pois obtivemos resultados similares para o coeficiente de correlação de Pearson.

Análise das propriedades semânticas com aspecto temporal. As métricas GPC e LPC tendem a ser correlacionadas devido à sua forma de cálculo semelhante e às características das redes onde as mesmas foram aplicadas. A Figura 2 apresenta a correlação de Sperman para as três novas métricas. Conforme esperado, GPC e LPC são altamente correlacionadas monotonicamente. Em um modelo computacional para medir a força dos relacionamentos, a métrica escolhida seria a GPC por representar melhor a conexão do par de desenvolvedores no tempo em relação a todos os demais pares na rede. Além disso, a métrica PC não se relaciona com nenhuma das demais, independente do coeficiente de correlação. Este comportamento pode ser justificado por conta da janela temporal de colaboração. Para as métricas LPC e GPC, dois desenvolvedores só têm um valor de colaboração atribuído caso tenham trabalhado em um mesmo repositório na mesma janela temporal. Caso contrário, o valor tanto de LPC quanto de GPC para este par é igual a zero. No entanto, há uma aresta entre os dois desenvolvedores na rede, ainda que não tenham trabalhado na mesma janela temporal. Por isso, há um valor para PC calculado para tal par. Nesse cenário, a maioria dos pares de desenvolvedores tem valor zero para LPC e GPC e valor diferente de zero para PC, o que explica a não correlação de ambos. Assim, a não correlação entre tais métricas pode ser justificada pelos valores das arestas que não possuem interseção de tempo de contribuição, ou pode acontecer ainda que apenas os mesmos pares sejam considerados.

A fim de avaliar a correlação entre as métricas, realizamos a análise descartando as arestas em que não houve contribuição em uma mesma janela temporal. Após a remoção das arestas, a quantidade de pares de desenvolvedores é 16% da base de dados original para JavaScript, 27% para Java e 14% para Ruby. A análise das correlações para o coeficiente de Spearman e Pearson com os pares de desenvolvedores filtrados revela resultados



(a) JavaScript (b) Java (c) Ruby

Figura 3: Coeficiente de correlação de Spearman entre todas as métricas apresentadas

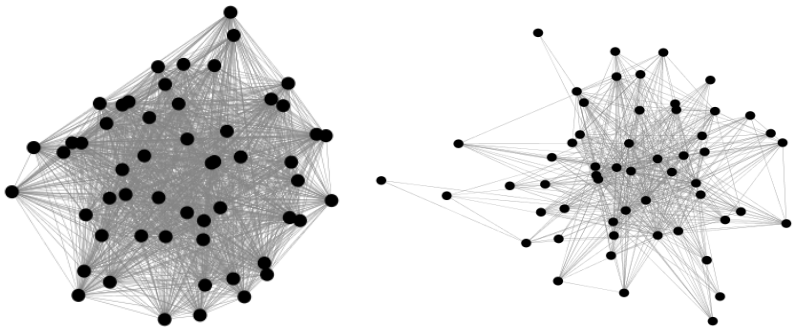
muito próximos aos da Figura 2. Assim, podemos considerar que a métrica PC gera uma informação de colaboração diferente das demais, fazendo com que seja importante calculá-la e considerá-la para medir a força de colaboração.

Correlação com propriedades existentes. Considerando as propriedades topológicas e semânticas existentes e as propriedades semânticas com aspectos temporais, a Figura 3 apresenta as correlações para o coeficiente de Spearman. Nas três linguagens de programação, a métrica GPC é altamente correlacionada à JCOSR. Desta forma, podemos afirmar que o tempo de contribuição entre desenvolvedores está bem relacionado com a quantidade de commits realizados pelo par de desenvolvedores em relação aos demais contribuidores em um mesmo repositório. Esta correlação é pertinente, uma vez que os commits de ambos os desenvolvedores são mais frequentes na janela temporal de trabalho entre eles. Além disso, a GPC também é bem correlacionada com a métrica JCSR, que avalia a contribuição de um par de desenvolvedores normalizada pela quantidade de desenvolvedores total nos repositórios. Em todas as correlações, as métricas SR e PC não estão fortemente correlacionadas a nenhuma outra. Em relação às métricas topológicas, AA e PA são fortemente correlacionadas em todos os cenários. Dessa forma, podemos considerar apenas uma entre elas em conjunto com a NO.

4. Aplicação

A baixa correlação entre LPC e GPC com as demais é devido à atribuição do valor zero a arestas entre desenvolvedores sem colaborações na mesma janela temporal de um repositório. Com isso, podemos utilizar o critério de janela temporal para a formação das arestas entre desenvolvedores na modelagem da rede. Para exemplificar a nova modelagem, selecionamos um repositório de JavaScript com um total de 55 colaboradores e 1.485 arestas, conforme Figura 4a. O repositório selecionado teve início em março de 2012 e término em setembro de 2015. Note que a janela temporal considera esse período de duração do repositório. Esta primeira rede apresenta densidade igual a 1 devido à formação de uma grande clique com conexões entre todos os desenvolvedores.

Aplicando os novos critérios temporais, a Figura 4b apresenta a rede do mesmo repositório, mas com as arestas geradas apenas entre desenvolvedores que colaboraram em uma mesma janela temporal. Desta forma, a quantidade de arestas na rede é reduzida para 571, ou seja, 38% da quantidade original. A densidade da nova rede é igual a 0,385,



(a) (b)

Figura 4: Grafo de um único repositório (a) com links entre todos os desenvolvedores e (b) com links entre desenvolvedores com contribuição numa mesma janela de tempo.

inferior à anterior com densidade igual a 1. Portanto, essa nova modelagem expressa um cenário mais realista do relacionamento entre desenvolvedores em um grande repositório. Neste caso, o grau médio dos nós presentes na rede fica em torno de 20,8.

5. Conclusão e Trabalhos Futuros

Analisamos como diferentes propriedades topológicas e semânticas se relacionam com as que consideram o aspecto temporal. Os resultados mostram que GPC e PC podem ser utilizadas em conjunto em um modelo computacional para medir a força dos relacionamentos. A principal contribuição é melhorar a modelagem do grafo para representar a rede social de colaboração. A modelagem sem o aspecto temporal conecta todos os desenvolvedores que contribuíram em um mesmo repositório. Tal abordagem cria várias cliques para cada repositórios. No entanto, as propriedades com aspecto temporal revelaram que cerca de 80% dos pares de desenvolvedores não contribuíam em uma mesma janela temporal. Assim, em trabalhos futuros planejamos modelar as redes atribuindo arestas apenas entre desenvolvedores que colaboraram num mesmo período de tempo.

Agradecimentos. Trabalho parcialmente financiado por CAPES, CNPq e FAPEMIG.

Referências

Alves et al., G. B. (2016). The strength of social coding collaboration on github. In *SBBD Short Papers*, pages 247–252, Salvador, Brasil.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Batista, N. A. et al. (2017). Collaboration strength metrics and analyses on github (aceito para publicação). In *IEEE/WIC/ACM WI*, Leipzig, Germany.

Casalnuovo, C. et al. (2015). Developer onboarding in github: The role of prior social links and language experience. In *ESEC/FSE*, pages 817–828, Bergamo, Italy.

Dabbish, L. et al. (2012). Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *CSCW*, pages 1277–1286, Seattle, USA.

Easley, D. and Kleinberg, J. (2010). *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press.

Gousios, G. (2013). The GHTorrent Dataset and Tool Suite. In *MSR*, pages 233–236, S.Francisco, USA.

Tsay, J. et al. (2014). Influence of Social and Technical Factors for Evaluating Contribution in GitHub. In *ICSE*, pages 356–366, Hyderabad, India.

paper:174074

Detecção de Anomalias no Transporte Rodoviário Urbano*

Ana Beatriz Cruz[1], João Ferreira[1], Bernardo Monteiro[1], Rafaelli Coutinho [1], Fabio Porto[2], Eduardo Ogasawara[1]

[1] CEFET/RJ

[2]LNCC - DEXL Lab

anacruz@acm.org, joao.parana@acm.org

rafaelli.coutinho@cefet-rj.br, fporto@lncc.br, eogasawara@ieee.org

Abstract. *The constant increase in road traffic jams demands research related to urban mobility. These studies model the traffic analysis as a trajectory problem,* i.e.*, from the individualized analysis of vehicles that continuously send their geolocations. Such moving objects function as trajectory sensors and produce large amounts of data. It is possible to observe, however, an absence of studies related to spatial-temporal aggregations from trajectories of the urban vehicles. As a preliminary work on this subject, this paper establishes a baseline for anomaly identification in urban mobility, which may be found useful for developing new approaches that provide a better comprehension of urban mobility systems.*

Resumo. *O constante aumento de congestionamentos no tráfego rodoviário demanda pesquisas relacionadas a mobilidade urbana. Esses estudos modelam a análise do tráfego como um problema de trajetória,* i.e.*, a partir da análise individualizada de veículos que continuamente transmitem suas geolocalizações. Tais objetos móveis funcionam como sensores de trajetória e produzem grande quantidade de dados. Observa-se, entretanto, que há uma lacuna de estudos associados a agregações espaço-temporais das trajetórias dos veículos urbanos. Como um trabalho preliminar no assunto, este artigo estabelece uma base de comparação para identificação de anomalias em mobilidade urbana, que pode ser útil para o desenvolvimento de novas abordagens que provenham uma maior compreensão dos sistemas de mobilidade urbana.*

1. Introdução

O aumento da população mundial em áreas urbanas associado ao grande número de veículos presentes nas cidades provocam problemas como congestionamentos, acidentes e poluição [Ferreira et al., 2013; Chen et al., 2015]. Por conta disso, observou-se um aumento das pesquisas relacionadas a mobilidade urbana. Diversos estudos vêm sendo feitos motivados por compreender o comportamento do trânsito e minimizar o impacto de engarrafamentos.

Tais estudos têm por foco o comportamento das trajetórias de objetos móveis. Ferreira et al. [2013], por exemplo, usaram dados emitidos por GPS de táxis para criar uma

*Os autores agradecem a CAPES, CNPq e FAPERJ pelo financiamento parcial do projeto.

ferramenta que favorece a análise visual da mobilidade da cidade de Nova York. Verhein and Chawla [2008] usaram dados emitidos por celulares, RFID e rastreamento por satélite para identificar trajetórias de trânsito intenso a partir da movimentação dos veículos. Pan et al. [2013] também estudaram as variações nas movimentações das trajetórias para observar comportamentos anômalos em algumas rotas.

Objetivando análises de comportamento mais sistêmicas, Andrienko and Andrienko [2008] propuseram métodos de agregação espaço-temporal, consolidando os dados de trajetória em mapas que representam principais fluxos no espaço urbano. Esses mapas de fluxo podem ser descritos por séries espaço-temporais das regiões (objetos permanentes) que ajudam a entender comportamentos da mobilidade urbana pela perspectiva das regiões [Adrienko and Adrienko, 2011].

Neste contexto, observa-se uma lacuna de estudos pela ótica das séries espaço-temporais de objetos permanentes que possam trazer uma melhor compreensão ou uma visão complementar do tráfego. Como um trabalho preliminar no assunto, este artigo estabelece uma metodologia para identificação de anomalias em dados agregados de mobilidade urbana. Tal abordagem estabelece uma base de comparação que pode ser útil para o desenvolvimento de trabalhos futuros que objetivam a sistematização e análise mais aprofundada dessas anomalias.

Além desta introdução, o trabalho está organizado em mais quatro seções. Na seção 2, são apresentados conceitos gerais necessários para o entendimento do problema e da metodologia aplicada. Em seguida, na seção 3, apresenta-se a metodologia adotada nesse trabalho. A seção 4 descreve uma avaliação preliminar da proposta. A seção 5 finaliza com as considerações finais.

2. Formalização

As séries espaço-temporais são definidas como sequências de observações de objetos que contêm dados sobre o local e momento em que as coletas foram realizadas [Cressie and Wikle, 2015]. Os objetos com localização fixa são classificados como permanentes (interpretados como sensores fixos), enquanto aqueles cujas localizações variam com o tempo são classificados como móveis (interpretados como trajetórias). A trajetória é o modelo de dados mais aplicado a problemas relacionados ao tráfego [Chen et al., 2015]. Entretanto, estudos que se relacionam mais diretamente com o tema deste trabalho são aqueles em que se agregam informações do objeto [Tao et al., 2004], gerando séries espaço-temporais associadas a objetos permanentes.

Uma trajetória de um objeto tr_k é descrita como sequência = $<\hat{tr}_1, \cdots, \hat{tr}_{co}>$ de pares $\hat{tr}_o = (p, v)$, onde $\hat{tr}_o.v$ corresponde a um valor ($\hat{tr}_o.v \in \mathbb{R}$) e $\hat{tr}_o.p$ a uma posição ($\hat{tr}_o.p \in \mathbb{R}^2$) no tempo o. O número de observações (co) em uma trajetória é dado por $|tr_k|$. Define-se também um *dataset* de trajetórias $TR = \cup_{k=1,ck} tr_k$, representado pelo conjunto de trajetórias tr_k em um sistema ao longo de um período de análise. O número de trajetórias (ck) em TR é dado por $|TR|$.

Durante o processo de agregação, é necessário estabelecer partições espaciais e temporais. Considere $S = \cup_{i=1,ci} \hat{s}_i$, regiões bi-dimensionais $|S| = ci, \hat{s}_i = [(xl_i, yl_i), (xu_i, yu_i)]$ que particionam o espaço S em ci regiões retangulares definidas pelos limites inferiores xl_i, yl_i e superiores xu_i, yu_i para os eixos x e y. Considere também

$T = \cup_{j=1,cj}\ \hat{t}_j, |T| = cj, \hat{t}_j = [(j-1)\cdot ma + 1, j \cdot ma)]$ um particionamento do tempo T em cj intervalos de duração ma.

Seja $ST = \cup_{i=1,ci}\ st_i, st_i = <\hat{u}_{i,1}, \cdots, \hat{u}_{i,cj}>$ um conjunto de séries espaço-temporais st_i provenientes das ci partições espaciais. Cada st_i contém uma série temporal de objeto permanente de tamanho cj provenientes das cj partições no tempo. Cada uma das observações $\hat{u}_{i,j}$ é produzida a partir da agregação espaço-temporal das trajetórias dos objetos. A Equação 1 descreve a agregação, aplicando-se a função de agregação θ sobre os valores $tr_k.\hat{tr}_o.v$ dos objetos que estavam presentes no particionamento espaço-temporal $\hat{s}_i, \hat{t}_j$.

$$\forall \hat{s}_i \in S, \hat{t}_j \in T\ (\hat{u}_{i,j} = \ _{\hat{s}_i,\hat{t}_j}\Gamma_{\theta(tr_k.\hat{tr}_o.v)}(\sigma_{o\in\hat{t}_j, tr_k.\hat{tr}_o.p\in\hat{s}_i} tr_k.\hat{tr}_o)) \quad (1)$$

Seja st_i uma série espaço-temporal de ST. Considere ma o intervalo de particionamento do tempo em minutos, tem-se que um dia contém $obs = 1440/ma$ observações. Por simplicidade, considere que ma é divisor de 1440. Para $w \in [1..obs]$, as observações de $\hat{u}_{i,w+d\cdot ma}$, para $d \in N$ representando dias, correspondentes às observações realizadas no mesmo horário nos diferentes dias. Fazendo-se variar w e tendo-se d um conjunto de dias, tem-se as $st_{i,w,d}$ as distribuições nos diferentes horários (w) caracterizados por cada séries st_i em d.

A partir destas distribuições, pode-se estabelecer os valores esperados $\overline{st_{i,w,d}}$ e intervalos típicos para as observações na região i, no horário w ao longo dos dias d. Assumindo-se distribuição não normal, o intervalo típico é descrito por $IT_{i,w,d} = [q_1 - 3 \cdot IQR, q_3 + 3 \cdot IQR]$ [Larsen and Marx, 2005], onde q_1 é o primeiro quartil, q_3 é o terceiro quartil e IQR é a distância interquartil da distribuição $st_{i,w,d}$. Os valores acima ou abaixo destes intervalos podem ser estatisticamente interpretados como anomalias.

3. Metodologia

O presente trabalho objetiva prover um método básico para identificar anomalias no sistema de transporte urbano a partir das séries espaço-temporais derivadas de agregações das geolocalizações emitidas minuto a minuto pelos ônibus do Rio de Janeiro. Os ônibus atuam, portanto, como sensores de trajetória. Os seus dados podem ser obtidos no Portal de Dados Abertos da Prefeitura do Rio de Janeiro [DataRio, 2016]. O processo para identificação dessas anomalias é dividido em cinco atividades principais: (i) seleção de objetos permanentes, (ii) agregação espaço-temporal, (iii) definição de intervalo típico, (iv) identificação das anomalias e (v) análise dos resultados.

A seleção dos objetos permanentes (regiões) é caracterizada pela divisão do município do Rio de Janeiro em um reticulado de 500 por 500 metros projetado sobre a área estudada. Devido as particularidades da geografia de algumas cidades, como o Rio de Janeiro, uma seleção foi feita a fim de descartar quadrantes localizados em regiões em que não há tráfego de ônibus. A etapa de agregação espaço-temporal foi realizada considerando este reticulado e o particionamento temporal de dez minutos. A agregação espaço-temporal considera a velocidade média, quantidade e lista de ônibus e de linhas presentes na região para cada intervalo de dez minutos. Ao fim dessa etapa, são obtidas séries espaço-temporais para cada região. Posteriormente, é feito o cálculo do intervalo típico de cada região que serve de base de comparação para as observações realizadas. Os

valores de velocidade identificados fora desse intervalo indicam anomalias nos padrões do trânsito.

Este processo foi implementado como um workflow. Dado o cenário de larga escala de dados e a necessidade de processamento de alto desempenho, o workflow foi desenvolvido em Apache *Spark*. O encadeamento do workflow foi implementado em Scala. As atividades foram escritas em Python e R [Ferreira et al., 2017] e invocadas a partir da especificação em Scala.

4. Avaliação Experimental

A Metodologia descrita na seção 3 foi aplicada sobre o período de dois meses: julho e agosto de 2014. O mês de julho teve em média 2.961.726 observações diárias, enquanto que a média de observações diárias de agosto é de 3.403.180. Inicialmente, para cada mês, foram calculados os intervalos típicos. A metodologia de identificação de anomalias aplicada sobre o mês de julho resultou na identificação de 42.050 anomalias. Esse volume representa aproximadamente 1,03% das observações. Em agosto, foram identificadas 38.843 anomalias, volume que representa aproximadamente 1,07% da base comparada. Também aplicamos os intervalos típicos computados em julho para servir de base de comparação relativa com as observações do mês de agosto. Desta forma, foram identificadas 64.041 anomalias, correspondente a 1,76% do volume da base comparada.

A Figura 1 ilustra a quantidade de anomalias por área de planejamento da cidade nos períodos estudados. Embora as análises sobre meses de julho e agosto individualmente indiquem a maior concentração de anomalias na área de planejamento da Zona Norte do Rio de Janeiro, a comparação relativa entre agosto e julho mostrou que o maior número de anomalias ocorreu na área de planejamento da Zona Oeste, seguida pela Zona Norte. Na Zona Norte, o maior número de anomalias aponta para um aumento da velocidade média, enquanto que na Zona Oeste, a maioria das anomalias identificadas indicam o aumento de engarrafamentos. As anomalias que indicam redução da velocidade na Zona Oeste, concentram-se principalmente nos bairros de Guaratiba, Santa Cruz e Campo Grande, que, juntos, representam aproximadamente 71,93% das anomalias de lentidão na Zona Oeste e 23,8% das anomalias de lentidão de todo o Rio de Janeiro.

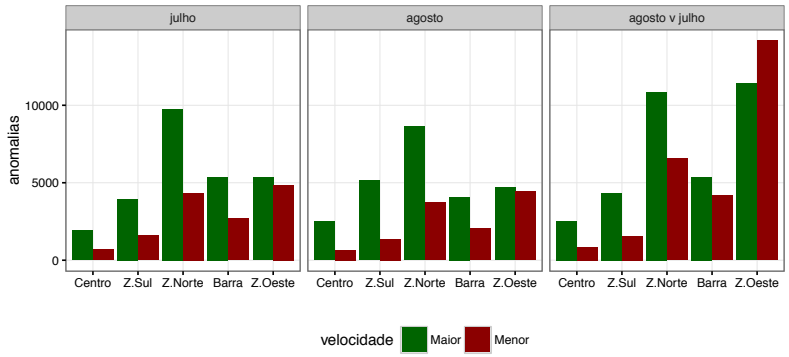


Figura 1. Anomalias identificadas por área de planejamento

As anomalias abertas por dia da semana podem ser vistas na Figura 2. Um resultado esperado é que ocorram aumentos significativos de velocidades aos domingos. No mês de julho, em especial, anomalias que indicam o aumento da velocidade média são

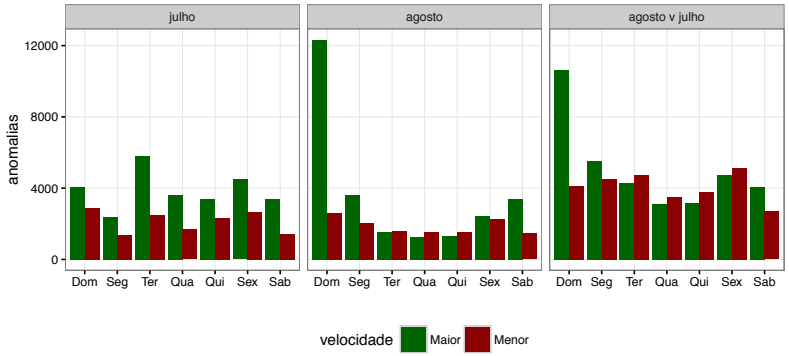


Figura 2. Anomalias identificadas por dia da semana

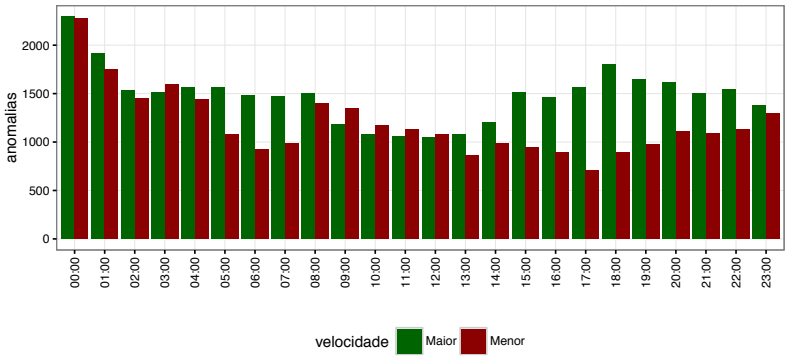


Figura 3. Anomalias identificadas por faixa de horário (ago v julho)

maiores nas terças. Esse comportamento específico do mês de julho ocorreu por causa da Copa do Mundo, que teve alguns de seus jogos no Rio de Janeiro. Nesse mês, os dias 08/07/2014 e 04/07/2014 foram os que apresentaram mais anomalias de maior velocidade. O dia 08/07/2014 (terça feira) foi o dia do jogo entre Brasil e Alemanha nas semifinais da Copa do Mundo. O dia 04/07/2014 (sexta feira) foi feriado devido à Copa do Mundo. Foi identificado ainda um volume de anomalias associados à diminuição de velocidade no domingo superior ao observado no mês de agosto. Esse comportamento anômalo foi decorrente dos protestos que ocorreram na cidade no dia 06 de julho, da 14ª parada gay que ocorreu no dia 20 de julho e da demolição do último segmento do Elevado da Perimetral, interdição da Avenida Rodrigues Alves e alteração no percurso de diversas linhas de ônibus no dia 27 de julho.

As anomalias abertas por faixa de horário, comparando-se agosto versus julho, podem ser vistas na Figura 3. A partir desta comparação, foram identificadas anomalias que indicam o aumento da velocidade média em horários considerados de pico como o intervalo entre cinco horas da tarde e oito horas da noite. No período entre seis e oito horas da manhã, apesar do número de anomalias que indicam velocidades maiores ser superior ao número de anomalias que indicam diminuição da velocidade média, pode-se observar o aumento gradual do número de anomalias que indicam velocidades mais lentas. Esse aumento gradual de anomalias culmina na maior proporção de anomalias de diminuição de velocidade a partir das nove horas da manhã, normalizando após meio dia. Esse comportamento pode ser visto no gráfico ilustrado na Figura 3 e reflete o comportamento de formação de engarrafamentos no horário de pico da manhã.

5. Considerações finais

O presente trabalho apresenta um método básico para identificação de anomalias no comportamento do trânsito rodoviário analisados por meio de agregações espaço-temporais. Tomou-se como base os dados abertos do município do Rio de Janeiro. A metodologia proposta foi implementada em Apache Spark e identificou algo equivalente a 1% de anomalias por mês estudado. Tal abordagem estabelece uma base de comparação que pode ser útil para o desenvolvimento de trabalhos que objetivam a sistematização e análise mais aprofundada dessas anomalias. Por exemplo, observou-se que há diferentes tipos de anomalias. Alguns têm comportamentos pontuais oriundos de fenômenos singulares (como o dia do jogo do Brasil versus Alemanha na semi-final da Copa do Mundo de 2014), enquanto outros são recorrentes e descrevem comportamentos mais sistêmicos. Neste caso, abre-se espaço para explorar técnicas de padrões frequentes capazes de aglutinar as anomalias recorrentes como também aquelas que são perenes em intervalos de tempo e espaço contíguos.

Referências

Adrienko, N. and Adrienko, G. (2011). Spatial generalization and aggregation of massive movement data. *IEEE Transactions on visualization and computer graphics*, 17(2):205–219.

Andrienko, G. and Andrienko, N. (2008). Spatio-temporal aggregation for visual analysis of movements. In *Visual Analytics Science and Technology, 2008. VAST'08. IEEE Symposium on*, pages 51–58. IEEE.

Chen, W., Guo, F., and Wang, F.-Y. (2015). A survey of traffic data visualization. *Intelligent Transportation Systems, IEEE Transactions on*, 16(6):2970–2984.

Cressie, N. and Wikle, C. K. (2015). *Statistics for spatio-temporal data*. John Wiley & Sons.

DataRio (2016). Portal de dados abertos da prefeitura do Rio de Janeiro. Technical report, `http://data.rio/`.

Ferreira, J., Gaspar, D., Monteiro, B., Silva, A. B., Porto, F., and Ogasawara, E. (2017). Uma Proposta de Implementação de Álgebra de Workflows em Apache Spark no Apoio a Processos de Análise de Dados. In *Brazilian e-Science Workshop*.

Ferreira, N., Poco, J., Vo, H. T., Freire, J., and Silva, C. T. (2013). Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2149–2158.

Larsen, R. J. and Marx, M. L. (2005). *An Introduction to Mathematical Statistics and Its Applications*. Prentice Hall, Upper Saddle River, N.J, 4 edition edition.

Pan, B., Zheng, Y., Wilkie, D., and Shahabi, C. (2013). Crowd Sensing of Traffic Anomalies Based on Human Mobility and Social Media. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 344–353, New York, NY, USA. ACM.

Tao, Y., Kollios, G., Considine, J., Li, F., and Papadias, D. (2004). Spatio-temporal aggregation using sketches. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 214–225.

Verhein, F. and Chawla, S. (2008). Mining spatio-temporal patterns in object mobility databases. *Data mining and knowledge discovery*, 16(1):5–38.

paper:174636

Don't Keep My UDFs Hostage - Exporting UDFs For Debugging Purposes

Pedro Holanda[1], **Mark Raasveldt**[1], **Martin Kersten**[1]

[1] Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands

{holanda,m.raasveldt,mk}cwi.nl

Abstract. *User-defined functions (UDFs) are an integral part of performing in-database analytics. Executing data analysis inside a database provides significant improvements over traditional methods, such as close-to-the-data execution, low conversion overhead and automatic parallelization. However, UDFs have poor support for debugging. Since they are executed from within the database process, traditional debugging tools such as Integrated Development Environments (IDEs) and Read-Eval-Print Loops (REPLs) cannot be used during development. As a result, writing functional UDFs is challenging. In this paper, we present an extension to the open-source database system MonetDB that allows developers to debug their UDFs using modern debugging techniques.*

1. Introduction

Data scientists rely on scripting languages, such as R and Python, to perform data analysis tasks. Combining these languages with a database has many advantages. The traditional method of using a database in conjunction with these scripting languages is to connect to a database using a client protocol. The data is then transferred from the database to the analytical tool. However, this is very inefficient when dealing with large amounts of data [Raasveldt and Mühleisen 2017].

In-database analytics promises to solve this issue. By performing the analytics inside the database, the data transfer overhead is mitigated [Raasveldt and Mühleisen 2016]. The primary way of performing in-database analytics is through the use of UDFs. As shown in Table 1, most database vendors support UDFs in at least one scripting language frequently used for analysis.

While UDFs can perform highly efficient data analysis, debugging them has been a long-standing problem. Because the UDFs are executed within the database server, developers cannot use sophisticated debugging techniques (e.g., Interactive Debugging). Instead, they have to resort to inefficient debugging strategies in an effort to make their code work.

With the increasing number of R and Python packages, these analytical programs are quickly growing in complexity and scope. Finding and correcting bugs in complex programs is a challenging and time consuming task. In some cases this can take up to 50% of development time [Hailpern and Santhanam 2002]. However, when using more primitive debugging techniques, such as print or log-file debugging, this can take even longer.

Table 1. DBMSs Support for UDFs

DBMS	Python	R
Greenplum	X	X
DB2	-	-
MonetDB	X	X
MySQL	-	-
Oracle	-	X
PostgreSQL	X	X
Redshift	X	-
SQL Server	X	X
SQLite	X	-
Vertica	-	X

In this paper, we propose a solution to this problem. From within a client connected to a database, we ask the user to provide us with a problematic SQL statement that they want to debug. We then analyze it, and extract any required input data from the database server. We transfer the input data and the source code of the UDF to the client. Finally, we execute the function in the client process exactly as it would have been executed in the database server. The developer can then easily use interactive debugging tools to aid him in creating and modifying the UDFs.

The main contributions of this paper are:

- We introduce a novel strategy that allows developers to use interactive debugging techniques in conjunction with UDFs.
- We provide an Open-Source implementation of our work for the popular database MonetDB.
- We show our debugging framework being used for a concrete use-case, and compare it against the old ways of debugging UDFs.

Outline. This paper is organized as follows. Section 2 discusses the main debugging techniques and the state-of-the-art debugging for scripting UDFs. In Section 3 we describe our solution and demonstrate a concrete use-case that shows the difficulties of Python UDF debugging. In Section 4 we present our conclusions and discuss future work.

2. Background & Related Work

Debugging is the process of localizing, classifying, understanding and repairing an error in a piece of software. Many debugging techniques have been developed and are frequently used in the software development process [McConnell 2004]. The most commonly used debugging techniques are:

- **Print Debugging.** The developer adds print statements to the code in order to track the values of different variables while the code is running. This technique has several disadvantages. The developer needs to decide upfront which variables he wants to track and where he wants to track them. If the developer later decides that different variables should be tracked, he needs to recompile and rerun the code. Another disadvantage is that temporary code has to be inserted into the software, that then need to be removed after fixing the bug. Print debugging also

works poorly in multithreaded programs, because print statements from different threads get mixed in the output stream.

- **Log-File Debugging.** Log-File debugging is similar to print debugging, however, instead of writing to a standard output stream the developer writes output to a log-file. This has an advantage over print-debugging, in that log files are kept around even after the console has been closed. However, it still suffers from many of the same problems as print-debugging, as it requires the source code to be altered by the programmer upfront.
- **Interactive Debugging.** Interactive debugging allows the developer to inspect the state of a program while it is running. It allows the developer to work interactively with the source code, printing any variables he thinks are relevant to the program, and even stepping through the code line-by-line. Certain interactive debuggers even allow the developer to change the source code while the program is running. This form of debugging can be used without any modification to the source code. The developer only needs to attach a debugger to a running program to start his debugging session.

Scripting languages have several popular interactive debugging tools. The two main types of tools are console debugging (i.e. REPLs) and IDEs. Python has many different options for both REPLs and IDEs, such as pdb[1], IPython[Pérez and Granger 2007], PyDev[2] and PyCharm[3]. However, these cannot be used together with UDFs because these debugging tools need to control the code execution. This is not possible for code that runs within a database process, since the database controls the code flow. Consequently, developers of UDFs are forced to use either print or log-file debugging.

3. Debugging UDFs

The main issue with using interactive debuggers in conjunction with UDFs is that the execution of the UDFs happens inside the database server. To control the execution of the UDF, the debugger needs to be in control of the code. Hence, client-local execution of UDFs is necessary.

To accomplish client-local execution we transfer the UDF from the database server to the clients' machine and execute it in the local environment. A challenge here is that errors can be data-dependent, and the input data of the UDFs depend on the query that the UDF is used in. Consequently developers of UDFs often want to debug their UDFs in the context of a specific SQL query. To facilitate this, we allow developers to input an arbitrary SQL query that involves a call to the UDF they want to inspect. Both the UDF and the required input data are then transferred from the database and sent to the client. Afterwards they are either exported to a file for use in external IDEs, or directly executed in an interactive debugger.

The problem with transferring the input data of UDFs from the database to the client is that the amount of data can be very large. This is only required for debugging, however. The actual execution of the UDFs still occurs completely inside the database server. Still, it is possible that the clients' machine does not have space for all the required

[1] `www.docs.python.org/2/library/pdb.html`
[2] `www.pydev.org`
[3] `www.jetbrains.com/pycharm`

input data. For this reason, we also introduce an optional sampling step that only transfers a uniform random sample of the data instead of the full input data set.

Listing 1 depicts an example of a UDF that is supposed to compute the mean absolute deviation of a given column using the following formula:

$$\text{mean_deviation}(X) = \frac{\sum |X - X'|}{N}$$

where X represents a column, X' the columns' mean value and N the columns' size. However, the example has a bug. On line 9, the regular difference is calculated instead of the absolute difference, producing a semantic error (i.e., syntactically correct but logically incorrect). This bug can be fixed by computing the absolute difference instead. However, to locate the bug, the developer must inspect the values of variables throughout the programs' execution.

Listing 1. Mean Deviation UDF

```
1  CREATE FUNCTION mean_deviation(column INTEGER)
2  RETURNS DOUBLE LANGUAGE PYTHON {
3      mean = 0
4      for i in range (0, len(column)):
5          mean += column[i]
6      mean = mean / len(column)
7      distance = 0
8      for i in range (0, len(column)):
9          distance += column[i] - mean
10     deviation = distance/len(column)
11     return deviation;
12 };
```

To use print debugging to locate the error, the developer must alter the original source code. The source code must be modified to include print statements that track variable changes. Using Listing 1 as an example, print statements could be introduced after line 6 and 10 to track the mean and distance variables respectively. This solution requires many iterations, placing the print statements in different positions and tracking different variables. Each iteration includes a modification to the source code and requires rerunning of the function, which can be very time consuming. Finally, after finding and fixing the bug, the print statements must be removed.

We can see that print debugging is very cumbersome, and requires rerunning the query multiple times before the bug is found, even in this simple example. Listing 2 depicts how the developer can debug this function using our solution. First, the developer connects to a running database using the Python client (pymonetdb[4] for MonetDB). Then, the developer creates a SQL query involving the UDF, just as he would if he were to normally execute the query. However, instead of executing it, he calls either the `debug` function (line 5) for console debugging or the `export` function (line 6) for IDE debugging.

[4] `www.pymonetdb.readthedocs.io/`

Listing 2. Debugging UDFs using pymonetdb

```
1 import pymonetdb
2 conn = pymonetdb.connect(database='demo')
3 c = conn.cursor()
4 sql = 'select mean_deviation(id) from tables;'
5 c.debug(sql, 'mean_deviation')
6 c.export(sql, 'mean_deviation')
```

The `debug` function directly executes the UDF inside the client environment and attaches an interactive console debugger (pdb) to the UDF. The developer can then use the debugger to step through the code, and obtain the values of variables throughout the codes' execution. In addition, the developer can set breakpoints or watchpoints to automatically monitor changes to variables.

The `export` function writes the UDF and the input data to a file, which can then be inspected by the developer and imported into an IDE of his choice. The IDE can then be used to interactively debug the function and step through the code.

Both functions take an optional sample parameter, that can be used to control the maximum allowed input size. This way, transfer of a large amount of data to the client can be avoided.

Loopback Queries

MonetDB/Python supports loopback queries inside UDFs. Loopback queries allow users to query the database directly from within the UDF. The results of the query are converted to the host language of the UDFs. In Python UDFs, they can be can used through the `_conn` object that is passed to every UDF. Loopback queries are useful because they can bypass cardinality restrictions of the relational querying model. Listing 3 depicts an example of a UDF that uses a loopback query to retrieve a classifier from the database, and subsequently uses the classifier on its input data.

Listing 3. Loopback Queries

```
1 CREATE FUNCTION classify(id INTEGER, value INTEGER)
2 RETURNS TABLE(id INTEGER, prediction STRING)
3 LANGUAGE PYTHON
4 {
5     import pickle
6     res = _conn.execute("SELECT * FROM classifier WHERE name='RFC';")
7     classifier = pickle.loads(res['classifier'][0])
8     return {'id': id, 'prediction': classifier.predict(value) }
9 };
```

Loopback queries are a challenge for client-local execution. Because the UDF no longer runs inside the database, it cannot be queried in the same fashion, hence they have to be modified to work in this scenario. One solution would be to run all the loopback queries when exporting the function and retrieving their results in addition to the input data of the UDF, however, this will not work in all scenarios because the queries themselves are not required to be constant and might depend on external data. Instead, the actual loopback queries must be issued to the database when the UDF is running. This can be done by maintaining an open client connection to the database during client-local execution.

Extending to Other Databases

Our solution is implemented for MonetDB/Python. The same implementation strategy can be used for other RDBMSes by extending their respective clients. However, the processing model of the respective database needs to be taken into consideration. MonetDB uses the operator-at-a-time processing model, which means the UDFs are only called once with the entire columns as input. Row-store databases (e.g. Postgres or MySQL) use the tuple-at-a-time processing model, under which the UDFs are called many times with only individual rows as input. As this changes the way UDFs are called, the debugging framework must be adapted to these differing processing models.

4. Conclusions & Future Work

In this paper, we identify the problem of debugging scripting language UDFs in databases and propose two different solutions to allow developers to quickly and easily debug UDFs using modern techniques. The developer provides a problematic SQL query in which the UDF is executed, after which we export the UDF and any required data from the database so it can run locally in an interactive debugger on the developers' machine.

Future Work.

In our current solution, the UDFs and required data are exported so they can be debugged outside the database. However, this introduces both security and performance issues. A way to solve this issue is to run the debugger directly in the database. Another possibility of future work would be the development of an IDE plugin that easily allows the user to export, import and modify his UDFs.

Acknowledgments

This work was funded by the Netherlands Organisation for Scientific Research (NWO), projects "Data mining on high volume simulation output" (Holanda) and "Process mining for multi-objective online control" (Raasveldt).

References

Hailpern, B. and Santhanam, P. (2002). Software debugging, testing, and verification. *IBM Systems Journal*, 41(1):4–12.

McConnell, S. (2004). *Code complete*. Pearson Education.

Pérez, F. and Granger, B. E. (2007). IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29.

Raasveldt, M. and Mühleisen, H. (2016). Vectorized udfs in column-stores. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management, SSDBM 2016, Budapest, Hungary, July 18-20, 2016*, pages 16:1–16:12.

Raasveldt, M. and Mühleisen, H. (2017). Dont hold my data hostage-a case for client protocol redesign. *Proceedings of the VLDB Endowment*, 10(10):1022–1033.

paper:174232

Empirical Evaluation of Strategies to Process Range Queries of Numeric Sequences in Batch-Mode

Luiz F. A. Brito[1], **Marcelo K. Albertini**[1]

[1]Faculty of Computing – Federal University of Uberlândia (UFU)
Av. João Naves de Ávila, 2121 – Santa Mônica – 1A236 – Uberlândia – MG – Brazil

{luiz.brito,albertini}@ufu.br

Abstract. *Tree structures are largely used to index and search sequences on secondary memory. In some situations, many range queries are processed almost simultaneously and the resulting number of disk accesses can be high. In order to reduce the number of disk accesses, similar sequences can be grouped and spanned as a single query. A simple strategy is to unify all sequences into a single group. However, other strategies for grouping sequences can also be used. In this paper, we present and empirical evaluation of* 5 *common grouping strategies for* R*-trees and* M*-trees. Our results indicate that for inputs modelled as a random walk distribution the overall best implemented strategy for grouping queries is indeed the one unifying all queries in a single group.*

1. Introduction

Fast retrieval of similar sequences are essential in many applications. In query-by-humming systems for example, a hum is sent as a query to retrieve songs with similar passages [Kotsifakos et al. 2015]. In protein similarity search, large indexes containing DNA sequences allow querying for known genomes [Camoglu et al. 2003].

The problem of range query for sequences is defined as follows: given a query sequence, find sequences in database that are similar within a distance ϵ [Faloutsos et al. 1994]. The naive strategy would retrieve all sequences from secondary memory and evaluates them sequentially. However, it is costly since every sequence in database may be considered.

Index structures such as R-trees [Guttman 1984] and M-trees [Ciaccia et al. 1997] are commonly used to answer range queries efficiently. These indexes organize similar sequences in order to discard tree branches not containing candidates during a range query.

In problems where multiple range queries can be processed in batch, we can use query grouping strategies to improve performance. These strategies group similar query sequences so that we can use a smaller number of range queries to search the index structure. This reduces the number of disk accesses because similar queries would traverse almost identical paths in the tree.

A simple strategy is to group all queries together and to search through the index only once [Moon et al. 2001]. This can improve the search when queries are similar. However, when many query sequences are dissimilar the resulting range query can cover most of the index. In the worst case, the search is degraded to scanning all sequences. Therefore, the choice of query grouping strategies impacts directly on the performance of batch-mode range queries.

The problem of grouping sequences is mentioned in literature. For example, [Faloutsos et al. 1994] compares grouping techniques for finding R-tree hyperrectangles during insertion to reduce the size of indexes. In [Fu et al. 2008] the authors perform multiple queries using a single hyperrectangle motived by its simplicity. Although the impact of grouping strategies in performance of range queries may be high, this subject is usually little discussed.

In this paper, we empirically evaluate 5 strategies to group sequences. They are described in Section 3. Some of these strategies, originally described for R-trees [Faloutsos et al. 1994, Moon et al. 2001, Fu et al. 2008], were adapted to the M-tree structure.

In our experiments, we model sequences by a random walking distribution. We use this model because often sequences can be transformed into random walks by using difference operations. We have performed two types of tests: using $\epsilon = 0$ to check if a sequence is in the database and using small values of $\epsilon > 0$ to retrieve only a few similar sequences. Our results show for small $\epsilon > 0$, R and M-trees number of disk accesses is significantly reduced by batch-mode strategies.

In the next section, we describe R and M-tree operations to index sequences. Next, we detail the query grouping strategies implemented for our tests and how they are applied to R and M-trees. In Section 4, we describe our methodology and discuss our results. Finally we draw our conclusions regarding the usage of query grouping strategies.

2. Background

R and M-trees are data structures optimized to secondary-memory storage. Often, they are employed to index sequences for fast retrieval through queries such as range queries. In R-trees, sequences are treated as d-dimensional points where similar points are grouped in the same node limited by a d-dimensional hyperrectangle, also known as Minimum Bounding Rectangle (MBR). M-trees can index any kind of object as long as its representation is comparable by a metric function. Differently from R-tree, M-tree nodes are delimited by hyperspheres.

There are two types of nodes in these structures: directory nodes, containing objects of directory type, and leaf nodes, containing objects of leaf type. Each directory object contains an id and a `container` representation, which is a hyperrectangle in a R-tree and is a sequence in a M-tree, along with a radius distance. Each directory also has a reference to a child node and, for M-tree nodes, a precomputed distance to the parent object in order to speed up calculations by using triangle inequality [Orchard 1991]. Each leaf object contains an id; a sequence, which works as a key added by an insertion, and a value to be associated with the key.

In an insertion, given a new sequence to be stored, the procedure for both trees can be described as follows: at the root level, if the node is not a leaf then choose the directory object that has the `container` that best fits or is closer to the new sequence. If the sequence does not fit entirely inside the `container`, then enlarge it and retrieve the child node referenced by the directory object. This step is applied recursively for child nodes until a leaf node is found. Finally, it adds a new leaf object containing the new sequence.

During the insertion, if a node is already full then a split policy is applied in order to redistribute its objects between itself and a new node. When a leaf node is full, the insertion algorithm firstly creates a new node and calls the split policy to redistribute the leaf objects. Then it creates two new `containers` in order to fit the two nodes and inserts them in the parent's object list, replacing the old `container`. In case the full node is a directory, its objects are also redistributed and the new `containers` are created from other `containers`. For M-trees the algorithm also needs to update parent distance of modified objects. This splitting process is applied recursively up to the root node. If the root is also full, then a new root node is created and the tree grows one level.

In a range query, given a sequence and the maximum distance allowed ϵ, first the algorithm creates a query `container`. This `container` has the center equals to the query sequence and each dimension width equals to 2ϵ. The search operation begins at root level and descends every branch that has possibility to be inside the container. This process repeats recursively until all possible leaves are reached. Finally, the algorithm retrieves every sequence reached within a distance ϵ to the query sequence.

3. Strategies for Processing Range Queries in Batch-Mode

In the literature, the predominant approach, named `Single Grouping`, is to group all queries together and search through the index just once [Moon et al. 2001, Fu et al. 2008].

In order to verify the importance of grouping queries to process range queries in batch-mode, in this paper we evaluate the following strategies: `k-Medoids Grouping`, `Single Grouping`, `n-Random Grouping`, `Maximum Capacity Grouping`, and `Adaptive Grouping` [Fu et al. 2008, Park and Jun 2009, Faloutsos et al. 1994].

`Single Grouping` (SG) strategy puts every query in a single group and performs just one range query using a single query `container`. For R-tree, this strategy is the same as the `n-Random Grouping` strategy with $n = 1$ since any query can be chosen at the beginning of the grouping process and it would still result in the same `container`. However, for M-trees is important to choose the query that is centralized. In this case, our algorithm chooses the query that has the lowest distance sum to all other queries.

`n-Random Grouping` (NRG) strategy selects n queries randomly and assigns them to different groups. They are enclosed by an initial `container`. It assigns each remaining query to the best-fitting container. If a query does not fit entirely in the `container`, then enlarge it. Finally, for each group enlarge its `container` by ϵ and process a range query.

In the `Maximum Capacity Grouping` (MCG) strategy, given the maximum number of sequences allowed per group m, the algorithm first creates a new group enclosing an initial `container`. Then it assigns the next m queries to the current group, enlarging the `container` when necessary. If the current group becomes full, then the algorithm creates a new group. This process continues until every query have been assigned to a group.

`k-Medoids` (KMG) strategy finds the best representative queries to form k clusters, namely medoids, such that the sum of distances among every object and its medoid is minimized. In this paper we used the algorithm presented in [Park and Jun 2009] because

it outperforms the well known clustering algorithm Partition Around Medoids (PAM) despite its simplicity.

The algorithm to find the k medoids first computes the distance matrix of all objects. Then, it selects k initial medoids based on lowest sum of normalized distances computed as $v_j = \sum_{i=0}^{n-1} (d_{ij} / \sum_{l=0}^{n-1} d_{il})$, where j is the target object, i and l can be any object and d_{xy} is the distance between objects x and y. In next step, this algorithm assigns the remaining objects to clusters that have the closest medoids and it calculates the cost of the current configuration. For the following steps, similar to partitioning clustering algorithms, the algorithm iteratively updates medoids such that the total distance inside cluster is minimized and reassigns the remaining objects to clusters. This algorithm finishes when objects stop changing groups.

Our `Adaptive Grouping` (AG) strategy is adapted from [Faloutsos et al. 1994], originally introduced to group d-dimensional points in MBRs during insertion. They proposed to group sequences at insertion time but not at querying time. We have adapted this strategy to the context of querying. AG calculates the marginal cost for assigning a query in the current group. The marginal cost mc for R-tree is calculated by: $mc = \prod_{i=1}^{d}(L_i + 0.5)/k$, where L_i is ith dimension width of the resulting MBR and k is the number of queries in resulting group. The strategy described in [Faloutsos et al. 1994] says if the marginal cost for a given query is lower than the previous cost then it assigns the query to the current group and enlarges the `container` as necessary. Instead, in our implementation we also look for other groups to check for lower maginal costs. It assigns the current query to a new group if its cost is larger than the current cost for all groups.

For M-trees, we propose to compute the marginal cost as: $(r + 0.5)^d/k$, where r is the radius of resulting hypersphere. The marginal cost for M-tree is justifiable because its value increases proportionally to the volume of the query hypersphere. We note the data needs to be normalized in range $[0, 1)^d$ in order to compute these marginal costs.

4. Experiments

In order to compare the query grouping strategies previously discussed we have built indexes using R and M-tree structures for dimensions ranging from 2^2 to 2^{10}. For each dimension we generated a dataset containing 10^5 sequences using the random walk distribution. Sequence values were normalized to the interval $[0, 1]$ using the formula $S_i^{norm} = (S_i - min(S))/(max(S) - min(S))$, where S_i is a value in sequence S, treated by the indexes as a dimension, and, $min(S)$ and $max(S)$ are the lowest and highest values of S, respectively.

In our experiments, for each dimension we used the same indexes to test different grouping strategies in order to prevent potentially effects related to the index construction and data randomness. We organize our experiments as following: for each index we processed batch-mode range queries containing 25, 100 or 1000 sequences. For `n-Random Grouping` and `k-Medoids Grouping` strategies we chose the number of groups to be the base 2 logarithm and the square root of the batch size. For the `Max Grouping` we chose the maximum allowed number of sequences such that the number of resulting groups was almost the same as NRG and KMG strategies.

The experiments were divided in two categories. In the first category, we processed

batch-mode range queries with $\epsilon = 0$ in order to check the existence of sequences in indexes. In the second category, we computed ϵ based on the dataset average energy. The mean energy E of a dataset is computed by $E(D) = (\sum_{i=0}^{n-1} \sum_{j=0}^{d-1} |D_{ij}|^2)/n$ where D_i is a sequence of size d and n is the number of sequences in the dataset. The final formula was $\epsilon = \frac{1}{4}\sqrt{E(D)}$. This value for ϵ allows retrieval of roughly the same number of results given a batch-mode range query.

In order to assess the grouping strategies we measured the ratio of the number of disk accesses per query when batch-mode was not used divided by the total disk accesses used per query by a given strategy, and a similar formulation of ratios for distance calculations. Ratio values higher than 1 indicate improvement of performance when using batch-mode strategies. The naive strategy, which process a range query for each sequence independently, was the baseline for our experiments and we refer as `No Grouping` (NG) strategy. Figure 1 shows the results regarding disk accesses. Each experiment considers: data structure, grouping strategy, ϵ value, amount of sequences per batch, and sequence dimensionality.

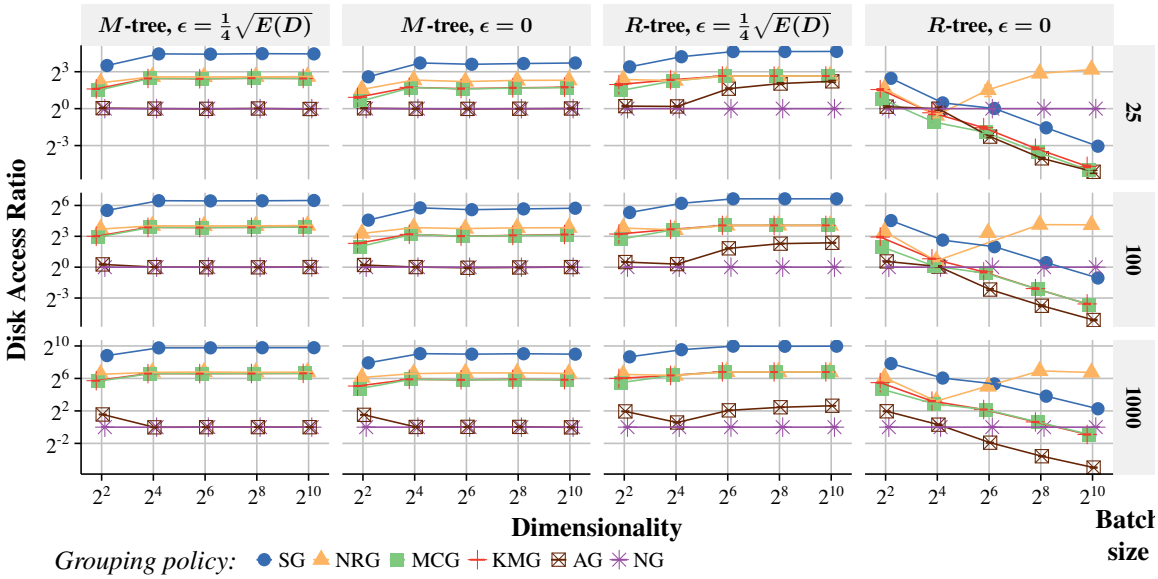


Figure 1. Disk accesses ratios for query grouping strategies using R-trees and M-trees.

In Figure 1, we noted the `Single Grouping` strategy was the best overall strategy. This strategy used less disk accesses in order to answer range queries. The `n-Random Grouping`, `Max Capacity Grouping` and `k-Medoids Grouping` strategies were similar and obtained better results than the `Adaptive Grouping` strategy. However, for R-tree and $\epsilon = 0$, the NRG strategy showed improvement in higher dimensionality. In our tests we noted the adaptive strategy usually found too many groups thus this approach was similar to performing non-batched range queries.

Results also shown large batches of sequences have positive impact on the number of disk accesses. For example, the combination of R-tree, batch of size 1000 and `Single Grouping` strategy used approximately 2^{10} times less accesses to disk for range queries with $\epsilon = \frac{1}{4}\sqrt{E(D)}$. The same behavior happened when using M-tree indexes.

We also analysed the ratios of distance calculations [1]. These results have shown when $\epsilon = 0$ the distance calculations increase considerably and correlates with dimensionality. At the other hand, the number of distance calculations reduces when $\epsilon = \frac{1}{4}\sqrt{E(D)}$ using R-tree.

5. Conclusions

We evaluated 5 query grouping strategies in order to verify whether its use could be employed to improve performance of batch-mode range queries using R and M-trees. Our results suggest the best strategy is the one which groups every sequence in a single group and performs a single range query. For range queries that retrieves considerable amount of results, the use of grouping strategies greatly improves its performance by using any of the addressed trees. We suggest this is due to the ratio of disk accesses per query is very high and the number of calculations needed is similar to performing the naive approach. We noted the number of sequences in a single batch is an important factor to perform multiple range queries simultaneously. The larger the batch, the less disk accesses are performed. Considering all results, we suggest usage of single grouping strategy for all scenarios except for querying R with $\epsilon = 0$.

References

Camoglu, O., Kahveci, T., and Singh, A. K. (2003). Towards index-based similarity search for protein structure databases. In *Computational Systems Bioinformatics*, pages 148–158. IEEE.

Ciaccia, P., Patella, M., and Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. In *VLDB '97*, pages 426–435. M. K. Publishers Inc.

Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *SIGMOD Rec.*, volume 23, pages 419–429. ACM.

Fu, A. W.-C., Keogh, E., Lau, L. Y., Ratanamahatana, C. A., and Wong, R. C.-W. (2008). Scaling and time warping in time series querying. In *The VLDB Journal*, volume 17, pages 899–921. Springer-Verlag New York, Inc.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *SIGMOD Rec.*, volume 14, pages 47–57. ACM.

Kotsifakos, A., Karlsson, I., Papapetrou, P., Athitsos, V., and Gunopulos, D. (2015). Embedding-based subsequence matching with gaps-range-tolerances: A query-by-humming application. In *The VLDB Journal*, volume 24, pages 519–536. Springer-Verlag New York, Inc.

Moon, Y.-S., Whang, K.-Y., and Loh, W.-K. (2001). Duality-based subsequence matching in time-series databases. In *Proc. 17th Int. Conf. Data Engineering*, pages 263–272. IEEE.

Orchard, M. T. (1991). A fast nearest-neighbor search algorithm. In *ICASSP 91: 1991 Int. Conf. Acoustics, Speech, and Signal Processing*, volume 4, pages 2297–2300. IEEE.

Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. In *Expert Systems with Applications*, volume 36, pages 3336–3341. Elsevier.

[1] Figure is not shown due to space constraints

paper:175059

Estratégia Distribuída para Análise de Assuntos Abordados no Twitter Via Evolução de Clusters

Priscila Rocha Ferreira Rodrigues[1], Ticiana L. Coelho da Silva[2], José Maria da Silva M. Filho[1], José Antônio F. de Macêdo[1]

Universidade Federal do Ceará (UFC)

[1]Fortaleza – CE – Brasil
[2]Quixadá – CE – Brasil

priscila.rfr@gmail.com,

{ticianalc, monteiro, jose.macedo}@dc.ufc.br

Abstract. *Recent techniques have applied algorithms of clusters evolution to analyze transitions of subjects in social networks and present themselves effective in the monitoring of these. However, the high rate of data production in social networks creates the need to process an increasing amount of data. This paper proposes a more scalable strategy for analyzing the evolution of subjects in social networks, through the use of a distributed solution in the data clustering stage. The experiments were performed using data obtained from Twitter and demonstrate that the proposed solution is promising, presenting considerable gains in performance.*

Resumo. *Recentes técnicas têm aplicado algoritmos de evolução de clusters para analisar transições de assuntos em redes sociais e apresentam-se eficazes no monitoramento destes. No entanto, a elevada taxa de produção de dados nas redes sociais cria a necessidade de processamento de uma quantidade de dados cada vez maior. Este trabalho propõe uma estratégia mais escalável para análise da evolução de assuntos em redes sociais, por meio do emprego de uma solução distribuída na etapa de clustering dos dados. Os experimentos foram realizados utilizando dados obtidos do Twitter e demonstram que a solução proposta é promissora, apresentando ganhos consideráveis de desempenho.*

1. Introdução

Considerando a relevância do monitoramento de assuntos e comunidades em redes sociais ao longo do tempo, referenciados em [Kim e Han 2009], [Lee et al 2014], [Tajeuna et al 2015] e [Rodrigues et al 2016], algumas técnicas têm sido propostas para automatizar esse tipo de análise. Dentre essas técnicas, destaca-se a evolução de *clusters,* que consiste na investigação das alterações que um *cluster* pode sofrer ao longo do tempo em virtude da constante evolução e produção dos dados. O acompanhamento da repercussão de determinado evento, *feedback* sobre algum produto, elaboração de melhores estratégias de marketing e melhoria de serviços, são algumas das finalidades que podem ser alcançadas por meio do emprego desse método em dados produzidos nas redes sociais.

[Rodrigues et al 2016] demonstra a aplicabilidade da utilização da estratégia de evolução de *clusters* em bases de dados altamente dinâmicas como as de redes sociais.

Utilizando o algoritmo DBSCAN [Ester et al 1996], [Rodrigues et al 2016] aplicou *clustering* em uma base de dados composta por *tweets*, detectando os assuntos repercutidos no *dataset* analisado. Por meio do monitoramento da evolução dos *clusters*, verificou-se a dinâmica e transição dos assuntos detectados, apresentando uma visão panorâmica que permitiu compreender as motivações de tais evoluções. Através do algoritmo de evolução de *clusters* empregado em [Rodrigues et al 2016] foi possível detectar as seguintes transições: Criação, Sobrevivência, Desaparecimento, União, Divisão, Expansão e Retração de *clusters.*

Apesar dos bons resultados apresentados no trabalho supramencionado, ressalta-se que a elevada taxa de produção de conteúdo em redes sociais cria a necessidade de processamento de uma quantidade de dados cada vez maior. Logo, a utilização de algoritmos de alto custo computacional, como o DBSCAN, pode agravar ainda mais o desempenho da análise de grandes volumes de dados [Gan e Tao 2015], e este é um problema em aberto na proposta de [Rodrigues et al 2016]. Problemas que fazem uso intensivo de CPU em suas resoluções ou que processam grandes volumes de dados são essencialmente resolvidos por meio da distribuição de tarefas entre vários núcleos de processamento.

Motivados pelos benefícios que o monitoramento de assuntos numa janela temporal propicia a diversas áreas, e com o intuito de estender a proposta de [Rodrigues et al 2016], este trabalho propõe uma solução mais escalável para o monitoramento dos assuntos em redes sociais via evolução de *clusters*. Na etapa de *clustering* dos dados, é proposta uma abordagem distribuída, aplicando o algoritmo DBSCAN através do paradigma MapReduce, com o intuito de obter melhor desempenho na análise da evolução de assuntos ao longo do tempo. As observações obtidas até o momento, por meio de experimentos utilizando *tweets*, apontam que a solução proposta é adequada, apresentando uma melhoria considerável no tempo de execução quando comparada com a abordagem centralizada.

Este artigo está organizado da seguinte forma: Na Seção 2, é descrita a abordagem distribuída que está sendo proposta para acompanhamento da evolução dos assuntos nas redes sociais. Na Seção 3 é apresentada a análise dos resultados preliminares. Os trabalhos relacionados estão descritos na seção 4. Finalmente, a Seção 5 apresenta as conclusões obtidas até o momento e as direções futuras desta pesquisa.

2. Detecção Distribuída de *Clusters* e suas Evoluções

A estratégia para o monitoramento da evolução de *clusters* proposta por [Rodrigues et al 2016], consiste em 3 etapas principais: coleta e pré-processamento dos dados, *clustering* e evolução de *clusters*. Na primeira etapa, os dados são coletados da rede social *r* e em seguida, são aplicadas técnicas de processamento de linguagem natural para limpeza dos dados. Na etapa seguinte, para o emprego da técnica de *clustering*, o algoritmo DBSCAN foi implementado e aplicado utilizando como base a medida de similaridade *Fading* [Lee et al 2014]. Tal medida faz uso do atributo de tempo para o cálculo da similaridade, considerando que publicações mais próximas no tempo têm maior possibilidade de versarem sobre o mesmo assunto. O DBSCAN recebe como entrada um conjunto de dados X, o tamanho da vizinhança denominado *eps* e o número mínimo de pontos de uma vizinhança, denotado *minPts*. O algoritmo classifica os dados em *core*, *border* ou *outlier*. Um ponto é *core* se possuir uma vizinhança maior ou igual do que o estabelecido em *minPts*. É considerado *border* se pertence à vizinhança de um ponto *core*, ou seja, se a distância entre eles é menor que *eps*. Pontos *outliers* são pontos atípicos ou inconsistentes que, por não pertencerem a *cluster* nenhum, são descartados [Ester et al 1996].

Para detecção das evoluções sofridas pelos *clusters* ao longo dos dias, foi implementado e executado o algoritmo de evolução proposto por [Silva et al 2014]. Através de comparações entre os *clusters*, o algoritmo detecta criação, sobrevivência, desaparecimento, união, divisão, expansão e retração de *clusters.* Considerando que das 3 etapas da estratégia de Evolução de *Clusters* apresentadas, a etapa de *clustering* é a que apresenta uma maior necessidade de processamento, fazendo uso de um considerável custo computacional, aplicamos o paradigma MapReduce [Dean e Ghemawat 2004] na etapa de *clustering* dos dados. Tal paradigma destaca-se como uma das abordagens mais utilizadas para solucionar problemas de processamento de grandes conjuntos de dados, uma vez que foi projetado com a finalidade de tornar simples e eficiente o processamento distribuído através de um conjunto de computadores comuns. Dentre as implementações mais conhecidas do paradigma MapReduce destaca-se o framework de código aberto Hadoop [White 2009], desenvolvido pela Apache Software Foundation, e referenciado recentemente na literatura em trabalhos que têm aplicado soluções distribuídas do DBSCAN para domínios diversos [He et al 2011], [Dai et al 2012] e [Neto et al 2015]. A execução de tarefas nos nós de processamento e a tolerância a falhas são de total responsabilidade do framework, restando ao desenvolvedor apenas a programação das funções Map e Reduce, além da configuração da infraestrutura.

A função Map recebe todos os dados gerando um conjunto intermediário de dados, no formato <chave, valor>. A função Reduce é executada para cada chave intermediária gerada, com todos os conjuntos de valores intermediários associados aquela chave combinados. Em geral, a tarefa de mapeamento é usada para selecionar um conjunto de valores, e a tarefa de redução é usada para fazer a sumarização do resultado. Além disso, em processamentos distribuídos é importante que as partições de dados a serem processadas tenham tamanhos aproximadamente iguais, uma vez que o tempo total de processamento é delimitado pelo tempo que o nó com uma maior quantidade de dados leva para finalizar a computação dos dados a ele atribuídos. Logo, a base de dados a ser analisada é particionada de forma proporcional com base na quantidade de *tweets* nela contidos.

As funções Map e Reduce definidas para tornar a proposta de [Rodrigues et al 2016] escalável são explicadas a seguir:

1. **Map–** O algoritmo DBSCAN é aplicado ao conjunto de dados. Em seguida, cada *cluster* é descrito como um par <chave, valor>, onde a chave corresponde ao *tweet core* e o valor corresponde aos *tweets borders* do *cluster*.

2. **Reduce–** Essa função recebe como entrada o conjunto de pares de *clusters* gerados anteriormente e aplica a medida de similaridade *Fading* nos pontos *cores* desses *clusters*, unindo possíveis *clusters* do mesmo assunto que foram detectados em partições diferentes. Os pares de *clusters* gerados nessa etapa serão utilizados na etapa posterior pelo algoritmo que detecta as transições dos *clusters*.

Os Algoritmos 1e 2 mostram como é feita a implementação das funções Map e Reduce descritas acima, respectivamente.

Algoritmo 1: Map

```
Entrada: Conjunto de dados D,minPoints,eps
Saída: Conjunto PC de pares de clusters
1Início
2 |  DBSCAN (P, eps, minPoints)
3 |  retorna PC;
4fim
```

Algoritmo 2: Reduce

```
Entrada: Conjunto PC de pares de clusters <c, b>, eps
Saída: Conjunto PC de pares de clusters após uniões
1Início
2 |  para c1∈ PC faça
3 |       para c2∈ PC e c2 ≠ c1 faça
4 |            se SimilaridadeFading (c1,c2) ≥ eps então
5 |            |  PC <- cx<- União (c1,c2)
6 |            |  PC-> Remove (c1,c2)
7 |            fim
8 |       fim
9 |  fim
10|  retorna PC;
11fim
```

3. Análise dos Resultados Preliminares

Os dados do Twitter foram coletados entre os dias 17/05/2017 e 23/05/2017 por meio de um *crawler* que recebeu como parâmetro a palavra "Temer" e retornou *tweets* que continham essa palavra. Foram coletados 2.254.628 *tweets*, o que resultou em aproximadamente 2,4 Gb de dados. Para estratégia distribuída, os experimentos foram realizados em uma infra-estrutura de nuvem privada com 8 máquinas virtuais, 4 núcleos (vCPU), 8G de RAM e 80GB de disco cada. No experimento centralizado, apenas uma dessas máquinas foi utilizada. Os parâmetros utilizados nos algoritmos, são os mesmos empregados em [Rodrigues et al 2016]. Logo, isso corresponde a *eps*: 0.3 e *minPts*: 0.25 * (quantidade de *tweets* do snapshot). Cada snapshot corresponde a um dia de coleta.

A Figura 1 apresenta os assuntos que foram detectados e a maneira como esses assuntos repercutiram e se relacionaram ao longo dos dias. Foram detectados os mesmos *clusters* e transições para ambas as estratégias aplicadas: com *clustering* centralizado e distribuído. No dia 17/05 foi identificado um assunto que se tratava sobre a divulgação de uma gravação que o Dono da JBS, Joesley Batista, havia feito em uma conversa com Michel Temer que se tratava do pagamento pelo silêncio do ex-deputado Eduardo Cunha[1]. No dia seguinte, 18/05, esse tópico sofreu uma **divisão**, entre o assunto que se tratava das manifestações contrárias ao presidente Temer ocorridas no país[2] e o assunto que repercutia o afastamento de Aécio Neves do cargo de senador[3]. No dia 19/05 ambos os assuntos passaram a repercutir juntos (**união**) sobre um possível impeachment de Michel Temer. Em 20/05 esse assunto **expandiu,** porém nos dias subsequentes passou a sofrer pequenas **retrações**. No dia 22/05 surgiu também um pequeno *cluster* que se tratava do atentado ocorrido em Londres no *show* da cantora Ariana Grande[4]. Usuários ironizavam falando que o atentado deveria ter ocorrido no Congresso Nacional.

Apesar de nesse estudo de caso, os *clusters* gerados utilizando as abordagens centralizada e distribuída terem produzido mesmos resultados, é possível que em outros experimentos possamos obter resultados distintos. Isto ocorrerá quando determinado *tweet* classificado *border* em uma partição, poderia se tornar um *tweet core* se fosse comparado a outra partição de dados. Apesar dessa possibilidade, observa-se que a estratégia distribuída é, em média, 10,3 vezes mais rápida que a abordagem centralizada. A Figura 2 apresenta um gráfico com o tempo de processamento do *clustering* para cada *dataset* analisado, comparando os resultados entre as abordagens empregadas.

[1] https://oglobo.globo.com/brasil/dono-da-jbs-grava-temer-dando-aval-para-compra-de-silencio-de-cunha-21353935

[2] http://www.opovo.com.br/noticias/politica/2017/05/protestos-acontecem-em-cidades-apos-divulgacao-de-denuncia-contra-teme.html

[3] http://g1.globo.com/jornal-nacional/noticia/2017/05/aecio-neves-e-afastado-da-funcao-de-senador-e-deixa-presidencia-do-psdb.html

[4] http://g1.globo.com/jornal-nacional/noticia/2017/05/explosao-mata-19-pessoas-em-show-de-ariana-grande-na-inglaterra.html

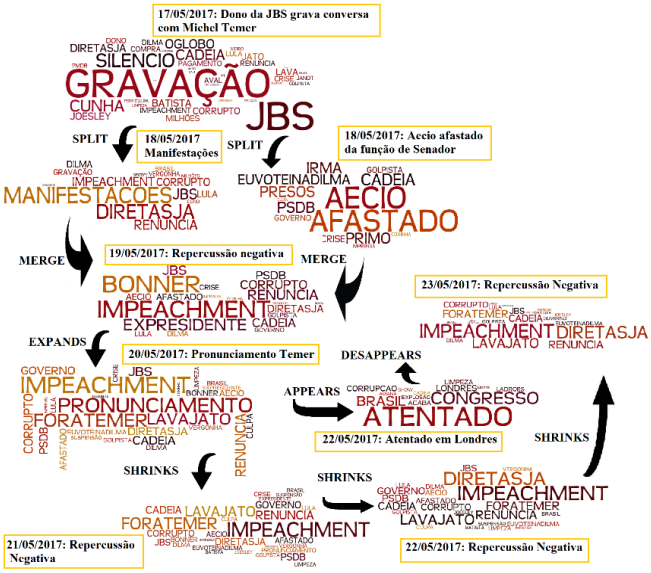


Figura 1- Gráfico de transições dos assuntos sobre "Temer" - 17/05/2017 à 23/05/2017

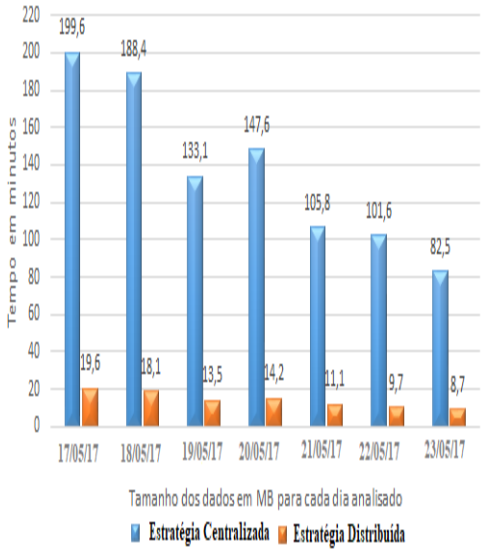


Figura 2- Tempo de execução dos algoritmos na etapa de clustering

4. Trabalhos Relacionados

Em [Tajeunaet al 2015], técnicas de evolução de *clusters* são aplicadas em dados de redes sociais com a finalidade de acompanhar a evolução de grupos de usuários ao longo do tempo. Além disso, o trabalho propõe uma medida de similaridade com o intuito de capturar eventos de transição significativos para evolução de comunidades. No entanto, além de [Tajeuna et al 2015] aplicar a estratégia de evolução de *clusters* para o contexto de comunidades, e não de assuntos como no trabalho em questão, sua abordagem é centralizada. Já [Neto et al 2015] estuda o problema de processar o algoritmo de *clustering* DBSCAN de maneira distribuída através do paradigma MapReduce. Sua estratégia propõe também uma abordagem de particionamento de dados, que usa estruturas de grade e grafo para auxiliar na divisão do espaço em regiões de baixa densidade. No entanto, [Neto et al 2015] direciona sua estratégia para o contexto de dados espaciais, o que torna sua abordagem, em muitos pontos, diferente da proposta neste trabalho, tendo em vista as características específicas de ambos os contextos. [Dai e Lin 2012] e [He et al 2011], também abordaram o problema de distribuição do DBSCAN, no entanto, diferente deste trabalho, as técnicas propostas requerem que parte dos dados sejam replicados nas partições.

Por fim, em [Rodrigues et al 2016] são apresentados resultados sobre a efetividade do uso de técnicas de evolução de *clusters* para monitorar a transição de assuntos em redes sociais. [Rodrigues et al 2016] apresentou a evolução de temas que giraram em torno da ex-presidenta Dilma Rousseff em 2015. No entanto, sua estratégia é centralizada e não aborda questões de desempenho. Além disso, os autores destacam que é importante que outros estudos de caso sejam realizados para validar a estratégia proposta. Sendo assim, o presente trabalho visa estender a pesquisa iniciada por [Rodrigues et al 2016], propondo uma abordagem distribuída para a *clustering* dos dados e realizando um novo estudo de caso em um *dataset* com *tweets* coletados em 2017.

5. Conclusões e Direções Futuras

Os resultados apresentados mostram um ganho considerável de desempenho na estratégia de evolução de *clusters*, ao aplicar uma abordagem distribuída na etapa de *clustering* dos dados. É ainda reforçado que a aplicabilidade das técnicas de evolução de *cluster* para

monitorar as transições de assuntos em redes sociais é válida e promissora. Ressalta-se que a aplicação dessas estratégias é recente no contexto de redes sociais, e a apresentação de uma abordagem distribuída para este cenário dá abertura para o surgimento de novas estratégias de distribuição de processamento, que podem tirar proveito desse modelo para abordagens mais eficientes. Posteriormente, pretende-se realizar um estudo mais profundo da estratégia distribuída em comparação com a centralizada, a fim de garantir que ambas gerem os mesmos resultados. Adiante, objetiva-se também analisar a variação dos parâmetros utilizados no DBSCAN para este contexto de *clustering* em redes sociais, além de realizar experimentos que verifiquem o desempenho da estratégia distribuída com diferentes quantidades de máquinas.

Referências

Dai, B. R., & Lin, I. C. (2012). Efficient map/reduce-based dbscan algorithm with optimized data partition. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on (pp. 59-66). IEEE.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd(Vol. 96, No. 34, pp. 226-231).

Gan, J., & Tao, Y. (2015). DBSCAN revisited: mis-claim, un-fixability, and approximation. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (pp. 519-530). ACM.

He, Yaobin, et al. (2011) "Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce." Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on. IEEE.

Kim, M. S., & Han, J. (2009). A particle-and-density based evolutionary clustering method for dynamic networks. Proceedings of the VLDB Endowment, 2(1), 622-633.

Lee, P., Lakshmanan, L. V., & Milios, E. E. (2014). Incremental cluster evolution tracking from highly dynamic network data. In Data Engineering (ICDE), 2014 IEEE 30th International Conference on (pp. 3-14). IEEE.

Neto, Antonio Cavalcante Araujo, et al. (2015) "G2P: A partitioning approach for processing dbscan with mapreduce." International Symposium on Web and Wireless Geographical Information Systems. Springer, Cham.

Rodrigues, Priscila R.F. et al. (2016)“Dinâmica de Temas Abordados no Twitter Via Evolução de Clusters.” Proceedings of the 31thSBBD, p. 151-157.

Silva, Ticiana L. Coelho, José AF de Macêdo, and Marco A. Casanova. (2014) "Discovering frequent mobility patterns on moving object data." Proceedings of the Third ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems.ACM.

Tajeuna, Etienne Gael, Mohamed Bouguessa, and ShengruiWang. (2015) "Tracking the evolution of community structures in time-evolving social networks." DSAA, 2015. 36678 2015.IEEE International Conference on.IEEE,.

White, T. (2012). Hadoop: The definitive guide. " O'Reilly Media, Inc.".

paper:174665

Mineração de Perfis Sociais em Redes Temporais

Jeancarlo C. Leão, Michele A. Brandão, Pedro O. S. Vaz de Melo, Alberto H. F. Laender

Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – MG

{jcl,micheleabrandao,olmo,laender}@dcc.ufmg.br

Abstract. *The number and types of relationship that members of a social network establish suggest the profile that each individual develops over time. In this paper, we propose a mining method capable of identifying and recovering social profiles from topological aspects in temporal networks. The application of this method allows the identification of persistent profiles and explicit patterns that suggest relationship models in social networks.*

Resumo. *A quantidade e os tipos de relacionamento que os membros de uma rede social estabelecem sugerem o perfil que cada indivíduo desenvolve ao longo do tempo. Neste artigo, propomos um método de mineração capaz de identificar e recuperar perfis sociais a partir de aspectos topológicos em redes temporais. A aplicação desse método permite identificar perfis persistentes e explicitar padrões que sugerem modelos de relacionamento em redes sociais.*

1. Introdução

Uma rede social pode ser definida como uma estrutura composta de entidades em que o conjunto de interações entre os pares caracterizam algum tipo de relação. Tais redes podem ser modeladas por meio de um grafo, no qual os vértices representam as entidades e as arestas as interações ocorridas entre elas ao longo do tempo. Apesar do grande volume de dados existentes sobre redes sociais, eles geralmente são subutilizados em razão do pouco aproveitamento de suas propriedades temporais. Embora seja um desafio considerar a dimensão temporal, a topologia de uma rede social pode ser analisada em intervalos de tempo. Tal análise revela como cada entidade se relaciona, possibilitando determinar a sua tendência de interação ou perfis de relacionamento social. Nesse contexto, as informações topológicas e temporais de uma rede permitem responder à questão fundamental sobre qual seria o perfil social de seus nodos.

O estudo sobre perfis sociais é motivado pela escassez de abordagens para caracterização dos membros de uma rede social considerando apenas os seus aspectos topológicos e temporais. A obtenção de perfis sociais contribui para a quantificação do seu capital social[1], bem como permite caracterizar e distinguir diferentes redes sociais. Assim, são contribuições deste trabalho duas técnicas de mineração de redes sociais que atuam sobre classes de relações para obter o perfil social de relacionamento de cada um de seus

[1]O capital social de uma rede é definido como sendo os recursos a ela incorporados que podem ser utilizados para obter melhorias como aumento do fluxo de informação, identificação de posições estratégicas na rede e elevação da credencial social de seus indivíduos [Lin 1999].

membros, e que foram avaliadas em redes sociais reais com diferentes características e de domínios distintos.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 descreve trabalhos relacionados. A Seção 3 apresenta a metodologia adotada e uma breve caracterização das redes sociais utilizadas, introduzindo também a técnica adotada para classificar os seus relacionamentos. A seguir, a Seção 4 descreve as técnicas propostas para obtenção de perfis a partir das relações classificadas e a Seção 5 discute os resultados experimentais obtidos. Finalmente, a Secão 6 apresenta conclusões e comenta sobre trabalhos futuros.

2. Trabalhos Relacionados

Propriedades e padrões topológicos podem ser explorados para medir e distinguir tipos de relacionamento em redes sociais. Nesse contexto, David et al. [2010] analisam as funções que as arestas de uma rede desempenham em sua estrutura. Além disso, Granovetter [1973] define a força dos relacionamentos sociais a partir da estrutura topológica de uma rede. Tal conceito é utilizado por Alves et al. [2016] para medir a força dos relacionamentos em redes sociais de desenvolvedores e por Leão et al. [2017] para filtrar ruído informacional causado pela aleatoriedade em relações sociais.

O perfil social dos indivíduos de uma rede pode ser aplicado na obtenção de capital social mesmo quando os atributos desses indivíduos não estão acessíveis. Estudos em diferentes áreas propõem o uso das propriedades topológicas em aplicações distintas. Burt et al. [1995] argumentam que é útil se concentrar no padrão de relações entre pessoas. Podolny & Baron [1977] complementam o argumento de Burt et al. [1995] e apoiam-se também na teoria do capital social em que redes com alto fechamento (*closure*) e coerência (*structural holes*) são propícias à criação de uma identidade social que explica as expectativas por trás do seu papel.

Este trabalho baseia-se na força dos laços definida por Granovetter [1973] que pode ser obtida em redes temporais utilizando-se a técnica de classificação de relações proposta por Vaz de Melo et al. [2015]. Para isso, perfis sociais são definidos por meio da representatividade e importância dos relacionamentos para responder ao questionamento deixado por David et al. [2010] em relação aos papéis que diferentes nós desempenham na rede. Dessa forma, este trabalho propõe uma abordagem conjunta de meios já utilizados para estudo do capital social [Cohen et al. 2014; David et al. 2010] e revela informações ainda não observadas na literatura sobre os perfis sociais.

3. Metodologia

Dado um grafo não direcionado $G = (V, E)$, no qual $V = \{v_1, ..., v_n\}$ é o conjunto de vértices e $E = \{e_1, ..., e_m\}$ é o conjunto de arestas que representam as interações entre dois vértices, cada grafo temporal $G_t(V_t, E_t)$ em G representa a agregação das interações em períodos discretos de tempo t. Assim, para um dado valor de t, V_t inclui todos os vértices que interagem no período de tempo t. Analogamente, as arestas do conjunto E_t representam as interações entre os pares de vértices (v_i, v_j) durante o período de tempo t.

Esse modelo de grafo temporal foi aplicado a redes de colaboração científica derivadas de dados da APS, DBLP e PubMed[2], a redes $WiFi$ de mobilidade em *campi*

[2]APS (`http://www.aps.org/`), PubMed (`http://www.ncbi.nlm.nih.gov/pubmed`) e DBLP (`http://dblp.org/`)

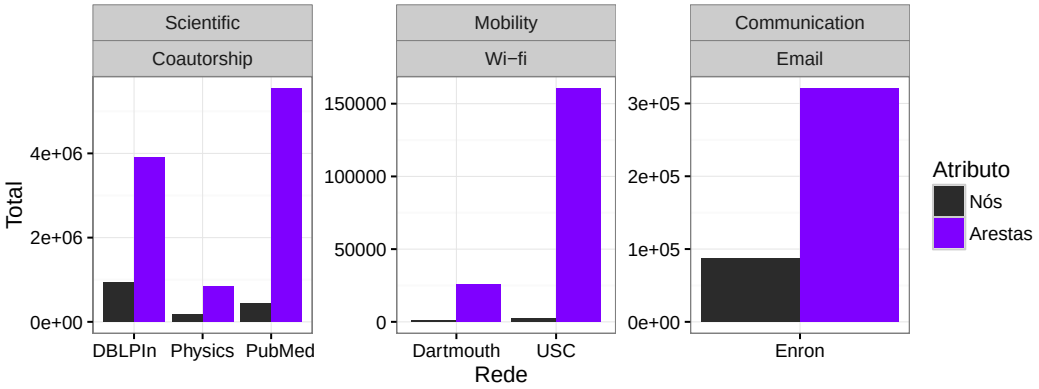


Figura 1: Características das redes sociais temporais.

Tabela 1: Classe atribuída quando as métricas estão acima (+) ou abaixo (-) do limiar l. Fonte (adaptado): [Vaz de Melo et al. 2015].

Métrica comparada com limiar l	Classe *Amizade* (Forte e Persistente)	Classe *Conhecido* (Forte e Rara)	Classe *Ponte* (Fraca e Persistente)	Classe *Aleatório* (Fraca e Rara)
Neighborhood Overlap (NO)	+	+	-	-
Persistence (P)	+	-	+	-

universitários (Dartmouth e USC) e a uma rede de comunicação de *e-mails* entre funcionários da Enron[3] entre 1999 e 2003. A Figura 1 apresenta as características dessas redes, onde os nós representam, respectivamente, pesquisadores, usuários do sistema e funcionários, e as arestas coautorias, pontos de acesso de rede sem fio e *e-mails* individuais.

O problema de classificação de relações em redes sociais consiste em atribuir um rótulo do conjunto $R = \{r_1, r_2, ..., r_k\}$ a cada par de vértices $r(v_i, v_j)$. Assim, são considerados aspectos topológicos e temporais para determinar qual rótulo será atribuído à relação [Vaz de Melo et al. 2015]. De fato, estudos sociológicos revelam que a topologia em que pares (i, j) de indivíduos estão envolvidos sugere o tipo ou classe de relação entre eles [David et al. 2010; Granovetter 1973]. Neste trabalho, a classificação é obtida pela métrica *neighborhood overlap (NO)* [David et al. 2010] que considera os N vizinhos de um nodo (Equação 1) e pela métrica de persistência P dos relacionamentos que é dada pelo somatório dos intervalos de tempo em que ocorrem alguma interação entre os pares i e j (Equação 2).

$$NO(i,j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j| - 2} \quad (1) \qquad P(i,j) = \frac{1}{t}\sum_{k=1}^{t}[(i,j) \in \varepsilon_k] \quad (2)$$

Para classificar as relações foi utilizado o algoritmo *RECAST* (*Random Relationship Classifier Strategy*) que é uma técnica de classificação de arestas em redes temporais [Vaz de Melo et al. 2015]. O *RECAST* baseia-se nas duas características que estão sempre presentes nas relações sociais: a persistência (P) das interações e o compartilhamento de vizinhos entre pares (NO) para atribuir a cada relação uma das classes, conforme comparação dos valores dessas métricas com um limiar l (Tabela 1).

[3]Enron email dataset: `https://www.cs.cmu.edu/~./enron/`

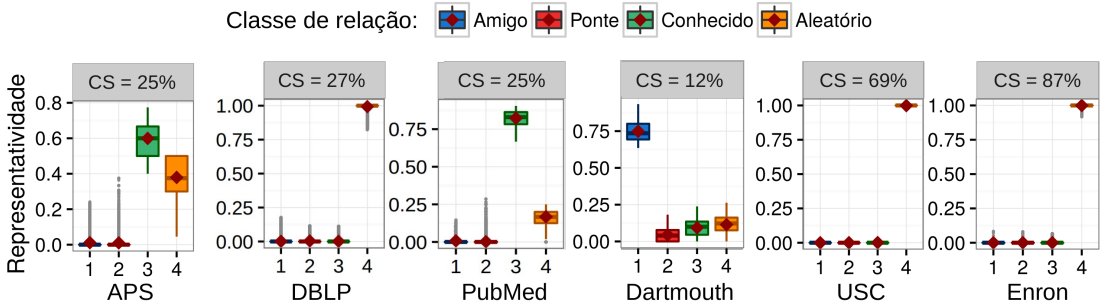


Figura 2: Perfis que caracterizam o maior grupo de indivíduos em cada rede.

4. Técnicas de Mineração de Perfis

O perfil social de um dado nodo de uma rede é caracterizado pela quantidade de relações que possui em cada classe e pode ser interpretado como a probabilidade que o nodo tem de estabelecer cada tipo de relação social. Dado o conjunto de todos os relacionamentos $R_i = \{r_1, r_2, ..., r_n\}$ de um indivíduo i e sendo $R_{ic} \in R_i$ um subconjunto contendo todas as suas relações da classe c, definimos a representatividade das relações da classe c para o perfil do indivíduo i pela equação $Rep(i, c) = |R_{ic}|/|R_i|$. A importância $Imp(i, c)$ do conjunto de relacionamentos R_{ic} é relativa ao nodo j que possui maior número de relacionamentos da classe c na rede. Essa importância é obtida pela equação $Imp(i, c) = |R_{ic}|/|R_{jc}|$. Finalmente, o perfil de um indivíduo i é o conjunto formado pelas representatividades das quatro classes de relacionamentos.

Neste trabalho, propomos duas técnicas de recuperação de perfis. A primeira técnica permite identificar perfis existentes na rede através do agrupamento de seus nós pela representatividade de seus relacionamentos calculada por $Rep(i, c)$. A segunda técnica proposta se baseia no uso da métrica de importância de relacionamentos ($Imp(i, c)$) para obtenção de uma ordenação por classe para cada membro da rede. Assim, torna-se possível recuperar perfis relevantes a partir de uma consulta onde são especificados os níveis de importância de cada classe de relacionamento. Por exemplo, para elaborar uma consulta Q_1 que recupere os *perfis com relacionamentos mais importantes da classe* c, basta especificar um nível de importância alto para a classe c e baixo para as demais classes.

5. Resultados Experimentais

Os principais resultados deste trabalho são apresentados em dois segmentos de aplicação das técnicas propostas para mineração de perfis sociais: mineração de perfis similares e recuperação dos perfis com os relacionamentos mais importantes da rede. Em cada uma dessas aplicações, são analisadas também as características das redes utilizadas.

Mineração de Perfis Sociais Similares. A técnica de mineração de perfis sociais similares objetiva identificar os principais perfis sociais assumidos por membros de redes sociais com base nas classes de relacionamentos que possuem. Para isso é considerada a semelhança na representatividade de seus relacionamentos em cada classe.

Assim, para cada rede foram obtidos 10 agrupamentos de perfis dos quais os maiores (aqueles que possuem maior percentual CS de indivíduos da rede) são apresentados na Figura 2. É possível observar que nas redes DBLP, USC e Enron os maiores perfis

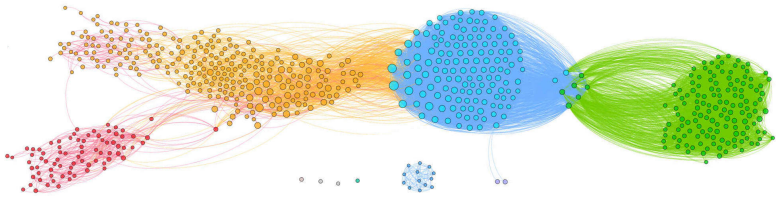


Figura 3: Subgrafo da rede PubMed com os indivíduos com perfis mais extremos.

possuem na maioria de suas relações interações raras ($P < l$) e pouco similares ($NO < l$), sendo, portanto, classificados como *aleatórios*. Por outro lado, o maior perfil na rede Dartmouth é extremamente social, pois seus membros mantêm, em sua maioria, relações regulares com pessoas com quem compartilham vizinhos na rede ($P > l$ e $NO > l$). As redes PubMed e APS, por sua vez, possuem um perfil com proporção considerável de relacionamentos da classe $conhecido$. Isto conclui que o maior grupo dessas redes mantém relações de coautoria pouco frequentes com um número relativamente grande dos coautores de seus coautores. Grupos que mantêm relações predominantemente de $ponte$ são raros em todas as redes utilizadas neste trabalho a ponto de inviabilizar a formação de um grupo próprio utilizando o algoritmo *K-means* com $k = 10$. Isso confirma que pontes são raras em redes reais com propriedades de *Mundo Pequeno* [David et al. 2010].

Recuperação de Perfis Sociais Relevantes. Esta abordagem permite recuperar membros da rede com perfis relevantes para uma dada consulta definida pelos níveis de importância de cada classe de relacionamento. Por exemplo, ao consultar níveis de importância extremos, são selecionados membros com a importância mais expressiva na rede em cada classe. Na Figura 3, o resultado dessa consulta sobre a rede PubMed é apresentado como um subgrafo com grupos de perfis bem definidos e persistentes. Ou seja, para cada classe de relacionamento, os nós presentes nesse subgrafo são os que possuem maior importância em suas relações de amizade (em azul), com conhecido (verde), de ponte (vermelho) ou aleatórias (laranja) em toda a rede. Os atributos profissionais de alguns dos membros das redes de colaboração científica foram inspecionados individualmente revelando grupos de trabalho com padrões bem definidos para as diferentes classes de relacionamentos. Grupos com membros de mesmo perfil apresentam características topológicas similares em densidade, grau e coeficiente de agrupamento mesmo considerando sua participação em toda a rede. Ademais, é possível observar na Figura 3 que, no maior componente conectado, existe um grande grupo de membros altamente socializados (em azul), pois são persistentes e interagem muito e predominantemente com os demais membros desse grupo. Esses membros possuem características topológicas[4] muito semelhantes entre si sugerindo forte influência social ou homofilia [David et al. 2010].

6. Conclusões

A contribuição fundamental deste trabalho está na modelagem da representatividade e da importância dos relacionamentos sociais. Por meio da aplicação dessas métricas foram concebidas técnicas para obtenção de perfis sociais que foram analisados em seis redes sociais reais. Dentre esses perfis, foram apresentados os que pertencem ao maior nú-

[4]As métricas topológicas locais foram medidas na rede completa. Dentre elas estão: centralidade, grau, coeficiente de clusterização e classe de modularidade.

mero de indivíduos, demonstrando diferenças no tipo de relacionamento que prevalece em cada domínio. Ademais, observamos que em redes de colaboração científica o grupo com maior número de indivíduos possui perfis diferentes, o que permite caracterizar e distinguir essas redes. Além disso, a técnica de recuperação de perfis relevantes permite elaborar consultas e extrair da rede o conjunto de membros com um determinado perfil de relacionamento. Os resultados também mostram como perfis com características mais extremas também podem ser obtidos para cada uma das classes de relacionamentos. As propriedades dos membros das redes, como a sua função profissional, grupo de trabalho ou área de atuação, foram analisadas permitindo relacionar os papéis reais desses indivíduos com o perfil obtido através de características puramente topológicas. Em trabalhos futuros, pretendemos verificar se os padrões dos relacionamentos e as similaridades topológicas observados em grupos de indivíduos com perfis extremos e similares podem ser explicados pela homofilia ou pela influência social.

Agradecimentos. Este trabalho é apoiado pelos projetos InWeb (processo MCT/CNPq 573871/2008-6) e MASWeb (processo FAPEMIG/PRONEX APQ-01400-14), e por auxílios individuais de pesquisa concedidos aos autores pelo CNPq, Fapemig e CAPES. Particularmente, o primeiro autor agradece ao Instituto Federal do Norte de Minas Gerais - IFNMG, pela concessão de uma bolsa de estudos no âmbito do Programa de Bolsas para Qualificação de Servidores (PBQS).

Referências

Alves, G. B., Brandão, M. A., Santana, D. M., da Silva, A. P. C., and Moro, M. M. (2016). The Strength of Social Coding Collaboration on GitHub. In Anais *do 31º Simpósio Brasileiro de Banco de Dados*, pages 247–252, Salvador, Bahia, Brasil.

Burt, R. S. (1995). *Structural Holes: The Social Structure of Competition*. Harvard University Press, Cambridge.

Cohen, E., Delling, D., Pajor, T., and Werneck, R. F. (2014). Computing classic closeness centrality, at scale. In Proceedings *of the Second ACM Conference on Online Social Networks*, pages 37–50, Dublin, Ireland.

David, E., Jon, K., Easley, D., and Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA.

Granovetter, M. S. (1973). The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380.

Leão, J. C., Brandão, M. A., de Melo, P. O. V., and Laender, A. H. F. (2017). Classificação de relações sociais para melhorar a detecção de comunidades. In Proceedings *of the VI Brazilian Workshop on Social Network Analysis and Mining*, São Paulo, SP, Brazil.

Lin, N. (1999). Building a network theory of social capital. *Connections*, 22(1):28–51.

Podolny, J. M. and Baron, J. N. (1997). Resources and relationships: Social networks and mobility in the workplace. *American Sociological Review*, 62:673–693.

Vaz de Melo, P. O. S., Viana, A. C., Fiore, M., Jaffrès-Runser, K., Mouël, F. L., Loureiro, A. A. F., Addepalli, L., and Guangshuo, C. (2015). RECAST: Telling Apart Social and Random Relationships in Dynamic Networks. *Performance Evaluation*, 87:19–36.

paper:174664

Mineração de Regras de Associação Multirrelação em Grafos: Direcionando o Processo de Busca

Felipe A. de Oliveira[1]**; Raquel L. Costa**[2]**; Ronaldo R. Goldschmidt**[1]**; Maria C. Cavalcanti**[1]

[1] Departamento de Sistemas e Computação
Instituto Militar de Engenharia (IME) – Rio de Janeiro, RJ – Brasil

[2]Instituto Nacional do Câncer (INCA) – Rio de Janeiro, RJ – Brasil

{faoliveira,ronaldo.rgold,yoko}@ime.eb.br, quelopes@lncc.br

Abstract. *Nowadays, the Web of Data is a highly diverse and rich source of information. One of its great challenges lies in extracting useful information that leads to knowledge and advancement in the scientific area. The data mining algorithms help in the process of knowledge discovery, based on different search strategies. However, in general, they produce a considerable amount of rules, which are difficult to manipulate by the user. In this work, we present an adaptation of the MRAR algorithm based on the search mask concept, which is used to direct the mining process in order to find rules that can really be useful to the user in a shorter time.*

Resumo. *A Web de Dados é hoje uma fonte altamente diversa e rica de informações. Um dos seus grandes desafios está na extração de informações úteis que levem ao conhecimento e avanço na área científica. Os algoritmos de mineração de dados auxiliam nesse processo de descoberta do conhecimento, baseando-se em diferentes estratégias de busca. Porém, em geral são custosos e produzem um grande volume de regras, dificultando a manipulação pelo usuário. Neste trabalho, apresentamos uma adaptação do algoritmo MRAR baseada no conceito de máscara de busca, com o objetivo de direcionar o processo de mineração, reduzindo o custo e encontrando regras úteis para o usuário.*

1. Introdução

O avanço tecnológico, a popularização do uso da internet, a criação e utilização de novas tecnologias e ferramentas com os mais variados tipos de sensores para captação de dados do mundo real estão cada vez mais presentes no cotidiano das pessoas, empresas e instituições. O grande volume de informações que tais tecnologias geram inviabilizam a capacidade humana de realizar análises sob esses dados. Nesse contexto, a Web de dados surge para tratar e organizar esses conjuntos de informações, assim como o processo de extração do conhecimento KDD (do inglês, Knowledge Discovery in Databases). O processo de mineração de dados (Data Mining) corresponde a uma das etapas de KDD, e é comumente utilizado para extrair informações das bases de dados.

Algoritmos baseados em mineração de regras de associação são comumente utilizados em Data Mining e, em geral, acarretam em uma grande quantidade de regras geradas. Por exemplo, o algoritmo MRAR [Ramezani 2014], uma versão adaptada do algoritmo *Apriori* [Agrawal et al. 1993] (clássico para mineração de regras de associação),

explora regras de associação de multirrelação sobre grafos direcionados, como os que são usados para representação de dados na Web de Dados. Embora capaz de identificar todas as regras de associação de multirrelação existentes em um conjunto de dados, o MRAR, em geral, produz um volume considerável de regras que, mesmo ordenadas segundo critérios pré-definidos, são de difícil manipulação pelo usuário.

Sendo assim, este trabalho visou adaptar o MRAR para direcionar o processo de busca das regras com maior importância, tendo como objetivo principal gerar as regras baseadas nas informações sugeridas pelo próprio usuário. Com base nessas informações, o algoritmo analisa cada caminho possível no grafo e verifica se ali está presente o interesse do usuário, descartando o que for possível.

2. Mineração de Regras de Associação

Uma das principais tarefas na busca por novos conhecimentos em bases de dados é a chamada Mineração de Regras de Associação [Goldschmidt et al. 2015]. Em essência, essa tarefa consiste em identificar regras de associação frequentes e válidas em um conjunto de dados [Agrawal et al. 1993].

Uma regra de associação é uma implicação da forma $X \rightarrow Y$, onde X e Y são conjuntos de itens e $X \cap Y = \emptyset$. Um item é um predicado que pode assumir valor verdadeiro ou falso em função do registro de dados selecionado. Um exemplo de regra de associação pode ser $\{Sexo = M, JogaFutebol = S\} \rightarrow \{Saude = Boa\}$, onde $Sexo = M$, $JogaFutebol = S$ e $Saude = Boa$ são exemplos de itens.

Uma regra de associação $X \rightarrow Y$ é dita frequente (resp. válida) se, e somente se, $|X \cup Y|/|D| \geq$ *minsup* (resp.$|X \cup Y|/|X| \geq$ *minconf*). $|D|$ representa a quantidade total de registros de dados disponíveis no conjunto de dados D. Sendo I um conjunto de itens qualquer, $|I|$ representa a quantidade de registros do conjunto de dados que satisfazem simultaneamente a todos os itens pertencentes a I. *Minsup* (resp. *minconf*) é um parâmetro definido pelo usuário que estabelece uma frequência (resp. confiança) mínima para que a regra seja considerada frequente (resp. válida) no conjunto de dados.

3. Trabalhos Relacionados

Artigos sobre mineração de regras de associação em grafos são relativamente recentes e ainda pouco explorados. O artigo de [Hendrickx et al. 2015] apresenta estratégias de mineração de regras de associação entre os rótulos de nó no grafo, focando em informações adicionais sobre as correlações e interações entre esses nós. O algoritmo proposto pressupõe que, se um conjunto de rótulos é encontrado em um grafo, há uma alta probabilidade de que algum outro conjunto de rótulos possa ser encontrado nas suas proximidades. Embora o algoritmo leve em consideração os nós rotulados, ele não considera a diversidade de tipos de relacionamentos existentes em um grafo multirrelação.

O artigo de [Elseidy et al. 2014] apresenta o GRAMI (GRAph MIning) para a mineração subgrafos frequentes em um único grafo grande. A abordagem utilizada encontra os conjuntos mínimos de casos para satisfazer o nível de frequência e evita a enumeração custosa de todos os casos exigidos por outras abordagens. GRAMI faz a avaliação de modelos frequentes como um problema de satisfação de restrições (CSP – Constraint Satisfaction Problem). Em cada interação, GRAMI resolve o CSP até encontrar um conjunto mínimo que são suficientes para avaliar a frequência do subgrafo e

ignora os conjuntos restantes. Apesar de eficiente, o GRAMI não atende a grafos com nós e arestas multirrelação como os provenientes da Web de Dados.

Já [Ramezani 2014] propõe um algoritmo para minerar regras de associação em grafo, o MRAR (Mining Multi-Relation Association Rules). Esse algoritmo considera um grafo onde cada nó pode ter um ou mais tipos de relacionamentos (grafos multirrelações), como ocorre na Web de Dados. Além disso, os relacionamentos são direcionados, formando um grafo direcionado.

A mineração de dados em grafo pelo MRAR se vale dos mesmos princípios aplicados à mineração de dados tradicional. No caso do grafo direcionado, a ideia é encontrar caminhos frequentes que possam ocorrer, i.e., caminhos que percorram diferentes relações e cheguem em um mesmo nó. Tomando como ponto de chegada o nó 'Humid' da Figura 1, há um caminho que se repete, formando a seguinte composição das relações 'Live_In(Near_By(Climate_Type(Humid)))', a partir dos nós 'Hasan' e 'Reza'. Nesse mesmo exemplo, temos que esses mesmos nós também formam caminhos até o nó 'Good', 'Health_Condition(Good)'. Com base nesses caminhos frequentes, tem-se a geração da regra multirrelação, onde quem vive perto de uma cidade com o clima úmido implica ter a condição de saúde boa, com o suporte de 11% e a confiança de 69%. Os conceitos usados para a geração do MRAR estão formalizados mais adiante.

O trabalho de [Ramezani 2014] é o mais próximo do presente trabalho, porém, diferentemente, não tem o objetivo de facilitar encontrar as regras mais desejadas.

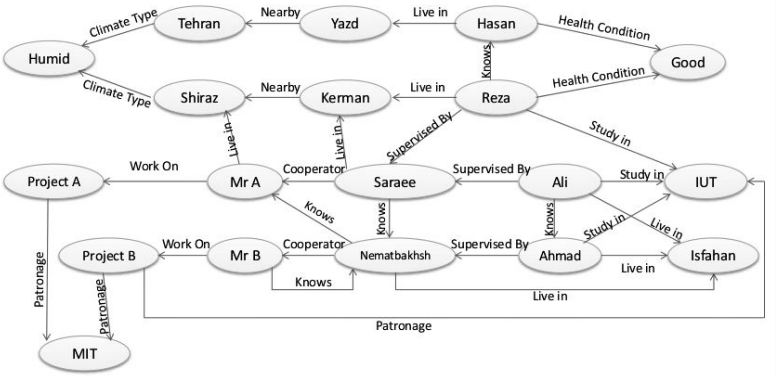


Figura 1. Grafo direcionado com rótulos nas arestas. Fonte: [Ramezani 2014].

4. Proposta

De forma geral, no processo de mineração de regras de associação muitas regras são encontradas e, nem sempre, as estratégias estão direcionadas para filtrar regras que são realmente úteis e válidas para um problema específico. Dependendo da configuração dos valores de suporte mínimo (*minsup*) e confiança mínima (*minconf*), o número de resultados (de regras) encontrados pode aumentar ou diminuir. Ao definir um valor alto, pode ocorrer que nenhuma regra seja gerada. Por outro lado, ao definir um valor baixo, um número muito grande de regras passa a fazer parte dos resultados, dificultando a análise dos resultados por parte do usuário. Assim, dependendo dos dados analisados, torna-se difícil encontrar um ponto de equilíbrio. Além disso, esse processo pode ser bem custoso em termos de tempo.

Neste trabalho é proposta uma nova estratégia para direcionar e minimizar o tempo do processo de mineração de regras de associação, visando encontrar regras mais viáveis e efetivas. Dessa forma, foi desenvolvido um ajuste no código do MRAR. Os conceitos para a mineração de regras de associação de multirrelação em um grafo, utilizados no MRAR, são similares aos usados na mineração tradicional feita utilizando o algoritmo *Apriori*.

No entanto, para que possamos expressar o ajuste realizado, algumas definições baseadas na funcionalidade do MRAR [Ramezani 2014], precisaram ser formalizadas, sendo essa também uma contribuição importante deste artigo.

Segundo a teoria dos grafos, um grafo dirigido $G = (V, A)$ é uma estrutura onde $V = \{v_1, v_2, ..., v_n\}$ é um conjunto de vértices e $A = \{(v_i, v_j)/v_i, v_j \in V\}$ é um conjunto de arestas. Dada uma aresta e qualquer de A, representada genericamente por (v_i, v_j), diz-se que e parte do vértice v_i e chega ao vértice v_j.

Definição 1: Dado um grafo $G = (V, A)$, define-se $G_r = (R, P, PI)$ como um grafo em termos dos elementos do RDF, onde:

- R é o conjunto de recursos em G_r: $R = \{r_j / \exists\, v_i \in V \wedge r_j = v_i\}$.
- P é o conjunto de propriedades/predicados sobre os recursos $r_i \in R$ que podem formar arestas em G_r: $P = \{p^1, p^2, ..., p^m\}$.
- PI é o conjunto de instâncias de propriedades/predicados que formam arestas direcionadas em G_r: $PI = \{p_y^x$ / $\exists\ p^x \in P \wedge r_i, r_j \in$ R $\wedge\ (r_i, r_j) \in A \wedge$ p_y^x = (r_i, r_j) $\}$. De outra forma, pode-se dizer que p_y^x = (r_i, r_j), ou ainda que $p_y^x(r_i) = r_j$.

Definição 2: Define-se $C(x, y)$ um caminho no grafo G_r, que leva um recurso x (chamado de origem do caminho) a um recurso y (chamado de destino do caminho), como um conjunto ordenado de instâncias de propriedades $p \in PI$ aplicadas sobre recursos de R, da seguinte forma: $C(x, y) = (p_1, ..., p_k)$, onde $p_1, ..., p_k \in PI$, e $p_1 = (x, r_j), p_2 = (r_j, r_{j+1}), ..., p_k = (r_{j+k-1}, y)$, e $x, y, r_j, ...r_{j+k-1} \in R$. O caminho $C(x, y)$ também pode ser expresso como: $C(x, y)$ = $x \xrightarrow{p_1} r_j \xrightarrow{p_2} r_{j+1} \xrightarrow{p_3} ... r_{j+k-1} \xrightarrow{p_k} y$.

No grafo da Figura 1, um caminho entre os nós *Reza* (origem) e *Humid* (destino) pode ser expresso da seguinte forma: $C(Reza, Humid)$ = Reza $\xrightarrow{Live_in}$ Kerman $\xrightarrow{Near_by}$ Shiraz $\xrightarrow{Climate_Type}$ Humid.

Definição 3: Define-se uma *cadeia* $\mathcal{C}_{y,(p^1,...,p^k)}$ no grafo G_r como uma coleção de caminhos $C(r_j, y)$, da seguinte forma: $\mathcal{C}_{y,(p^1,...,p^k)} = \{C(r_j, y)/C(r_j, y) = (p_{i_1}^1, ..., p_{i_k}^k)\}$

No exemplo da Figura 1, temos que os caminhos Reza $\xrightarrow{Live_in}$ Kerman $\xrightarrow{Near_by}$ Shiraz $\xrightarrow{Climate_Type}$ Humid, e Hasan $\xrightarrow{Live_in}$ Yazd $\xrightarrow{Near_by}$ Tehran $\xrightarrow{Climate_Type}$ Humid, são elementos da cadeia $\mathcal{C}_{Humid,(Live_in,Near_by,Climate_type)}$.

Salvo casos onde haja ambiguidade, optaremos por simplificar a notação de cadeia $\mathcal{C}_{y,(p^1,...,p^k)}$ pela seguinte representação: $\mathcal{C}_{y,s}$, onde $s = (p^1, ..., p^k)$.

Definição 4: Dada uma cadeia $\mathcal{C}_{y,s}$, define-se $\mathcal{I}(C_{y,s})$ como a coleção formada por recursos que são origens em caminhos que pertençam a $\mathcal{C}_{y,s}$. Em termos formais:

$$\mathcal{I}(C_{y,s}) = \{x/C(x, y) \in C_{y,s}\}$$

No exemplo da Figura 1, temos que $\mathcal{I}(C_{Humid,(Live_in,Near_by,Climate_type)})$=$\{Reza, Hasan\}$.

Definição 5: Define-se como uma regra de associação de multirrelação entre duas cadeias $\mathcal{C}_{y,s}$ e $\mathcal{C}_{z,r}$, a implicação $\mathcal{C}_{y,s} \rightarrow \mathcal{C}_{z,r}$, onde $\mathcal{C}_{y,s} \cap \mathcal{C}_{z,r} = \emptyset$.

Novamente considerando o exemplo da Figura 1, temos que $Live_In(Near_By(Climate_Type(Humid))) \rightarrow Health_Condition(Good)$ é um exemplo de regra de associação de multirrelação.

Definição 6: Seja $\mathcal{C}_{y,s} \rightarrow \mathcal{C}_{z,r}$ uma regra de associação de multirrelação entre duas cadeias $\mathcal{C}_{y,s}$ e $\mathcal{C}_{z,r}$. Diz-se que $\mathcal{C}_{y,s} \rightarrow \mathcal{C}_{z,r}$ é frequente (resp. válida) se, e somente se, $|\mathcal{I}(C_{y,s}) \cap \mathcal{I}(C_{z,r})|/|R| \geq$ *minsup* (resp. $|\mathcal{I}(C_{y,s}) \cap \mathcal{I}(C_{z,r})|/|\mathcal{I}(C_{y,s})| \geq$ *minconf*).

Para evitar percorrer o grafo em busca de todas as regras de associação de multirrelação frequentes e válidas existentes, e reduzir o custo, propõe-se o algoritmo $MRAR_m$ que utiliza o conceito de máscara de busca [Goldschmidt et al. 2015]. Uma máscara de busca é uma implicação que estabelece dois conjuntos de recursos a serem utilizados pelo algoritmo como restrições em regras de associação de multirrelação. O primeiro (resp. segundo) é o conjunto de recursos a considerar como destinos na cadeia do antecedente (resp. consequente) das regras identificadas. A seguir, encontra-se a definição formal de máscara de busca.

Definição 7: Define-se uma máscara de busca $\mathcal{M}$ como uma implicação da forma $Y \rightarrow Z$, onde $Y, Z \subseteq R$ e $Y, Z \neq \emptyset$.

Ao aplicar uma máscara $\mathcal{M} : Y \rightarrow Z$ ao processo de busca do algoritmo $MRAR_m$, apenas regras de associação de multirrelação da forma $\mathcal{C}_{y,s} \rightarrow \mathcal{C}_{z,r}$, onde $y \in Y$ e $z \in Z$, serão consideradas, reduzindo o espaço de busca conforme o interesse do usuário. No grafo da Figura 1, $\{Humid\} \rightarrow \{Good\}$ é um exemplo de máscara de busca, que fará o algoritmo $MRAR_m$ considerar apenas regras de associação de multirrelação que atendam às seguintes condições, simultaneamente: (a) regras cujo antecedente contenha uma cadeia cujos caminhos tenham como destino o recurso *Humid*; (b) regras cujo consequente contenha uma cadeia cujos caminhos tenham como destino o recurso *Good*.

Cabe ressaltar ainda que, caso o usuário especifique uma máscara onde $Y = R$ e $Z = R$, o algoritmo MRAR será executado normalmente, buscando todas as regras de associação de multirrelação, sem imposição de restrições quanto ao conjunto de recursos a ser considerado. De forma análoga, ao usar $Y = R$ (resp. $Z = R$), o algoritmo realiza o processo de busca por quaisquer cadeias que possam ocorrer no antecedente (resp. consequente) da regra.

5. Implementação e estudo de caso

O algoritmo $MRAR_m$ foi implementado em PHP e Javascript. O protótipo gerado foi utilizado para um estudo de caso, considerando o grafo que representa uma rede de relacionamentos entre professores, alunos, instituições e cidades, construído a partir do exemplo visto na Figura 1 (19 nós e 26 arestas). Ao aplicar o $MRAR$, com os valores de corte (*cut-off*) para o suporte mínimo (*minsup* $>= 10\%$) e para confiança (*minconf* $>= 70\%$), os mesmos valores utilizados em [Ramezani 2014], o resultado inicial obtido foi de um conjunto composto por 470 variações de regras.

Assim, para direcionar os resultados de busca, utilizou-se o algoritmo $MRAR_m$ com as mesmas configurações anteriores para *minsup* e *minconf*. Ao aplicar a máscara de busca $\mathcal{M} : R \longrightarrow \{Humid, Good\}$, que restringe a busca por regras que envolvam

os recursos $Humid$ e $Good$ no consequente, ocorreu que apenas 5 regras foram geradas, como mostra a Figura 2. Como $\mathcal{M}$ não faz nenhuma restrição quanto antecedente das regras (R é o conjunto completo de recursos), todos os recursos são considerados na identificação das cadeias no lado esquerdo das regras buscadas.

Em testes preliminares com o $MRAR$ original, foi possível observar uma redução de tempo significativa, alterando o suporte para reduzir o número de regras. Esperamos uma redução ainda maior com o $MRAR_m$. Testes estão em andamento com datasets maiores.

Formatted Rules

	Humid,Good	Sup.	Conf
Live_In(Near_By(Climate_Type(Humid))) ➔	Health_Condition(Good)	0.11	0.69
Live_In(Kerman) ➔	Live_In(Near_By(Climate_Type(Humid)))	0.11	1.00
Live_In(Near_By(Shiraz)), Live_In(Kerman) ➔	Live_In(Near_By(Climate_Type(Humid)))	0.11	1.00
Health_Condition(Good) ➔	Live_In(Near_By(Climate_Type(Humid)))	0.11	1.00
Live_In(Near_By(Shiraz)) ➔	Live_In(Near_By(Climate_Type(Humid)))	0.11	1.00
Antecedent	**Consequent**	**Sup**	**Conf**

Figura 2. $MRAR_m$ usando a máscara $\mathcal{M}$ sobre os consequentes das regras.

6. Conclusão

Este trabalho apresentou as definições em que se baseia o algoritmo $MRAR_m$, uma adaptação do MRAR para direcionar o processo de busca por regras de associação de multirrelação que sejam de maior interesse para o usuário. O objetivo é facilitar a manipulação do resultado numeroso do algoritmo, bem como reduzir seu custo de processamento. Além da formalização e implementação de um protótipo, o trabalho apresenta também um estudo de caso, mostrando o potencial desta abordagem. Como trabalhos futuros, destacam-se a possibilidade de expressar máscaras abrangendo predicados, o "enriquecimento" de datasets interligados na Web de Dados, além de um estudo detalhado do impacto do uso das máscaras na redução do tempo de processamento do $MRAR_m$.

Referências

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216.

Elseidy, M., Abdelhamid, E., and Skiadopoulos, S. (2014). GRAMI: Frequent Subgraph and Pattern Mining in a Single Large Graph. *Proceedings of the VLDB Endowment*, 7(7):517–528.

Goldschmidt, R., Bezerra, E., and Passos, E. (2015). *Data mining: conceitos, técnicas, algoritmos, orientações e aplicações*. Rio de Janeiro: Elsevier.

Hendrickx, T., Cule, B., Meysman, P., Naulaerts, S., Laukens, K., and Goethals, B. (2015). *Mining Association Rules in Graphs Based on Frequent Cohesive Itemsets*, pages 637–648. Springer International Publishing, Cham.

Ramezani, R. (2014). MRAR : Mining Multi-Relation Association Rules. *Journal of Computing and Security*, 1(2):133–158.

paper:174648

Online Performance Modeling for NoSQL Databases using Extreme Learning Machines

Victor A. E. Farias [1], Pedro R. A. Pinheiro [1], Flávio R. C. Sousa [1], João P. P. Gomes [1], Javam C. Machado [1]

[1]Computer Science Department
Federal University of Ceara
Fortaleza, Brazil

{victor.farias, pedro.ramyres, flavio.sousa, joao.pordeus, javam.machado}
@lsbd.ufc.br

Abstract. *NoSQL databases rise as a solution to manage large amounts of data in the cloud. Mechanisms to guarantee Quality of Service in can significantly benefit from performance predictability. Building an accurate predictive model to estimate a DBMS performance in a cloud environment is challenging since i) workload and resources allocation change dynamically; ii) concurrency and distribution introduce nonlinearity on performance metrics and iii) predictive models should be trained and updated online to capture unseen workloads. This paper presents an online performance modeling approach for NoSQL databases using extreme learning machines. Experimental results confirm that our performance modeling can accurately predict throughput under several scenarios.*

1. Introduction

Cloud computing is a paradigm of remarkable success for service-oriented computing. Scalability, elasticity, pay-per-use pricing, and economy of scale are the major reasons for this achievement. Since most of cloud applications are data-driven, database management systems powering these applications are critical components in the cloud software stack [Elmore et al. 2011].

Cloud NoSQL database systems are leading to data processing environments that concurrently execute heterogeneous query workloads. At the same time, these systems need to satisfy diverse performance expectations defined in Service Level Agreements (SLAs). In these newly emerging settings, avoiding potential SLA violations rely on performance predictability, i.e., the ability to estimate the impact of concurrent query execution on the performance of the system in a continuously evolving workload [Duggan et al. 2011].

This work discuss a data-driven approach which applies machine learning to construct accurate performance models by generalizing training data. Performance models are able to predict system's performance metrics based on a given workload and a resource allocation configuration. Thus, monitoring system collects workload and systems configuration metadata from production environment to feed machine learning techniques.

Workload and resources allocation change dynamically, thus predictive models should update to capture the behavior of newly unseen workloads and system's settings

to improve accuracy. Retraining a new model on every update is impracticable since the number of workloads configurations may grow indefinitely which entails in a high computational cost. Thereby, in this scenario, a predictive model should update online in real-time with only most recent workloads while preserving accuracy.

Many NoSQL databases employ replication to improve reliability and performance, but it implies in an additional cost of maintaining consistency. Traditionally, consistency mechanisms are based on distributed locks which add waiting periods on queries execution. Authors in [Gray et al. 1996] suggests that the number of distributed locks grows quadratically, in the worst case, with the number of resource nodes and to the rate of transactions. Thus, the concurrent interaction of distributed queries execution is theses systems can degrade the performance in a non-linear fashion[Gray et al. 1996], which is a challenge for performance modeling approaches.

Modeling the performance impact of complex interactions of concurrent execution of a heterogeneous workload share computing resources and data is difficult albeit critical. Particularly, it supports many mechanisms used to guarantee Quality of Service (QoS) manning of cloud-based database platforms as efficient resource allocation and admission control. Performance prediction has been widely studied in the literature. This problem is also tackled by analytical approaches which analyzes each single factor that impacts the performance on a specific database system. However, these approaches can become very complex and system-specific.

Learning-based offline approaches were proposed to predict isolated queries performance in single node DBMSs [Ganapathi et al. 2009] and to predict concurrent workload performance for single node DBMSs [Duggan et al. 2011, Mozafari et al. 2013] and for distributed DBMSs [Farias et al. 2016b, Didona and Romano 2014, Farias et al. 2016a]. However, these approaches require a costly retraining when faced to new workloads and queries. An online approach is explored in [Sheikh et al. 2011] where the authors leverage gaussian models for predicting only single queries' response time under a concurrent workload in a single node DBMS. Our

This paper presents a machine learning approach for online performance modeling of NoSQL databases. The major contributions of this paper are as follows:

- We present a performance modeling approach for NoSQL database systems exploring machine learning techniques to predict system's performance metrics considering nonlinear effects of concurrency and distribution;
- We propose a strategy for online update in real-time of the predictive models;
- We present a full prototype implementation of the proposed approach and we evaluate models constructed by our prototype to assess their predictive accuracy.

Organization: This paper is organized as follows. Section 2 explains our approach. Section 3 describes the experimental evaluation. Conclusions and future research steps are shown in Section 4.

2. Our Approach

Our approach aims to construct, in a online manner, an accurate performance model M which predicts system's performance metrics based on a given workload configuration and on a resource allocation. The workload is composed by a set of request templates

$T_1, T_2, \ldots, T_n$. A workload configuration is the rate (request per second) $R_1, R_2, , R_n$ of each template, respectively.

NoSQL systems are designed to run in a distributed manner by taking advantage of a large resource pool. The data partitioning and management are database-specific. Thus, resource allocation is described by one parameter: the number of working nodes in the NoSQL cluster DB. This parameter is generic to any kind of NoSQL database system and it represents the capacity of the system.

The performance metric addressed to assess the accuracy of M is queries' mean response time (MRT) which is intuitive from the point of view of the database manager. We observe that our work is generic to other performance metric as throughput since the discussion in the remainder of this paper is not tied to MRT. So, formally, M is defined

$$M(DB, R_1, R_2, \ldots, R_n) \mapsto MRT$$

This setup of input variables allows us to capture several factors that can affect the performance of a distributed NoSQL database system. Particurlarly, three aspects: i) **Requests complexity**. Each single workload has a different access pattern and requires a different amount of disk operations. Each R_i variable independently accounts the complexity of template T_i; ii) **Concurrency**. Concurrent execution of two distinct workload classes may generate interference caused by lock resource sharing [Mozafari et al. 2013]; and iii) **Replication**. Most NoSQL databases distribution strategies employ replication to improve availability and reliability. However, this implies on an overhead to maintain consistency. Some consistency strategies use distributed locks leading to waits and deadlocks, and, consequently, degrading the performance in a non-linear fashion [Gray et al. 1996]. For factor ii and iii, We employ machine learning models capable of capturing interactions between inputs variables.

Our approach relies on machine learning algorithms and we show, by experimentation, that it can capture the aforementioned aspects with an acceptable error. In particular, since the performance metrics are continuous and non-finite, we address the performance modeling problem as a regression problem. Regression methods are capable of producing predictive models for estimating a system's performance given a set of training data under several scenarios of workload configuration and resource allocation.

2.1. Online Learning

Traditional regression methods require a training data set *a priori* to construct predictive models. In a cloud scenario, workload configuration and resource allocation change frequently. Machine leaning models tends to be more inaccurate when making predictions of a point $(DB, R_1, R_2, \ldots, R_n)$ distant from the samples on the training data set. Thus, a online continuous learning is necessary to capture newly unseen workload configuration and resource allocation to preserve accuracy.

Our approach is based on a feedforward neural network model with randomly assigned hidden weights as proposed in [Schmidt et al. 1992]. More recently, such model has gained much attention and is often referred as Extreme Learning Machine (ELM, [Huang et al. 2004]). This network is a standard feed forward single hidden layer network where the input layer is fully connected to hidden layer and the output layer is composed

by a single unit connected to all units in the hidden layer. The output of the network is given by:

$$O(x) = \sum_{1}^{n} w_i F(M_i^T X)$$

Each unit in the hidden layer of size n calculates a weighted ($M_i = [m_1, m_2, \ldots, m_n]$) sum of the inputs and applies a activation function F (sigmoid). Particularly, all M_i are uniform random values in $[-1, 1]$. Finally, the output of the network is the weighted ($W = [w_1, w_2, \ldots, w_n]$) sum of the output of the hidden layer. Weights W are adjusted to fit the data according to L_2 criterion.

In this work, we apply an online sequential approach for computing W proposed in [Liang et al. 2006]. Data arrives sequentially (one-by-one or chunk-by-chunk of varying size), it is learned and discarded at the end of the learning process. The algorithm learns in real time and does not have any adjustable hyper-parameter for tuning requiring least user intervention.

3. Experimental Evaluation

Our experiments were designed considering the goal of assessing learning effectiveness by measuring the accuracy in the learning online. We quantify accuracy by mean percentage error (MPE) metric so that the smaller the error, the more accurate is the model.

3.1. Experimental setup

We implemented a prototype of our approach using Python and Bash Script languages. We implemented ELM algorithm in python on top of Numpy library [Walt et al. 2011]. The model is update online in batches of 120 samples (120 seconds).

To generate the workload, we use the *de facto* NoSQL benchmark YCSB [Cooper et al. 2010]. This benchmark has four types of requests templates: point and range query, update and insert where we can simulate workloads by changing the rate of each template independently. In our experiments we fixed the following settings on the benchmark: 40 client threads, 1000000 records on database, 1-byte size record, uniform access pattern.

The test database in our experiments is MongoDB [Inc 2015] set on Master/Slave replication mode. The *ReadPreference* parameter is set to *Nearest* which means that MongoDB driver sends read for a randomly chosen node while writes are always issued to the master. This prototype was deployed in a Openstack private cloud environment. Each virtual machine holds 4 vCPUs and 4 GB main memory.

3.2. Experimental Results

We evaluate our performance model under a scenario that the workload configuration and the resource allocation vary over time. We explore two request templates of YCSB: point queries (T_1) and inserts (T_2). Thus, the experimental scenario is defined as the following ordered tuple representing (DB, R_1, R_2): $(7, 6000, 0)$,$(7, 4200, 1800)$,$(7, 5400, 600)$,$(7, 4800, 1200)$,$(7, 6000, 0)$,$(3, 6000, 0)$,$(5, 6000, 0)$ and $(6, 6000, 0)$. Each tuple is executed for 120 seconds.

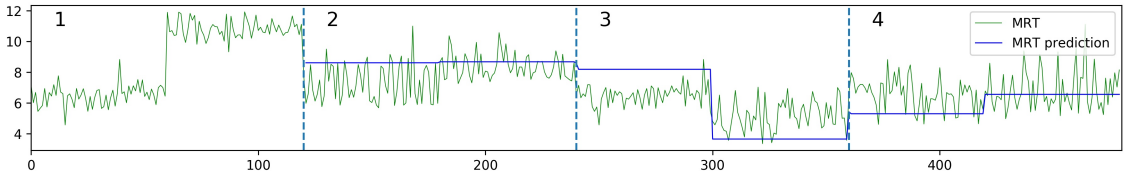


Figure 1. Experimental scenario demonstrating the online learning of our approach

Figure 1 plots the MRT for the experiment (in green) against the predicted MRT from the network. We divided the experiment in 4 windows separated by a vertical dashed line. In each windows, the monitoring data of previous windows were given for the network learning algorithm.

In the first 2 windows, the system the workload configuration changes and, in the last 2 windows, the resource allocation changes. We observe that in windows 2 the network could generalize the the behavior from windows 1 and make accurate predictions ($MPE = 16.5\%$). From second 300 to 360, the system had the first change of resource allocation so it obtained a inaccurate estimate ($MPE = 29.2\%$) since it has no information about resource allocation so far. However, in window 4, the network has learned from window 3 and made better estimates for resource allocation changes ($MPE = 17.1\%$). The MPE obtained for the whole experiment is 20.9%.

4. Conclusion

This paper presents a novel approach for online performance modeling for NoSQL databases. This approach can create performance models with production monitoring data. We conducted experiments to show that such model is accurate and can learn online to preserve accuracy. Experimental results confirm that this model predicts with 20.9% of error for a new workload configuration and resource allocation showing that this approach can generalize from the past experience. This work opens discussion for new research opportunities. There is a number of research opportunities that derive from this work, including: guided online learning where we can adjust system's knobs (as DB parameter) to obtain a better data set to improve model's accuracy; and concept drift handling, e.g., the cloud manager substitute the hard disk drive for a faster device as solid state disk, thus the performance changes abruptly making the trained models inaccurate.

Acknowledgment

This Research was partially supported by LSBD/UFC.

References

Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154. ACM.

Didona, D. and Romano, P. (2014). On bootstrapping machine learning performance predictors via analytical models. *arXiv preprint arXiv:1410.5102*.

Duggan, J., Cetintemel, U., Papaemmanouil, O., and Upfal, E. (2011). Performance prediction for concurrent database workloads. In *ACM SIGMOD*, pages 337–348. ACM.

Elmore, A. J., Das, S., Agrawal, D., and El Abbadi, A. (2011). Zephyr: live migration in shared nothing databases for elastic cloud platforms. In *SIGMOD '11*, pages 301–312.

Farias, V. A. E., Sousa, F. R. C., Maia, J. G. R., Gomes, J. a. P. P., and Machado, J. C. (2016a). Elastic provisioning for cloud databases with uncertainty management. In *ACM SAC*, pages 390–397.

Farias, V. A. E., Sousa, F. R. C., Maia, J. G. R., Gomes, J. P. P., and Machado, J. C. (2016b). Machine learning approach for cloud nosql databases performance modeling. In *CCGrid*, pages 617–620.

Ganapathi, A., Kuno, H., Dayal, U., Wiener, J. L., Fox, A., Jordan, M., and Patterson, D. (2009). Predicting multiple metrics for queries: Better decisions enabled by machine learning. In *ICDE*, pages 592–603. IEEE.

Gray, J., Helland, P., O'Neil, P., and Shasha, D. (1996). The dangers of replication and a solution. In *ACM SIGMOD Record*, volume 25, pages 173–182. ACM.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE.

Inc, M. (2015). *MongoDB*. `http://www.mongodb.com`.

Liang, N.-Y., Huang, G.-B., Saratchandran, P., and Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423.

Mozafari, B., Curino, C., Jindal, A., and Madden, S. (2013). Performance and resource modeling in highly-concurrent oltp workloads. In *ACM SIGMOD*, pages 301–312. ACM.

Schmidt, W. F., Kraaijveld, M. A., and Duin, R. P. (1992). Feedforward neural networks with random weights. In *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, pages 1–4. IEEE.

Sheikh, M. B., Minhas, U. F., Khan, O. Z., Aboulnaga, A., Poupart, P., and Taylor, D. J. (2011). A bayesian approach to online performance modeling for database appliances using gaussian models. In *Proceedings of the 8th ACM international conference on computing*, pages 121–130. ACM.

Walt, S. v. d., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

paper:166519

Redes Sociais Científicas: análise topológica da influência dos pesquisadores

Vitor Horta[1], **Victor Ströele**[1], **Fernanda Campos**[1], **José Maria N. David**[1], **Regina Braga**[1]

[1]Programa de Pós-graduação em Ciência da Computação
Universidade Federal Juiz de Fora (UFJF) – MG – Brazil

{vitor.horta, victor.stroele}@ice.ufjf.br,

{fernanda.campos,jose.david, regina.braga}@iufjf.edu.br

Abstract. *Communities in social networks are composed by people with common interests who influence or are influenced by themselves. In this work, complex network analysis concepts are applied to verify the influence level among researchers, analyzing the structure of the scientific social network and its communities. We propose a bidirectional graph-based model to analyze the influence between researchers, and two algorithms to analyze the network structure, to identify scientific communities and to locate multidisciplinary researchers. To evaluate the model and the algorithms, a large scientific database is used in a use case. The results point to the solution viability and effectiveness.*

Resumo. *Comunidades em redes sociais são compostas por pessoas com interesses comuns, que influenciam ou são influenciadas por elas mesmas. Neste trabalho são aplicados conceitos de análise de redes complexas para verificar o nível de influência entre os pesquisadores, analisando a estrutura da rede social científica e suas comunidades. São propostos um modelo baseado em grafo bidirecional para analisar a influência entre os pesquisadores e algoritmos para analisar a estrutura da rede, identificar comunidades científicas e localizar pesquisadores multidisciplinares. Para avaliação do modelo e dos algoritmos é utilizada uma base de dados científicos de grande porte em um caso de uso. Os resultados apontam para a viabilidade e eficácia da solução.*

1. Introdução

Uma rede complexa representa um conjunto de objetos que se conectam de maneira não-trivial. Para se entender como os objetos se relacionam e se agrupam é necessária uma análise cuidadosa sobre a estrutura e características dessas redes [Wasserman and Faust 1994].

Neste trabalho serão utilizados os conceitos e técnicas de redes complexas para analisar como ocorrem as interações entre pesquisadores em uma rede social científica, com o objetivo de identificar (i) pesquisadores influenciadores, (ii) comunidades científicas e (iii) pesquisadores que participam de mais de uma comunidade, ou seja, pesquisadores que influenciam ou são influenciados por diferentes grupos de pesquisadores. Este trabalho avança as pesquisas do NEnC (Grupo de Pesquisa em Engenharia de Conhecimento) em análise de redes sociais científicas [Ströele et al. 2013] [Almeida et al. 2016].

A rede social científica modelada neste trabalho é representada por um grafo bidirecional cujos vértices representam os pesquisadores e as arestas correspondem às relações científicas entre eles. Um grafo direcionado é utilizado para que seja possível definir o grau de influência entre os pesquisadores, permitindo a identificação de pesquisadores influenciadores e influenciados em suas áreas de pesquisa.

A principal contribuição deste trabalho é a proposta e desenvolvimento de um algoritmo de agrupamento que considera as características de pesquisas multidisciplinares de alguns pesquisadores, permitindo que eles pertençam a mais de um grupo. Com essa abordagem é possível identificar pessoas que possuem mais de uma área de atuação, e que participam de duas ou mais comunidades científicas. Outra característica importante do algoritmo proposto é o sub-agrupamento, cujo objetivo é identificar subgrupos de comunidades de pesquisa.

Além da Introdução, este trabalho está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados, a Seção 3 descreve o modelo proposto para a realização deste estudo, na Seção 4 é feita a análise da topologia da rede social científica, na Seção 5 é descrito o algoritmo de agrupamento desenvolvido, um estudo de caso é realizado na Seção 6 e, finalmente, na Seção 7 são apresentadas as considerações finais.

2. Trabalhos Relacionados

[Luo et al. 2011] utiliza métodos hierárquicos divisivos para detecção de comunidades. Tais métodos consistem, essencialmente, em retirar as arestas com maior índice de centralidade (*edge betweenness*), e identificam comunidades em um dendograma. Essa abordagem é frequentemente utilizada, porém a alta complexidade computacional do método não favorece o seu uso em redes sociais de grande porte.

O DENGRAPH [Falkowski et al. 2007] foi desenvolvido utilizando a estratégia de agrupamento por densidade através de uma implementação do DBSCAN [Ester et al. 1996] adaptada para grafos não-direcionados. Como resultado alcança um melhor desempenho em performance e maior capacidade de detectar ruídos em redes com maior volume de dados.

Como o DBSCAN em sua implementação original não prevê que o conjunto de dados pode possuir múltiplas granularidades na densidade, [Gialampoukidis et al. 2016] desenvolveram o DBSCAN*-Martingale. Este algoritmo adapta os parâmetros de densidade de forma iterativa com a finalidade de descobrir grupos com diferentes níveis de similaridade de seus respectivos membros.

[Li et al. 2016] observaram a necessidade de se considerar a direção nas relações dos membros em redes sociais e propuseram um método que utiliza a estratégia de agrupamento por densidade para detecção de comunidades em grafos direcionados. Entretanto, não consideram a possibilidade de um elemento pertencer a dois ou mais grupos distintos.

Como diferencial deste trabalho pode-se destacar o desenvolvimento de um algoritmo de agrupamento e subagrupamento, visto que o processo iterativo proposto por outros trabalhos é muito custoso tornando-se inviável para grandes bases de dados. Além disso, o algoritmo proposto considera diferentes níveis de influência entre pares de pesquisadores (grafo bidirecional), e fornece também um parâmetro que pode ser ajustado para aprimorar a decisão de se incluir membros com menor influência ou que já estejam

alocados em outras comunidades, localizando pesquisadores multidisciplinares.

3. Modelagem da Rede Social Científica

No modelo da rede científica proposto neste trabalho os nós do grafo representam os pesquisadores e as arestas possuem pesos que representam o nível de influência entre eles. Neste modelo, a influência que um pesquisador exerce sobre o outro não necessariamente é igual à influência que ele recebe desse mesmo pesquisador. Por isso, o grafo social que representa esse modelo é um grafo bidirecional no qual os pesos (ida e volta) da relação entre dois pesquisadores são analisados de forma diferenciada. Dessa forma dois vértices P_A e P_B estão conectados sempre por duas arestas direcionadas, que por sua vez são definidas através da relação de coautoria [Lopes et al. 2010].

$$IP_{AB} = \frac{\parallel P_A \bigcap P_B \parallel}{\parallel P_A \parallel} \tag{1}$$

4. Análise Topológica

Para analisar a estrutura da DBLP, composta por 1.306.546 vértices e 9.915.146 arestas, foi calculada a sua distribuição de graus (equação (2)), propriedade que caracteriza a topologia de uma rede complexa, e é obtida calculando-se quantos nós possuem determinado grau para todos os valores de graus existentes na rede.

$$f(k) = \frac{\textit{Número de vértices com grau } k}{\textit{Número total de vértices}} \tag{2}$$

Os gráficos da Figura 1 apresentam os resultados obtidos, onde é possível observar que há um grande número de pesquisadores com poucas relações (grau baixo) e poucos pesquisadores com muitas relações (grau alto). Esta característica é típica das redes livres de escala, nas quais poucos elementos centralizam a maioria das relações da rede, obedecendo uma lei de potência.

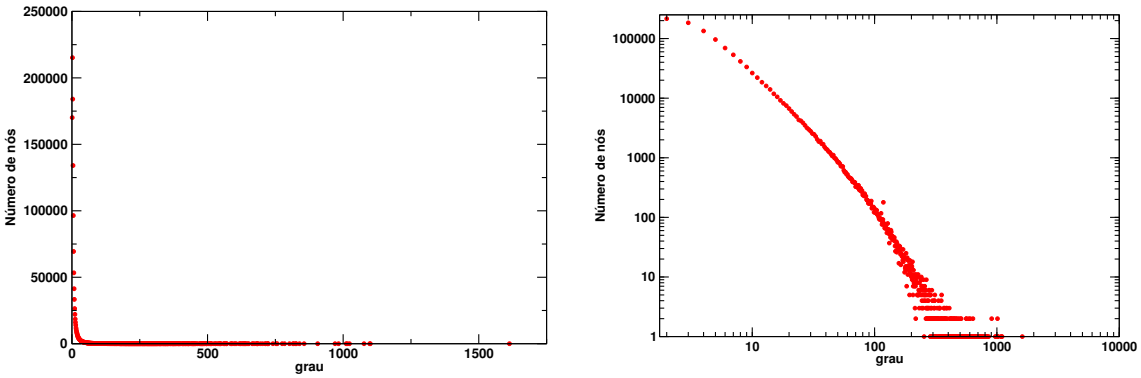


Figura 1. Distribuição de grau (esquerda) e gráfico log-log (direita)

Na análise de distribuição de grau foi detectado que existem poucos nós com grau elevado. Por outro lado, na análise da distribuição das influências entre os pesquisadores, observou-se que uma grande quantidade de nós possui a influência máxima. Para que essas duas análises sejam verdadeiras de forma simultânea, é necessário que a rede da DBLP

não seja totalmente conectada, sendo formada por componentes conexas independentes que contêm poucos nós de muita influência.

Para verificar o comportamento descrito anteriormente, foi elaborado um algoritmo cuja função é identificar as componentes conexas da rede. Em vista do grande volume de dados da rede, o algoritmo disponibiliza um parâmetro *raio* que pode ser utilizado para viabilizar sua execução, impactando a performance do algoritmo sem influenciar o resultado final. Um valor elevado para o *raio* irá diminuir o tempo de execução do algoritmo, porém, o consumo de memória aumenta consideravelmente. Isso acontece porque, com o aumento do *raio*, aumenta-se também o número de vértices armazenados em memória, o que contribui para acelerar o processamento. Assim, este algoritmo permite que mesmo computadores com baixa capacidade de processamento encontrem todas as componentes conexas de uma rede social de grande porte.

5. NETSCAN: algoritmo de agrupamento

Após as análises da topologia da rede social científica DBLP, pode-se definí-la como sendo uma rede complexa livre de escala, que possui baixa conectividade global e alguns nós centralizadores, ou seja, alguns nós altamente conectados. Neste contexto surge o interesse em identificar quais são os grupos de pesquisa mais bem definidos, quais são os membros de cada grupo, bem como os pesquisadores com maior influência sobre seus respectivos grupos. Para responder essas questões, é proposto e desenvolvido um algoritmo de agrupamento por densidade, chamado NETSCAN, baseado no algoritmo DBSCAN [Ester et al. 1996] e que possa ser aplicado em redes de grande porte como a DBLP.

Uma das principais diferenças com relação ao DBSCAN é que o NETSCAN considera que a distância entre dois vértices depende da direção do relacionamento que está sendo analisada. Além disso, permite que um mesmo vértice seja incluído em mais de um grupo e, dessa forma, é possível identificar pesquisadores que contribuem em diferentes áreas e grupos de pesquisa distintos, mesmo que esses pesquisadores não sejam caracterizados como centralizadores.

Além dos parâmetros ε e *minPts*, originados do DBSCAN, o NETSCAN permite a definição de um *raio*, utilizado para buscar por elementos influenciados por um vértice em uma profundidade maior, ou seja, pode ser definido quantas camadas de vizinhos a partir de um *core* serão analisadas.

6. Estudo de Caso

Neste estudo de caso, o algoritmo NETSCAN foi executado na maior componente conexa da rede social científica DBLP composta por 1.100.504 pesquisadores e 9.598.808 relacionamentos. Ao todo foram identificadas 34.776 comunidades de pesquisa, utilizando os parâmetros: $\varepsilon = 1$, $minPts = 5$ e $raio = 1$. Esses parâmetros foram definidos após alguns experimentos realizados com uma parte da base de dados.

Dado o grande volume de dados utilizados neste estudo, alguns agrupamentos foram selecionados com o intuito de ilustrar o comportamento do algoritmo proposto e analisar os resultados obtidos. A Figura 2 mostra um dos grupos definidos pelo NETSCAN onde os nós centralizadores estão representados pelos vértices verdes e os *border points* pela cor cinza. Para uma melhor visualização do grafo, as medidas de influência representadas nas arestas foram multiplicadas por 1000.

Observa-se que este grupo possui dois vértices influenciadores e que apenas dois *border points* estão ligados nestes dois vértices simultaneamente. Isso indica que este grupo possui uma grande chance de ser separado em dois subgrupos caso o parâmetro ε seja escolhido de forma mais rígida, exigindo um maior nível de influência no grupo.

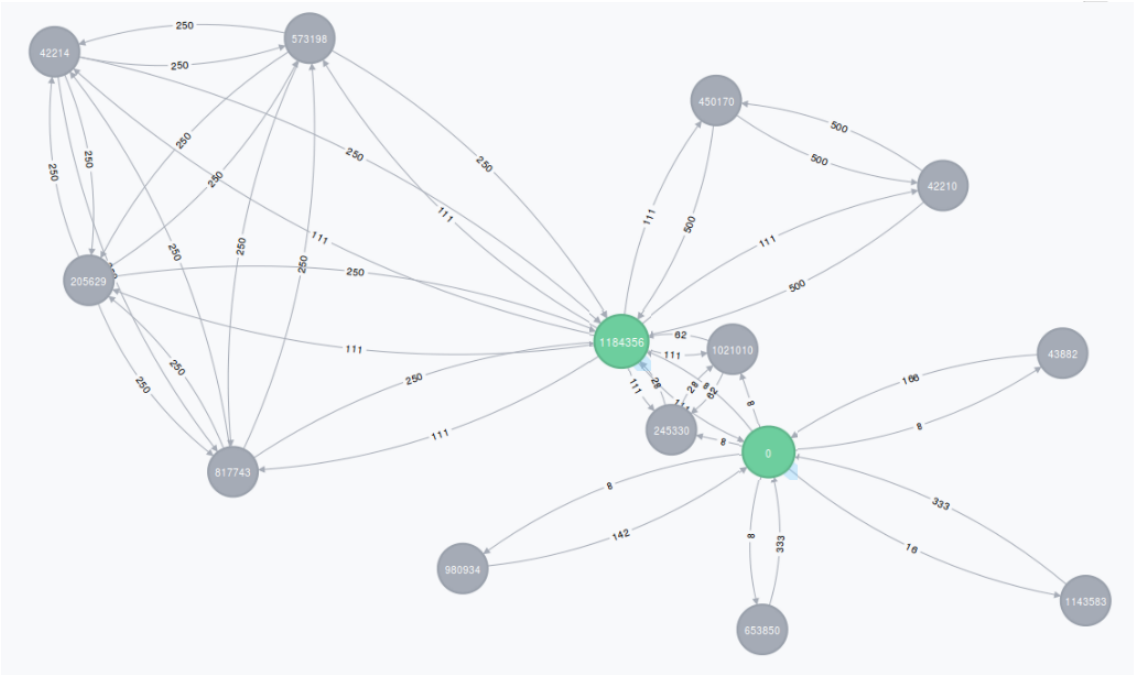


Figura 2. Grupo de pesquisadores identificado pelo NETSCAN

6.1. Análise dos Resultados

Através do NETSCAN foram identificadas comunidades científicas, pesquisadores influenciadores e pesquisadores que participam de mais de uma comunidade. Um dos desafios encontrados, ao analisar uma rede científica com grande volume de dados, foi a variação de densidade que dificulta a parametrização das entradas do algoritmo de agrupamento Para resolver este problema, sem a necessidade de se calcular os valores ideais de cada parâmetro repetidamente e em tempo de execução, foi proposto um método de subagrupamento que foi utilizado para refinar grupos formados por muitos pesquisadores.

Através de uma entrevista informal, com observação direta como fonte de coleta de dados, os resultados obtidos foram apresentados a seis pesquisadores para que eles os analisassem. Cinco deles reportaram que as comunidades nas quais eles estavam envolvidos estava condizente com a realidade dos seus grupos de pesquisa. Apenas um pesquisador reportou que, de fato, desenvolvia suas pesquisas com os pesquisadores que ele estava relacionado, mas considerou o grupo ao qual ele estava vinculado muito grande. Então, foi apresentado para ele os sub-agrupamentos do seu grupo original, o qual confirmou que o seu subgrupo refletia a realidade de seu grupo de pesquisa.

As análises realizadas apontam para a viabilidade tanto do modelo quanto dos algoritmos propostos. Através da avaliação realizada pode-se dizer que há indícios de que o resultado alcançado foi satisfatório e que os métodos apresentados podem ser evoluídos para alcançar uma coesão maior no agrupamento.

7. Considerações Finais

Neste trabalho foi modelada uma rede social científica (DBLP), onde os elementos representam pesquisadores e suas ligações como relacionamentos de coautoria. Para essa modelagem foi adotado um grafo bidirecional para possibilitar a representação de níveis de influência diferentes entre os pesquisadores.

Um dos objetivos da análise dessa rede social científica é compreender a sua estrutura e como os pesquisadores se influenciam no meio acadêmico. Para tal, foi desenvolvido um algoritmo de agrupamento por densidade, baseado no DBSCAN. Este possui adaptações para trabalhar em grafos com arestas direcionadas e com grande volume de dados. Os resultados obtidos no desenvolvimento deste trabalho apontam para a viabilidade da solução proposta.

Em trabalhos futuros uma evolução temporal das componentes conexas encontradas deve ser realizada para verificar se as comunidades menores tendem a serem incluídas na componente conexa dominante. Deseja-se também elaborar uma estratégia para a parametrização ótima no refinamento dos grupos. Por fim, avaliações adicionais devem ser realizadas, incluindo pessoas de diferentes regiões e países para que se possa explorar a solução em profundidade.

Referências

Almeida, R., Pereira, C. K., Campos, F., and Stroele, V. (2016). Recomendação de recursos educacionais para grupos: buscando soluções em redes sociais. In *Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016)*.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press.

Falkowski, T., Barth, A., and Spiliopoulou, M. (2007). Dengraph: A density-based community detection algorithm. In *In Proc. of the 2007 IEEE / WIC / ACM International Conference on Web Intelligence,*, pages 112–115.

Gialampoukidis, I., Tsikrika, T., Vrochidis, S., and Kompatsiaris, I. (2016). Community detection in complex networks based on dbscan* and a martingale process. In *2016 11th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)*, pages 1–6.

Li, X., Tan, Y., Zhang, Z., and Tong, Q. (2016). Community detection in large social networks based on relationship density. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE.

Lopes, G. R., Moro, M. M., Wives, L. K., and de Oliveira, J. P. M. (2010). Cooperative Authorship Social Network. In *AMW*.

Luo, T., Zhong, C., Ying, X., and Fu, J. (2011). Detecting community structure based on edge betweenness. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. IEEE.

Ströele, V., Zimbrão, G., and Souza, J. M. (2013). Group and link analysis of multirelational scientific social networks. *Journal of Systems and Software*, 86:1819–1830.

Wasserman, S. and Faust, K. (1994). *Social network analysis: Methods and applications*, volume 8. Cambridge university press.

paper:174092

Spark Scalability Analysis in a Scientific Workflow

Renan Souza[1, 2], Vítor Silva[1], Pedro Miranda[1], Alexandre A. B. Lima[1], Patrick Valduriez[3], Marta Mattoso[1]

[1]COPPE – Federal University of Rio de Janeiro
[2]IBM Research – Brazil
[3]Inria and LIRMM, Montpellier
{renanfs,silva,pmiranda,assis,marta}@cos.ufrj.br,
patrick.valduriez@inria.fr

Abstract. *Spark is being successfully used for big data parallel processing in many business domains (social media, finance, retail). Spark's scalability, usability, and large user community have motivated developers from scientific domains (bioinformatics, oil and gas, astronomy) to try it. However, scientific applications' profile, e.g., black-box programs and intense file writes, differs from traditional business workflows, which may affect its scalability. We present a scalability analysis of Spark in a real case-study in Oil and Gas domain. We explore workloads on a 936-cores HPC cluster processing 330 GB of scientific data. We show that it scales very well when running long-lasting scientific tasks, but its performance is lower for short-duration tasks.*

1. Introduction: Spark and Scientific Workflows

Spark is one of the most popular data parallel processing systems for business big data workflows. One common characteristic in such workflows is that they require large computing clusters to process vast amounts of data. This is also true for many workflows in scientific domains, such as bioinformatics, oil and gas, and astronomy [Atkinson et al. 2017]. Besides its scalability in business workflows [Armbrust et al. 2015, Shi et al. 2015], Spark has many other interesting features. For example, a dataflow-oriented execution model, an API using well-known programming languages (*e.g.*, Python, Scala, Java) that facilitates expressing linked data transformations (*e.g.*, map, reduce, filter, join) of typed data elements, an efficient fault-tolerance support, and a large user community. Such features have motivated scientific workflow users to migrate from traditional parallel scientific workflow management systems to Spark [Gittens et al. 2016, Oliveira et al. 2015, Zhang et al. 2017].

However, despite the shared characteristic of big data processing, scientific and business workflows have many differences. Scientific workflows are commonly designed as multiple linked Many-Task Computing (MTC) applications [Raicu et al. 2008]. That is, tasks in a scientific application consume input data, perform scientific computations, and produce output data that feeds a subsequent task of a different application, thus forming a dataflow between linked tasks. Each task may perform computations with varying complexity levels that last from milliseconds to a few minutes each. In addition, each task normally consumes and produces small files (tens of MB). Very often, scientific workflows expect that those files are stored at runtime on an HPC shared file system (*e.g.*, GPFS, Lustre) for data communication between tasks. In total, an entire workflow execution may take days of continuous run and process terabytes of data. In contrast, common big data workflows read large input files in their

beginning, process them, and write aggregated output data on disk to be analyzed, queried, and used for plotting charts and interactive dashboards. Spark is highly benefited in this case, as it can load from disk only once, process the data in its Resilient Distributed Datasets in the cluster memory, and write results to disk in the end. Besides, differently than typical scientific workflows, business workflows are good fit for shared-nothing clusters, often seen in Spark deployments, as disk I/O is reduced.

In addition, traditional scientific workflow systems deal with scientific applications as black-boxes. That is, tasks are aware of input and output data, but the internal computation is opaque. When developing a scientific workflow, users do not want to rewrite very complex (already tested, optimized, and stable) scientific programs using a different programming language or, oftentimes, they only have the executable binaries and not the source code. Spark API allows developers to completely write their code using native data transformations. It enables Spark's engine to be aware of the internals of each task and perform runtime optimizations for the benefit of in-memory data processing and data locality. This is not achievable when running typical scientific workflows developed as chains of black-boxes. Thus, these scientific workflows' characteristics may harm Spark's performance. Although several works have analyzed Spark's performance in business workflows, very few have investigated Spark in scientific workflows [Gittens et al. 2016, Oliveira et al. 2015, Zhang et al. 2017]. The ones that do either adapt their code to enable Spark's runtime optimizations, or limit those multiple disk I/O commonly seen in scientific applications, or use a shared-nothing cluster, rather than a traditional HPC shared file system. Therefore, we are not aware of works that have investigated Spark's scalability in scientific workflows by exploring multiple typical scientific workloads using the traditional profile of scientific applications, such as parameter sweep workflows, one of the most representative class of workflow applications [F. da Silva et al. 2017].

In this work, we use a real case study in Oil and Gas domain to build a scientific workflow in Spark and understand its scalability. We do not modify the original scientific application codes, but we model its parameter sweep as a dataflow using Spark. We use native Spark operators for external calls to the black-box applications, which write multiple files on a shared disk during execution. Source code of the dataflow implementation is on [GitHub]. We deploy Spark on a large HPC cluster, with 936 cores, where all nodes share a fast file system. We perform scalability analysis by varying multiple workloads, typical in scientific workflows, to understand the system performance. We show that Spark scales very well for workflows with long-lasting tasks, but its performance is lower for short-duration tasks. In the remainder of this paper, we explain the scientific workflow utilized in Section 2, the scalability analysis in Section 3, and the conclusion in Section 4.

2. Risers Fatigue Analysis Workflow in Spark

We model and execute a synthetic implementation of Riser Fatigue Analysis (RFA) scientific workflow based on a real case study in Oil and Gas domain. Specific details about this workflow can be found in [Souza et al. 2016]. Figure 1 shows RFA composed of 7 linked programs with a dataflow in between. The programs generate intermediary raw data files. Each of those files range from 8 to 14 MB, and there may be thousands of them in a real execution, generating terabytes of data.

To implement RFA in Spark, instead of re-writing those complex programs using its APIs, we follow the black-box approach to run external programs in a Spark dataflow. We use the natively available *Process* library in Spark to run external process executions. By doing so, it only manages the dataflow between each program call, while the actual task computation and file writes to the shared file system are outsourced in the scientific applications. Figure 2 shows a higher-level implementation of RFA in Spark, where we omit intermediary data manipulation for the dataflow, which is done inside Spark. The actual implementation is publicly available on [GitHub].

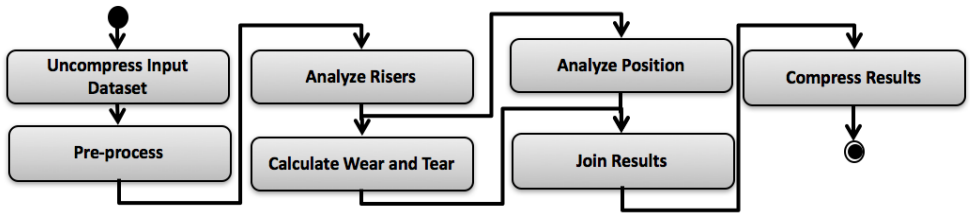


Figure 1. Risers Fatigue Analysis workflow.

```
1. val inputFiles = sc.textFile("/shared-disk/input-directory")
2. val uncompress = inputFiles.flatMap( x => Process("./Uncompress", x))
3. val preProcessing = uncompress.map(x =>Process("./Preprocess", x))
4. val analyzeRisers = preProcessing.map(x => Process ("./AnalyzeRisers", x))
5. val calcWearTear = analyzeRisers.filter(x => Process("./CalcWearTear", x))
6. val analyzePos = analyzeRisers.filter(x => Process("./AnalyzePos", x))
7. val jresults = calcWearTear.join(analyzePos)
8. val compressRes = jresults.reduceByKey((x,y) => Process("./Compress", x);  x)
```

Figure 2. Risers Fatigue Analysis workflow implemented in Spark.

3. Spark Scalability Analysis

We perform scalability analysis of Spark running RFA workflow. Scalability refers to the ability of a system to accommodate to a growth of the computing resources (*e.g.*, physical nodes) or the workload [Özsu and Valduriez 2011]. With respect to workload variation, in MTC workflows, there are at least two factors that compose a workload: number of tasks and task duration [Raicu et al. 2008]. Except for the last experiment, which has a real workload with varying task durations, we generate multiple synthetic workloads to represent different typical scenarios in scientific workflows to perform the scalability analysis. The task durations of the synthetic workloads follow a normal distribution with standard deviation 20% of the value of the task duration on average.

For software, we use native Spark version 1.6.1 with default settings, deploying on a standalone cluster using the cluster's shared file system. For hardware, we use a cluster with 39 nodes, with 24 cores each (936 cores in total), in Grid5000 (http://www.grid5000.fr). Each node has two AMD Opteron 1.7 GHz 12-core processors, 48GB RAM, connected via Gigabit Ethernet, and access a 10TB shared network storage. We repeat the experiments until the standard deviation of workflow execution times is less than 1, and the results are the average within the 1% margin.

Experiment 1: scalability varying workload and resources. We measure the execution time on 10 workers (240 cores) running about 6k tasks lasting 1 minute on average each. Then, we double the number of workers and tasks (480 cores - running 12k tasks). After, we execute using near four times the number of workers and tasks (936 cores - running 23.4k tasks). These numbers are representative for real RFA workloads [Souza et al. 2016] and other typical scientific workflows. Ideally, when the

workload increases proportionally to the computing resources added to the cluster, the execution time is expected to remain constant. As Figure 3-I shows, Spark is scalable for running a typical execution of RFA, as its curve remains close to a horizontal line. When the scale increases, the system scalability starts to decrease. However, running 936 tasks concurrently, Spark runs only 17% away from the ideal line, which is considered small in such large scale. These results lead us to further investigate the system performance by exploring a wider variety of scientific workloads fixing the computing resources in its maximum, as in all following experiments.

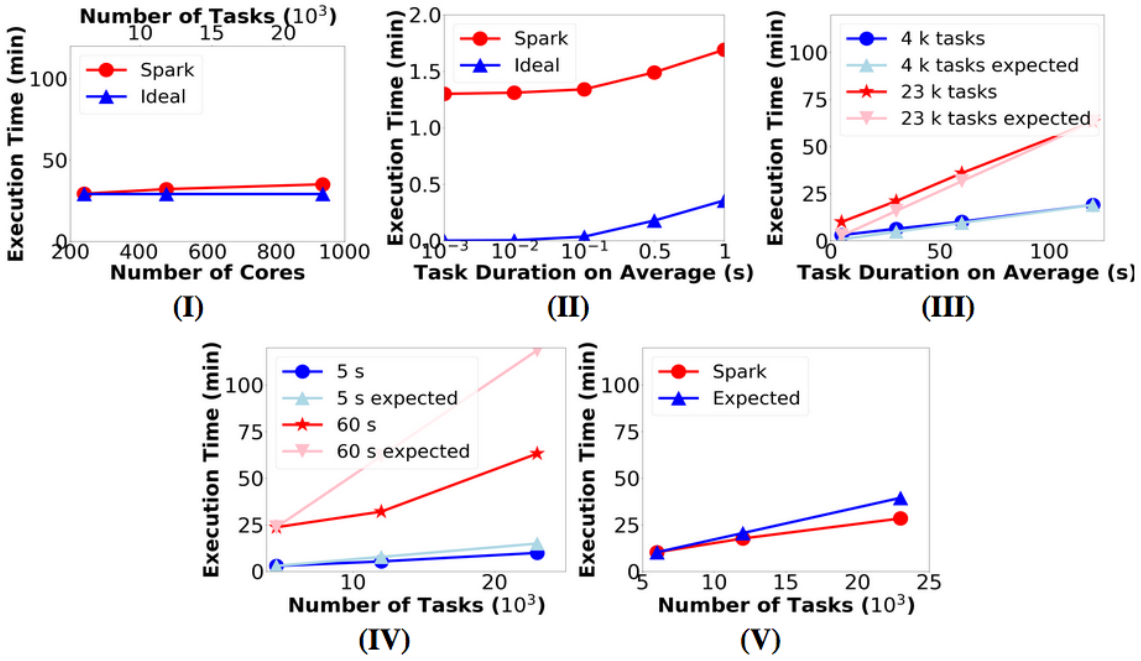


Figure 3. Experiments' results.

Experiment 2: varying task duration from very short to short-term. Although in MTC workflows tasks are commonly long-term (*i.e.*, many seconds or few minutes on average each) [Raicu et al. 2008], we evaluate Spark's scalability on 20 k tasks, varying task duration on average from very short (taking few milliseconds) to short (up to 1 second). These workloads are present in some scientific applications that perform very simple processing. By running short tasks, we are stressing the task scheduling system. Results are in Figure 3-II. We calculate the ideal time as *(Number of Tasks * Tasks Duration on Average)/Number of Cores*. Results show that Spark runs considerably below the ideal time. Spark has a higher initialization time (it is scheduling to all 936 cores) compared to the workload being computed (the entire computation takes at most 2 minutes only). When this happens, Spark's performance is compromised, as the best result (80% away from ideal) happened for 1-second tasks. It indicates that when the duration increases, Spark's performance gets slightly closer to the ideal line.

Experiment 3: fixed number of tasks, varying task duration. Experiment 2 indicates that when task duration increases, Spark's performance tends to increase. In this experiment, we further analyze this. We vary average task duration, from short (5 s) to long (120 s), fixing two different numbers of tasks: small (4.6 k) and large (23.4 k, about 5 times 4.6 k). Based on Experiment 2 results, we plot two expected lines by setting the base result (where Spark achieves best performance) as the longest task

durations evaluated, *i.e.*, 120 s on average, and then we expect Spark's performance to degrade when task duration decreases. As Figure 3-III shows, this is exactly what happens. To understand the results, we analyze the differences between the expected lines and the actual lines. In this experiment, when these differences are small, Spark maintains its high scalability for each workload analyzed. Ideally, this difference should remain small. For both fixed number of tasks, these differences vary similarly. However, in both cases we see that the shorter the task duration, the greater the difference. For 60-seconds tasks, the differences are about 10%, whereas for 5-seconds tasks, the difference is 73%. The reasons for this are that there are many more frequent concurrent writes to disk during execution and the task scheduling system is overloaded.

Experiment 4: Fixed task durations, varying number of tasks. In this experiment, we fix two task durations (5 s and 60 s) and increase the number of tasks from small (4.6 k), to mid (12 k), and large (23.4 k). We expect that execution time increases proportionally to the number of tasks. Results are in Figure 3-IV. We also analyze the difference between actual and expected results. In this experiment, the larger these differences, the faster Spark runs than the expected time. That is, Spark is more scalable when these differences are larger. The differences between expected and actual time does not vary significantly when number of tasks increases. Thus, Spark's performance is less sensitive to the number of tasks. However, for 23.4 k tasks, the difference between expected and actual time for 60 s task duration is 46%, whereas for 5 s task duration, the difference is 33%. Therefore, Spark has better scalability for longer tasks and number of tasks variation does not significantly affects its performance.

Experiment 5: Mixed task durations, varying number of tasks. Up to this point, we analyzed Spark's scalability under different conditions by executing synthetic workloads. Now, we investigate a more realistic scenario. We analyze a real workload for the RFA workflow, which mixes short, mid, and long-term tasks. In this workload, on average, *Uncompress* tasks take 1 s each; *Pre-processing* tasks take 10 s, *Analyze Risers* tasks take 60 s, *Calculate Wear and Tear* tasks take 30 s, *Analyze Position* tasks take 15 s, *Join Results* takes 1 s, and *Compress Results* tasks take 15 s. We vary number of tasks as 6 k, 12 k, and 23.4 k. We expect that the execution time increases proportionally to the number of tasks growth. Figure 3-V shows that Spark scales well on all executions as it runs faster than the expected. The difference between actual and expected computations increases as the number of tasks increases. An execution for the 23.4 k tasks workload takes about 28 min to complete, whereas the expected time would be 39.4 min (28% difference). Thus, Spark scales well for a real scientific workload.

4. Conclusion

In this paper, we analyzed Spark's scalability when running a typical HPC scientific workflow that processes large amounts of data, does multiple file writes in a shared file system, and is implemented as black-box scientific programs linked through dataflows. As mentioned in [Zhang et al. 2017], reusing existing codes avoids rewriting, which is error prone. We use a real case-study workflow in Oil and Gas domain on a large HPC cluster with 936-cores and a fast shared file system. We showed that Spark scales well, even for a real workload. We found that the number of tasks does not significantly affect its performance, but the task duration does. For multiple short tasks representing simple scientific tasks with little computation, Spark does not scale well, whereas for long-

lasting scientific tasks, it does. Since long-lasting tasks are more common in scientific workflows, using Spark to manage the dataflow between tasks with black-box scientific application external calls does not harm its scalability. Altogether, in addition to its large user community and easy APIs, its scalability encourages scientific workflow developers to migrate their many-task parameter sweep workflows to Spark, without requiring re-implementation of the complex linked scientific programs. As future work, we plan to carry out more and deeper performance analyses to investigate load balancing, memory utilization, and disk throughput during execution in different scientific workloads.

Acknowledgements

This work was partially funded by CNPq, FAPERJ and Inria (MUSIC and SciDISC projects), EU H2020 Programme and MCTI/RNP-Brazil (HPC4E grant no. 689772), and performed (for P. Valduriez) in the context of the Computational Biology Institute (www.ibc-montpellier.fr). The experiments were carried out using the Inria Grid'5000 testbed (https://www.grid5000.fr).

References

Armbrust, M., Zaharia, M., Das, T., Davidson, A., Ghodsi, A., Or, A., Rosen, J., Stoica, I., Wendell, P., et al., (2015), "Scaling spark in the real world: performance and usability", *PVLDB*, v. 8, n. 12, p. 1840–1843.

Atkinson, M., Gesing, S., Montagnat, J., Taylor, I., (2017), "Scientific workflows: past, present and future", *FGCS*, v. 75, p. 216–227.

F. da Silva, R., Filgueira, R., Pietri, I., Jiang, M., Sakellariou, R., Deelman, E., (2017), "A characterization of workflow management systems for extreme-scale applications", *FGCS*, v. 75, p. 228–238.

GitHub. RFA Spark Repository. Available on: github.com/hpcdb/RFA-Spark.

Gittens, A., Devarakonda, A., Racah, E., Ringenburg, M., Gerhardt, L., Kottalam, J., Liu, J., Maschhoff, K., Canon, S., et al., (2016), "Matrix factorizations at scale: A comparison of scientific data analytics in Spark and C+MPI using three case studies". In: *IEEE Int. Conf. on Big Data*, p. 204–213

Oliveira, D., Boeres, C., Neto, A., Porto, F., (2015), "Avaliação da localidade de dados intermediários na execução paralela de workflows bigdata". In: *SBBD*, p. 29–40

Özsu, M. T., Valduriez, P., (2011), *Principles of distributed database systems*. 3 ed. New York, Springer.

Raicu, I., Foster, I. T., Zhao, Y., (2008), "Many-task computing for grids and supercomputers". In: *MTAGS*, p. 1–11

Shi, J., Qiu, Y., Minhas, U. F., Jiao, L., Wang, C., Reinwald, B., Özcan, F., (2015), "Clash of the titans: MapReduce vs. Spark for large scale data analytics", *PVLDB*, v. 8, n. 13, p. 2110–2121.

Souza, R., Silva, V., Coutinho, A. L. G. A., Valduriez, P., Mattoso, M., (2016), "Online input data reduction in scientific workflows". In: *WORKS*, p. 44–53

Zhang, Z., Barbary, K., Nothaft, F. A., Sparks, E. R., Zahn, O., Franklin, M. J., Patterson, D. A., Perlmutter, S., (2017), "Kira: processing astronomy imagery using big data technology", *IEEE Trans. Big Data*, v. PP, n. 99, p. 1–14.

paper:171406

Spatial Join on Positional Uncertain Data

Welder B. Oliveira[1], Sávio S. T. Oliveira[1], Vagner J. S. Rodrigues[1], Helton S. B. Santos[2], Kleber V. Cardoso[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brazil

[2]Departamento de Estatística – Universidade de Brasília (UNB)
Campus da Unb, Sgan - Asa Norte, UnB - Brasília, DF, 70910-900.

Abstract. *This paper presents a probabilistic spatial join on positional uncertain data designed to be a) generalist; b) accurate and c) efficient. A proposed progressive Monte Carlo algorithm is used in the refinement step and the Chebyshev inequality is applied in the filtering one in order to provide efficiency, efficacy and generality. The experiments show that the current propose is Pareto efficient concerning these requirements, i.e., it is not outperformed by any competing method. Also, the solution's parameters relating accuracy and efficiency may be adjusted to maximize the gain in one while relaxing the other according to user's demand.*

1. Introduction

Spatial applications frequently need to combine two data sets based on some spatial relationship between objects in these two data sets [Patel and DeWitt 2000]. That is the case when a spatial join is required. While very useful, in general, the evaluation of spatial join predicates has high computing costs and demand sophisticated solutions in order to achieve satisfactory performance for large datasets as one can see in [Patel and DeWitt 2000].

The computational cost becomes even higher when we need to deal with objects which the position is not precise. This can happen due to several reasons, e.g., due to the imprecision of the georreferecing technique employed to acquire the data. In this context, the join predicate may not be accurately evaluated, having severe impact on the results presented by the applications. Thus, robust solutions based on some probabilistic approach are natural candidates for improving the spatial join. We indroduce a new algorithm based to perform a probabilistic spatial join based, which we call Progressive Monte Carlo Spatial Join (PMCSJ). Furthermore, PMCSJ is designed to address the following requirements: accuracy, efficiency and generality. Section 2 comments the related works, Section 3 shows the solution, Section 4 presents the results and Section 5 concludes the work.

2. Related Work

There exists several methods for performing a deterministic spatial join, using or not indexed datasets [Mishra and Eich 1992, Arge et al. 1998, Jacox and Samet 2003, Luo et al. 2002, Patel and DeWitt 1996, Elmasri 2008, Lo and Ravishankar 1994, Brinkhoff et al. 1993, Huang et al. 1997]. These methods assume absolute precision in spatial objects' coordinates. Also, several works

[Wolfson et al. 1999, Pfoser and Jensen 1999, Yu and Mehrotra 2003, Zhang et al. 2012] have been proposed to deal with spatial joins on moving objects, which is not the case of the present work since we are interested in modeling the error coming from imprecise georreferencing and not by moving objects. Spatial join on existentially uncertain were explored by [Dai et al. 2005] and [Ljosa and Singh 2008] presented an approach able to deal with both existencial and positional uncertainty by using a score function in which the two kinds of uncertainty are considered.

The work [Openshaw 1989] proposed the use of the Monte Carlo Method (MMC) to compute the intersection probabilities of positional uncertain geometries, which we will call here Random Spatial Join (RSJ). A limitation faced by RSJ is the computational cost of the MCM. However, RSJ is generic method concerning the PDF error assumption and we will compare it with PMCSJ in Section 4.

In the probabilistic spatial join proposed by [Ni et al. 2003], the error is modeled using a Circular Normal distribution. However, as it does not handle errors from others distributions, it will not be tested.

3. Probabilistic Spatial Join

Like the main proposals in deterministic spatial join, PMCSJ is devided in the classical two steps: *filtering* and *refinement*. The filtering step is performed on a novel probabilistic version of the MBR - the confidence rectangles (CR). The refinement is done by the novel Progressive Monte Carlo Method (PMCM).

3.1. Filtering Step

The index is built using an *R-tree* with CRs replacing MBRs. In order to do that, the MBR for a given geometry is expanded in vertical and horizontal directions by a shift value d. A formal definition for CR is presented below.

Definition 1. *Given a geometry g and a threshold probability p, a confidence rectangle (CR) with threshold probability p for g is the one which contains the MBR of g and the probability of g lies inside it is at least equal to $\sqrt{1-p}$.*

Theorem 1. *Let G and H be CRs for the geometries g and h respectively. If G and H do not intersect then the intersection probability between g and h are less than p.*

Proof. Since G and H are CRs for g and h then, follows from the definiton that $Pr(g \subset G) \geq \sqrt{1-p}$ and $Pr(h \subset H) \geq \sqrt{1-p}$. Then, assuming independence in the error direction in the g and h coordinates,

$$Pr(g \subset G, h \subset H) = Pr(g \subset G).Pr(h \subset H) \geq (\sqrt{1-p})(\sqrt{1-p}) = 1-p.$$

Thus, $Pr(g \not\subset G \cup h \not\subset H) < p$.

If G and H do not intersect then g or h must not be contained in its CR, whose probability is less than p by the above equation. Thus, $Pr(g \cap h) < p$. □

To build a CR, we use of the Chebyshev inequality, which declares: If X is a integrable random variable with finite mean $\mu = E(X)$ and standard deviation σ, then for any $k > 0$, $Pr(|X - E(X)| > k\sigma) < 1/k^2$. For X positive,

$$Pr(X - E(X) > k\sigma) < 1/k^2 \implies Pr(X \leq E(X) + k\sigma) \geq \frac{k^2 - 1}{k^2}.$$

In our case, X is the positional error. Thus, the probability for the error be at most $d = E(X) + k\sigma$ is at least $\frac{k^2-1}{k^2}$. As a given geometry must lie in the CR with a probability of at least $\sqrt{1-p}$, then

$$\sqrt{1-p} = \frac{k^2 - 1}{k^2} \implies k = \sqrt{\frac{1}{1 - \sqrt{1-p}}}.$$

The CR is given by the coordinates $P_{min} = (x_{min}, y_{min})$ and $P_{max} = (x_{max}, y_{max})$, with $x_{min} = x_{MBR_min} - d$, $y_{min} = y_{MBR_min} - d$, $x_{max} = x_{MBR_max} + d$ and $y_{max} = y_{MBR_max} + d$.

3.2. Refinement

The *refinement* step is given by the Algorithm 1 which perform the novel PMCM procedure. The performance gain comes from avoiding more simulations than the sufficient to guarantee the predicate evaluation with a condifence level γ for each candidate geometry pair.

Algorithm 1: Progressive Monte Carlo Method.

Data: a and b: two geometries;
p: threshold probability;
m: size of each simulation batch;
n_{max}: maximum number of simulations allowed
Result: Return TRUE if the success proportion is at least equal to p, FALSE otherwise.

1 Initialize with zero the counter n (number of Monte Carlo realizations up to now).
2 Shift the a and b coordinates m times.
3 Update n by m units ($n \leftarrow n + m$).
4 Compute the proportion $\hat{q}$ of success in the n simulations.
5 Compute the Confidence Interval (CI) for q, i.e.,

$$CI(q, \gamma) = \left[\hat{q} - t_c\sqrt{\frac{\hat{q}(1-\hat{q})}{n}}, \hat{q} + t_c\sqrt{\frac{\hat{q}(1-\hat{q})}{n}}\right]$$

where t_c is the $(1+\gamma)/2$ quantile of the t-Student distribution with $n-1$ degrees of freedom.
6 If $p \in IC(q, \gamma)$ and $n < n_{max}$, go to steps 2-5.
7 If $\hat{q} \geq p$ return **TRUE** else return **FALSE**.

4. Results

4.1. Test setup

The datasets used in test were: 1) vegetation, 2) deforestation and 3) wildfire areas, both from the Brazilian province called Goiás and procuced by the LAPIG-UFG laboratory. The join executed was vegetation against the union of deforestation with wildfire areas. The competing approach are Random Spatial Join (RSJ) with $m = 150$ simulations and PMCSJ with a maximum number of simulations $n.max$ of 150 and 1000. The positional error Y follows a Half Normal distribution, defined by $Y = |X|$, with $X \sim N(200, 100^2)$. The mean and standard deviation (required as parameters for CR computation) of the Half Normal distribution are: $E(X) = \sigma\sqrt{\frac{2}{\pi}}$ and $sd(X) = \sigma\sqrt{1 - \frac{2}{\pi}}$. The threshold probabilities tested are: 0.10, 0.20, ..., 0.80 and 0.90. The accuracy parameter for PMCSJ is $\gamma = 0.99$, the size of each simulation batch set to 50.

A ground truth database was used to compare the methods, which contains the reference intersection probabilities for each geometry pair (a_i, b_j), with $a_i \in A$, $b_j \in B$, with A and B being the two joined data sets. The ground truth was built with a larger number N of Monte Carlo simulations, in our work, $N = 500$. The number $N = 500$ provides a margin or error $E = 0.045$.

The metrics used for comparison are: i) proportion of neighborhood false positives; ii) proportion of neighboorhod false negatives; and iii) processing time. The RSJ and PMCSJ will be compared with relation to the metrics: proportion of false and positives negatives in a "safe" R-neighboorhod of the threshold probability p. An error margin interval of size ϵ is taken from the test to avoid a judgment error caused just by the imprecision of the ground truth. The metrics are defined as $S_{FN} = \frac{\#(p-\infty,p)_{method}}{\#(p+\epsilon,p+R)_{gTruth}}$ and $S_{FP} = \frac{\#(p,\infty)_{method}}{\#(p-R,p-\epsilon)_{gTruth}}$. The execution was performed by a Intel Core i5-4200U, 1.6GHz CPU (4 threads).

4.2. Results and Discussion

Table 1 presents the metric type, S_{FP} or S_{FN}, the threshold probability p in the parenthesis, the metric value for RSJ and PMCSJ for a maximum number of 150 and 1000 simulations and a signal: positive if PMCSJ was more accurate than RSJ, negative if the winner was RSJ and neutral in case of a tie.

métrica	JEA-150	JEPMCP-150	JEPMCP-1000	sinal 1	sinal 2
$S_{FN}(0, 10)$	0,200	0,000	0,000	+	+
$S_{FN}(0, 20)$	0,000	0,056	0,000	-	
$S_{FN}(0, 50)$	0,000	0,333	0,167	-	-
$S_{FN}(0, 70)$	0,000	0,111	0,000	-	
$S_{FN}(0, 80)$	0,133	0,200	0,000	-	+
$S_{FP}(0, 20)$	0,000	0,045	0,000	-	
$S_{FP}(0, 30)$	0,000	0,077	0,000	-	
$S_{FP}(0, 40)$	0,100	0,100	0,000		+
$S_{FP}(0, 70)$	0,200	0,000	0,000	+	+
$S_{FP}(0, 90)$	0,100	0,000	0,000	+	+

Table 1. Comparing JEA and JEPMCP accuracies (just non tied scenarios).

The non parametric signal test was used to evaluate if the differences observed were statistically significant. The test evaluated the two set of hypothesis: "Set 1: (H_0) PMCSJ is so accurate as RSJ" against "(H_1) PMCSJ is less accurate than RSJ" when comparing RSJ-150 with PMCSJ-150 and "Set 2: (H_0) PMCSJ is so accurate as RSJ"

against "(H_1) PMCSJ is less accurate than RSJ" when comparing RSJ-150 with PMCSJ-1000. The test statistic **t** is given by the number of "+" considering just the non tied cases. For set 1, the p-value was 0,254 which does not raise statistically significant evidence in favor of a relevant superiotity of RSJ-150 when comparing with PMCSJ-150. For set 2, the p-value was 0,11 which is a more strong evidence in favor of a PMCSJ-1000 superiotity against RSJ-150.

RSJ took an average 257 seconds to perform the deforestation X (vegetation U wildfire) join against 64 of PMCSJ-150 and 73 of PMCSJ-1000. Thus, PMCSJ was able to be more efficient while keeping at least the same accuracy as RSJ, being a preferable method to perform a probabilistic spatil join.

5. Conclusion

The present work built and applied a variant of the Monte Carlo method and a consequence of the Chebyshev inequality to create a geneneralist probabilistic spatial join solution (PSJ) - which we call PMCSJ. The experiments showed that PMCSJ are: a) generalist concerning the positional error distribution; b) accurate; and c) efficient. Previous correlated works either were not a generalist solution for PSJ or presented a poorer performance. PMCSJ is shown to be a better generalist method for PSJ concerning the these three requirements.

The high Monte Carlo simulations cost are mitigated both by the progressive approach which applies just the required number of Monte Carlo simulations and the probabilistic filtering step which avoid several unecessary Monte Carlo simulations by applying a probabilistic and geneneralist version of the minimum bounding rectangles, which we call confidence rectangles. Both are contributions of our work.

Future works may try to build more powerful filtering steps, for example by finding a more tight shift value for confidence rectangles which still guarantees that the non intersection of two of them imply that the geometry inside them also do not intersect with a probability of at least p. Another possibility is trying to adapt the two spatial join steps to specifities of a family distribution - such as the Exponential Family, for example. That would keep the generalist requirement while advancing in performance and accuracy.

References

Arge, L., Procopiuc, O., Ramaswamy, S., Suel, T., and Vitter, J. S. (1998). Scalable sweeping-based spatial join. In *VLDB*, volume 98, pages 570–581. Citeseer.

Brinkhoff, T., Kriegel, H.-P., and Seeger, B. (1993). *Efficient processing of spatial joins using R-trees*, volume 22. ACM.

Dai, X., Yiu, M. L., Mamoulis, N., Tao, Y., and Vaitis, M. (2005). Probabilistic spatial queries on existentially uncertain data. In *International Symposium on Spatial and Temporal Databases*, pages 400–417. Springer.

Elmasri, R. (2008). *Fundamentals of database systems*. Pearson Education India.

Huang, Y.-W., Jing, N., and Rundensteiner, E. A. (1997). Spatial joins using r-trees: Breadth-first traversal with global optimizations. In *VLDB*, volume 97, pages 25–29. Citeseer.

Jacox, E. H. and Samet, H. (2003). Iterative spatial join. *ACM Transactions on Database Systems (TODS)*, 28(3):230–256.

Ljosa, V. and Singh, A. K. (2008). Top-k spatial joins of probabilistic objects. In *2008 IEEE 24th International Conference on Data Engineering*, pages 566–575. IEEE.

Lo, M.-L. and Ravishankar, C. V. (1994). Spatial joins using seeded trees. In *ACM SIGMOD Record*, volume 23, pages 209–220. ACM.

Luo, G., Naughton, J. F., and Ellmann, C. J. (2002). A non-blocking parallel spatial join algorithm. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 697–705. IEEE.

Mishra, P. and Eich, M. H. (1992). Join processing in relational databases. *ACM Computing Surveys (CSUR)*, 24(1):63–113.

Ni, J., Ravishankar, C. V., and Bhanu, B. (2003). Probabilistic spatial database operations. In *International Symposium on Spatial and Temporal Databases*, pages 140–158. Springer.

Openshaw, S. (1989). Learning to live with errors in spatial databases. *Accuracy of spatial databases*, pages 263–276.

Patel, J. M. and DeWitt, D. J. (1996). Partition based spatial-merge join. In *ACM SIGMOD Record*, volume 25, pages 259–270. ACM.

Patel, J. M. and DeWitt, D. J. (2000). Clone join and shadow join: two parallel spatial join algorithms. In *Proceedings of the 8th ACM international symposium on Advances in geographic information systems*, pages 54–61. ACM.

Pfoser, D. and Jensen, C. S. (1999). Capturing the uncertainty of moving-object representations. In *International Symposium on Spatial Databases*, pages 111–131. Springer.

Wolfson, O., Sistla, A. P., Chamberlain, S., and Yesha, Y. (1999). Updating and querying databases that track mobile units. In *Mobile Data Management and Applications*, pages 3–33. Springer.

Yu, X. and Mehrotra, S. (2003). Capturing uncertainty in spatial queries over imprecise data. In *International Conference on Database and Expert Systems Applications*, pages 192–201. Springer.

Zhang, R., Qi, J., Lin, D., Wang, W., and Wong, R. C.-W. (2012). A highly optimized algorithm for continuous intersection join queries over moving objects. *The VLDB JournalâThe International Journal on Very Large Data Bases*, 21(4):561–586.

paper:174658

Uma Abordagem para Processamento Distribuído de Junção por Similaridade sobre Múltiplos Atributos

Diego Junior do Carmo Oliveira[1], **Felipe Ferreira Borges**[1], **Leonardo Andrade Ribeiro**[1]

[1]Instituto de Informática
Universidade Federal de Goiás (UFG) – Goiânia – GO – Brazil

{diegooliveira,felipeferreiraborges,laribeiro}@inf.ufg.br

Abstract. *Similarity join is a fundamental operation in data integration. Most existing algorithms consider single-attribute data. However, real data is typically multi-attribute. Besides requiring more complex similarity expressions, this type of data is larger and, therefore, processing cost on a single machine can be prohibitively expensive. This paper presents a distributed similarity join algorithm on multi-attribute data using Spark. Initial experimental results show that the proposed approach is efficient and scalable.*

Resumo. *Junção por similaridade é uma operação fundamental em integração dados. Algoritmos existentes assumem, em sua ampla maioria, dados representados por um único atributo. Contudo, dados reais são tipicamente compostos por múltiplos atributos. Além de demandar expressões de similaridade mais complexas, dados desse tipo são mais volumosos e, com isso, o custo de processamento em um único computador pode tornar-se proibitivo. Este artigo apresenta um algoritmo distribuído de junção por similaridade sobre múltiplos atributos usando a plataforma Spark. Resultados experimentais iniciais mostram que a abordagem proposta é eficiente e escalável.*

1. Introdução

Sistemas de integração de dados frequentemente requerem que registros sejam comparados em termos de sua similaridade. Por exemplo, quando fontes de dados independentes, mas contendo informações em comum são integradas, representações divergentes de um mesmo objeto poderão estar presentes no banco de dados consolidado. A presença desse tipo de redundância pode causar inúmeros problemas em diversos cenários de aplicação, do reenvio desnecessário de correspondências até a corrupção de modelos de mineração de dados. A comparação baseada em similaridade é imprescindível para identificação desses dados duplicados porque os mesmos não são cópias idênticas entre si.

Junção por similaridade é uma operação fundamental nesse contexto, pois a mesma retorna todos os pares de objetos similares em uma coleção. Dois objetos são considerados similares se o valor retornado por uma função de similaridade para os mesmos for maior ou igual a um limiar predefinido. Para dados representados por atributos textuais, um tipo popular de junção por similaridade primeiramente converte o texto de cada objeto em um conjunto e, posteriormente, a similaridade entre pares de objetos é determinada pela interseção dos conjuntos correspondentes [Chaudhuri et al. 2006, Ribeiro and Härder 2011].

Tabela 1. Registros com informações sobre pessoas.

Pessoa ID	RID	Nome	Via	Cidade
1	1	João Ávila	Av. Pedro Paulo de Souza	Goiânia
	2	J. Ávila	Av. Pedro Paulo de Souza	Goiânia
2	3	Pedro Paulo	Av. Esperança	Goiânia
3	4	Pedro Paulo	Av. João Naves Ávila	Uberlândia

A ampla maioria dos algoritmos de junção por similaridade propostos consideram objetos representados por apenas um atributo. Entretanto, dados reais são tipicamente compostos por múltiplos atributos. Algoritmos tradicionais ainda podem ser empregados nesta situação usando um atributo ou a concatenação de múltiplos atributos na representação dos objetos. Infelizmente, ambas abordagens possuem sérias limitações. Por exemplo, considere os registros sobre pessoas ilustrados na Tabela 1. Os quatro registros representam três pessoas distintas, pois registros 1 e 2 referem-se a mesma pessoa. Caso seja empregado apenas o atributo `Nome`, apenas o par formado por registros 3 e 4 poderia ser retornado, pois registros 1 e 2 possuem textos diferentes devido à abreviação no último. Caso todos atributos sejam concatenados, o par formado por registros 1 e 4 poderia ser retornado devido à similaridade entre os valores nos atributos `Nome` e `Via`.

Uma abordagem mais interessante seria realizar a avaliação de similaridade em cada atributo isoladamente. Mais ainda, diferentes funções e limiares de similaridade podem ser definidos, possibilitando a composição de expressões de similaridade mais sofisticadas. O principal desafio é incorporar tais modificações em uma junção por similaridade sem sacrificar desempenho e escalabilidade. O emprego de expressões de similaridade complexas torna a comparação entre registros mais custosa computacionalmente, além de que dados representados por múltiplos atributos são naturalmente mais volumosos.

Este artigo apresenta um algoritmo distribuído de junção por similaridade sobre múltiplos atributos. Até o momento, não se tem conhecimento de trabalhos anteriores que investigaram o processamento desse tipo de junção por similaridade em um ambiente distribuído. A abordagem generaliza algoritmos existentes para possibilitar o emprego de expressões de similaridade baseadas em fórmulas Booleanas arbitrariamente complexas. Uma estratégia simples, mas eficiente de particionamento de dados é adotada e implementada em *Apache Spark* [Zaharia et al. 2016]. Resultados iniciais mostram que a solução proposta é eficiente e escalável quando aplicada em grandes volumes de dados.

2. Conceitos Básicos e Trabalhos Relacionados

Seja R uma coleção de registros representados por uma lista de atributos textuais $\mathcal{A} = (a_1, ..., a_k)$. $R.a$ denota o domínio do atributo a em R e $r.a$ denota o valor do atributo a em um registro r. Seja $f : R.a \times R.a \rightarrow [0, 1]$ uma função de similaridade que, dados dois valores $r.a$ e $s.a$ obtidos dos registros r e s, retorna um valor entre 0 e 1, e quanto maior o valor, maior a similaridade. Seja p um predicado de similaridade da forma $f\,(r.a, s.a) \geq \alpha$, onde f é uma função de similaridade e $\alpha \in [0, 1]$ é um limiar de similaridade. Seja $\phi : R \times R \rightarrow \{true, false\}$ uma expressão de similaridade representada por uma fórmula em forma normal conjuntiva contendo um conjunto de cláusulas $C = \{c_1, ..., c_t\}$, cujos literais são definidos por predicados de similaridade e literais negados não são permitidos, isto é, $\phi = \bigwedge_{c \in C} \bigvee_{p \in c}$.

Definição 1 (Junção por similaridade sobre múltiplos atributos)**.** *Seja R uma coleção de registros e ϕ uma expressão de similaridade. Um junção por similaridade sobre múltiplos atributos retorna todos os pares de registros $(r, s) \in R \times R$ tal que $\phi\,(r, s) = true$.*

O foco deste artigo são funções de similaridade baseadas em conjuntos. Com isso, o cálculo de similaridade entre dois valores textuais requer que os mesmos sejam representados por conjuntos. Existem diversas maneiras de se mapear texto para conjunto. Uma abordagem comum é coletar todas as sequências de caracteres de tamanho *q* de um texto; essas sequências são chamadas de *q-grams*. Por exemplo, o texto *"similar"* pode ser mapeado para o conjunto de *3-grams* {*'sim', 'imi', 'mil', 'ila', 'lar'*}. A função de similaridade *Jaccard* é popularmente usada para determinar a similaridade entre conjuntos. Dados dois conjuntos x e y, a similaridade *Jaccard* entre os mesmos é dada por $Jaccard\,(x, y) = \frac{|x \cap y|}{|x \cup y|}$. Outras funções de similaridade com as mesmas características que Jaccard são *Dice* e *Cosine* [Ribeiro and Härder 2011].

A representação baseada em conjuntos permite a aplicação de diversas otimizações. Uma das técnicas mais populares e efetivas é a chamada *filtragem por prefixo* [Chaudhuri et al. 2006]. Os elementos de todos conjuntos são ordenados de acordo com uma ordem global. Dado um conjunto x e um limiar α, seja $pref\,(x) \subseteq x$ o subconjunto de x composto por seus primeiros $|x| - \lceil |x| \times \alpha \rceil + 1$ elementos. É possível demonstrar que se $Jaccard\,(x, y) \geq \alpha$, então $pref\,(x) \cap pref\,(y) \neq \emptyset$. A filtragem por prefixo permite descartar conjuntos dissimilares verificando apenas uma fração dos mesmos.

O trabalho em [Vernica et al. 2010] explora prefixos para guiar o particionamento de dados em um algoritmo distribuído de junção por similaridade. Outra estratégia de particionamento baseada na diferença simétrica entre conjuntos é apresentada em [Deng et al. 2014]. Ambos trabalhos consideram registros compostos por apenas um atributo. O único trabalho conhecido até o momento para junções por similaridade sobre múltiplos atributos é apresentado em [Li et al. 2015]. O algoritmo proposto é estritamente sequencial, não-distribuído e suporta apenas expressões de similaridade conjuntivas. Em contraste, este trabalho apresenta um algoritmo distribuído que permite expressões de similaridade mais complexas. Após a conversão dos valores textuais em conjuntos, um registro r será representado por uma família de conjuntos. Por simplicidade, o restante deste artigo usará o termo registro para referir-se à família de conjuntos correspondente.

3. Junção por Similaridade Distribuída sobre Múltiplos Atributos

De maneira análoga a trabalhos anteriores [Vernica et al. 2010, Deng et al. 2014], a estratégia geral consiste em derivar um conjunto *assinaturas* de cada registro e usar essas assinaturas para guiar o particionamento de dados. Registros com uma assinatura em comum são enviados para um mesmo processador virtual (*worker*), onde a expressão de similaridade será avalidada entre os registros, ao passo que registros que não possuem uma assinatura em comum jamais serão comparados. Note que, como os processadores executam em paralelo, o tempo total de execução é determinado pelo maior tempo de comunicação e computação entre todos os *processadores*. Uma questão central no contexto deste trabalho é a geração de registros para múltiplos atributos. Para este fim, um atributo é selecionado como *chave de assinatura* e as assinaturas são produzidas a partir do conjunto correspondente. Aqui será adotado o esquema proposto em [Vernica et al. 2010], onde cada elemento do prefixo corresponde a uma assinatura;

Algoritmo 1: Junção por Similaridade Baseada em Múltiplos Conjuntos

Entrada: Uma coleção de registros R, uma expressão de similaridade ϕ, um inteiro i identificando a chave de assinatura.

Saída: Um conjunto S contendo todos os pares (r, s) tal que $\phi(r, s) = true$.

```
// Fase de particionamento
```

1 $flatMap\,(r)$ **para cada** $t \in pref\,(r.a_i)$ **faça**

2 $\quad (Key, lista\,(r)) \leftarrow groupByKey\,(Key, r)$

```
// Fase de verificação
```

3 $flatMap\,(Key, lista\,(r))$ **para cada** $candidatos \in lista_{(r)}$ **faça**

4 $\quad$ **se** $\phi\,(r, s) = true$ **então**

5 $\quad\quad S' \leftarrow S' \cup (candidatos)$

```
// Extrair resultado final
```

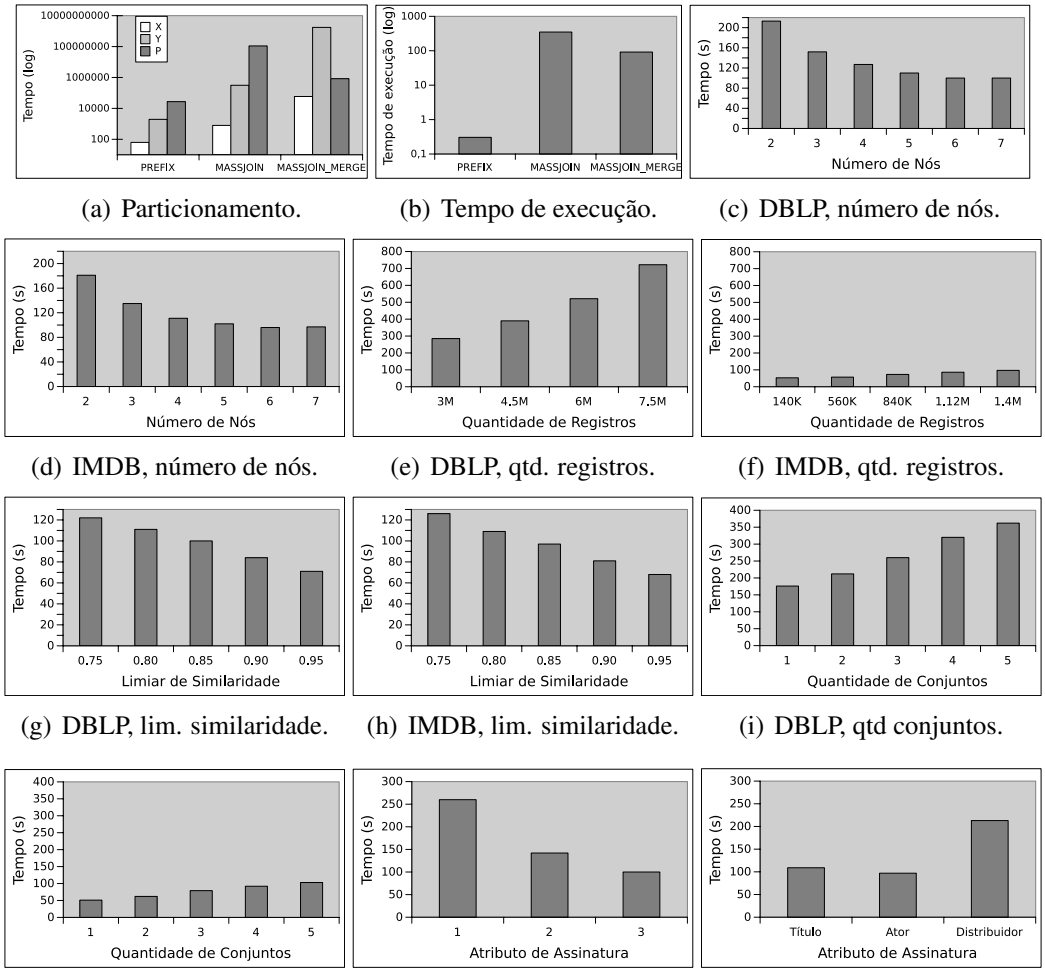
6 $S \leftarrow collect\,(S')$

uma comparação entre essa estratégia e o esquema em [Deng et al. 2014] é apresentada na Seção 4. A seleção da chave de assinatura deve minimizar o custo de máximo de comunicação e computação. Várias heurísticas podem ser usadas para este fim, por exemplo, pode-se escolher o atributo envolvido no predicado com o maior limiar de similaridade (e, portanto, possivelmente mais seletivo) ou contendo textos mais curtos. Outra opção, mais sofisticada, consiste na construção de modelos de predição de desempenho baseado em estatísticas [Sidney et al. 2015].

O Algoritmo 1 descreve os passos da abordagem proposta usando a plataforma *Spark*. O mesmo possui duas etapas: particionamento e verificação. Na etapa de particionamento, os elementos (*q-grams*) presentes no prefixo da chave de assinatura são usados para particionar os registros entre os processadores (linhas 1-2). Na segunda fase, registros associados a uma mesma assinatura são comparados em termos da expressão de similaridade (linhas 3-4). Como apresentado, o algoritmo pode produzir pares duplicados no resultado: dois registros similares podem conter mais de um *q-gram* em comum em seus prefixos e, com isso, estes registros serão enviados para diferentes processadores. Esse problema é evitado explorando a ordem global definida sobre os *q-grams* da chave de assinatura. A verificação de um par de registros avalia primeiramente o predicado associado à chave de assinatura. Caso o primeiro *q-gram* em comum encontrado for menor que a chave associada ao processador, então sabe-se que este par também foi enviado para outro processador e a avaliação pode ser seguramente interrompida.

4. Experimentos

As bases de dados usadas nos experimentos foram DBLP e IMDB, que contém artigos científicos e informações sobre filmes, respectivamente. Para o DBLP, foram usados os atributos título do artigo, nome da conferência, e nomes do primeiro autor e mais dois autores aleatórios. Para o IMDB, foram usados os atributos título, ator, distribuidor, diretor e produtor. Para cada registro extraído, foram criadas cópias sujas contendo $[1, 3]$ modificações textuais aleatórias. Os valores textuais foram mapeados para conjuntos de *3-grams*. Foram consideradas apenas expressões conjuntivas com um predicado baseado em Jaccard para cada atributo; o valor padrão do limiar de similaridade é 0.85.



(a) Particionamento. (b) Tempo de execução. (c) DBLP, número de nós.

(d) IMDB, número de nós. (e) DBLP, qtd. registros. (f) IMDB, qtd. registros.

(g) DBLP, lim. similaridade. (h) IMDB, lim. similaridade. (i) DBLP, qtd conjuntos.

(j) IMDB, qtd conjuntos. (k) DBLP, atr. assinatura. (l) IMDB, atr. assinatura.

Figura 1. Resultados dos experimentos.

O algoritmo foi implementado usando *Oracle Java 8, Scala 2.11 e Spark 2.1.1*. O *cluster* utilizado é composto por um mestre e sete escravos, todos eles equipados com processador *Intel Xeon W3565 quad-core com 3.2GHz, 8MB de cache CPU*, 8GB de memória principal, sistema operacional *Ubuntu 16.04 LTS* e recursos computacionais distribuídos gerenciados pelo *Yarn* baseado no *Hadoop 2.7.3*.

Inicialmente foram realizados testes para comparar os esquemas de assinaturas baseados no filtro de prefixo [Vernica et al. 2010] e os esquemas propostos em [Deng et al. 2014] para os algoritmos *MassJoin* e *MassJoin Merge*. Para este experimento, foram utilizados 10k registros do DBLP, contendo apenas um atributo formado pela concatenação de título e autor. Figura 1(a) mostra o X (tempo máximo de comunicação na rede), o Y (tempo máximo de processamento) e a quantidade de partições geradas P; note que os resultados estão em escala logarítmica. Pode-se observar uma grande superioridade da abordagem mais simples baseada em filtro de prefixo. O tempo total para geração de assinaturas é exibido na Figura 1(b), onde nota-se que o *MassJoin* é significativamente mais custoso computacionalmente. Deve-se ressaltar, contudo, que o *MassJoin* e *MassJoin Merge* foram implementados utilizando tipos inteiros de precisão arbitrária, que por sua vez têm desempenho significativamente inferior a tipos primitivos.

Figuras 1(c) e 1(d) reportam resultados para uma quantidade crescente de processadores. Nota-se que ao alocar mais processadores para a execução do algoritmo, em ambas as bases de dados, o tempo de execução tende a melhorar inicialmente, mas em certo ponto estabiliza e não diminui mais. Isso ocorre pois, em certo ponto, o trabalho gasto com comunicação e gerenciamento distribuído tende a influenciar, até mesmo negativamente, no resultado final. Nas Figuras 1(e), 1(f), 1(i) e 1(j) observa-se que ao se aumentar a quantidade de registros ou de conjuntos a serem processados, naturalmente o tempo de execução também aumentará. Entretanto, pode-se destacar aqui a escalabilidade da solução ao prover um aumento linear do tempo de execução. Como pode ser observado nas Figuras 1(g) e 1(h), a medida que o limiar de similaridade aumenta, o tempo de execução diminui. Isso ocorre pois prefixos menores são gerados e, por consequência, partições menores são produzidas. Figuras 1(i) e 1(j) mostram que o aumento da quantidade de atributos produz um aumento no tempo de processamento, como esperado. Finalmente, Figuras 1(k) e 1(l) mostram os resultados obtidos ao mudar o atributo de assinatura (nos experimentos anteriores fora utilizado o atributo de assinatura de melhor desempenho). Pode-se observar que o atributo de assinatura possui influência significativa no tempo de execução do algoritmo.

5. Conclusões e Trabalhos Futuros

Este artigo apresenta um algoritmo distribuído para junções por similaridade sobre múltiplos atributos utilizando a plataforma *Spark*. Em contraste com trabalhos anteriores, a abordagem proposta permite o emprego de expressões de similaridade complexas. Resultados iniciais são promissores, indicando bom desempenho em todos os cenários analisados. Trabalhos futuros serão concentrados em melhorar particionamento de dados para expressões complexas e explorar recursos de compartilhamento de dados do *Spark*.

Agradecimento Este trabalho foi parcialmente financiado pela CAPES.

Referências

Chaudhuri, S., Ganti, V., and Kaushik, R. (2006). A Primitive Operator for Similarity Joins in Data Cleaning. In *ICDE*, page 5.

Deng, D., Li, G., Hao, S., Wang, J., and Feng, J. (2014). MassJoin: A Mapreduce-based Method for Scalable String Similarity Joins. In *ICDE*, pages 340–351.

Li, G., He, J., Deng, D., and Li, J. (2015). Efficient Similarity Join and Search on Multi-Attribute Data. In *SIGMOD*, pages 1137–1151.

Ribeiro, L. A. and Härder, T. (2011). Generalizing Prefix Filtering to Improve Set Similarity Joins. *Information Systems*, 36(1):62–78.

Sidney, C. F., Mendes, D. S., Ribeiro, L. A., and Härder, T. (2015). Performance Prediction for Set Similarity Joins. In *SAC*, pages 967–972.

Vernica, R., Carey, M. J., and Li, C. (2010). Efficient Parallel Set-similarity Joins using MapReduce. In *SIGMOD*, pages 495–506.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. (2016). Apache Spark: a Unified Engine for Big Data Processing. *Communications of the ACM*, 59(11):56–65.

paper:174670

Uma Análise Comparativa entre Sistemas Gerenciadores de Bancos de Dados NoSQL no contexto de Internet das Coisas

Allexandre Sampaio Santos Soares, Pablo Freire Matos

Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)
Av. Amazonas, 3150, Zabelê – 45.075-265 – Vitória da Conquista – BA – Brasil

{allexandre.soares, pablofmatos}@ifba.edu.br

Abstract. *The search for improvement on applications performance is constant, especially in environments where data is analyzed in real time, as in Internet of Things (IoT). This paper aims to analyze the performance of non-relational Database Management Systems (DBMS), inside the IoT context. To do so, three DBMS where analyzed with a benchmarkimg tool, using a real IoT dataset. The tests evaluated response time, throughput, error rate and the use of CPU, memory and storage. The results showed each DBMS has characteristics that make it be the recommendation to a specific scenario, depending on the application's objective.*

Resumo. *É constante a busca pela melhoria no desempenho de aplicações, principalmente em um ambiente onde dados são analisados em tempo real, como na Internet das Coisas (IoT). Este artigo objetiva analisar o desempenho de sistemas gerenciadores de bancos de dados (SGBDs) não relacionais, no contexto da IoT. Para tanto, três SGBDs foram avaliados por meio de uma ferramenta de benchmarking, utilizando uma base de dados real de IoT. Os testes avaliaram o tempo de resposta, a vazão, a taxa de erros e o consumo de CPU, memória e armazenamento. Os resultados demonstraram que cada SGBD tem características que o faz ser recomendado a um cenário específico, a depender do objetivo da aplicação em questão.*

1. Introdução

Os Bancos de Dados (BDs) têm sido amplamente adotados como solução no armazenamento de dados. Mais especificamente, o modelo de BD relacional se tornou um grande sucesso, sendo implementado por vários desenvolvedores e utilizado na maioria dos projetos de BD ou sistemas computacionais [Sadalage e Fowler 2013]. No entanto, a evolução da computação, o aumento do tráfego de dados e a necessidade de visualização dos dados em tempo real, dentre outros motivos, fizeram com que as aplicações modernas buscassem alternativas de armazenamento de dados que priorizem escalabilidade e velocidade [Sadalage e Fowler 2013].

Como alternativas para evitar esse problema, pode-se verificar o uso de BDs de modelos hierárquico, de rede, orientado a objetos, objeto-relacional, dentre outros. Dentre essas abordagens, destaca-se o modelo NoSQL, do inglês *Not Only SQL*. Os BDs NoSQL têm como características a arquitetura flexível, a alta velocidade de acesso e gravação de dados e escalabilidade, que os tornam muito compatíveis com aplicações de alto desempenho e que priorizam a velocidade de acesso em detrimento à confiabilidade dos dados [Strauch 2011].

Dentre as diversas aplicações que podem se beneficiar da utilização de BDs NoSQL, estão aquelas ligadas à Internet das Coisas (IoT – *Internet of Things*). Geralmente, essas aplicações, por funcionarem em tempo real, necessitam de grande velocidade de manipulação de dados, uma vez que nesse ambiente há uma grande quantidade de dados de múltiplas fontes sendo trafegados [Porto, Ziviani e Ogasawara 2015].

A análise de Sistemas Gerenciadores de Bancos de Dados (SGBDs) é uma prática muito difundida, utilizada para se medir a aplicabilidade desses a um caso específico ou para se medir o desempenho entre diversos SGBDs para um determinado propósito. Apesar de existirem alguns trabalhos que enfoquem a utilização de SGBDs NoSQL no contexto de IoT, ainda são poucos os que tratam da análise do desempenho desses.

Pinto *et al*. (2013) realizaram um comparativo entre os SGBDs NoSQL MongoDB e DB4O, buscando compreender o comportamento desses quando submetidos a grandes cargas de dados. Uma aplicação foi desenvolvida especificamente para a realização do experimento, sendo possível especificar a quantidade de registros a serem inseridos no BD. Os resultados do trabalho sugerem o uso de um ou outro SGBD de acordo com o foco da aplicação em questão, seja desempenho ou economia de recursos. Carniel *et al.* (2012), por sua vez, compararam alguns BDs NoSQL orientados a colunas e documentos com o BD relacional PostgreSQL no contexto de processamento de consultas sobre Data Warehouse. Os resultados alcançados demonstraram superioridade do SGBD orientado a colunas, em comparação com os demais modelos, no contexto estudado.

Oliveira, Lóscio e Gama (2015) e Tavares, Oliveira e Lóscio (2016) têm como objeto de estudo a utilização de SGDBs NoSQL como forma de armazenamento de catálogos de produtores e consumidores de dados de IoT. O primeiro trabalho realizou experimentos de modo a verificar a viabilidade no uso do SensorML e do MongoDB com esta finalidade. O segundo desenvolveu diferentes protótipos de catálogos para adequação aos SGBDs OrientDB e MongoDB, de forma a verificar se esses são opções válidas para o contexto tratado, por meio do uso do SensorML e do WoTDataSchema. Ambos os estudos puderam demonstrar, por meio dos experimentos realizados com base em várias simulações, que há viabilidade no uso destes SGBDs nos contextos estudados. O presente estudo se assemelha com esses trabalhos, uma vez que trata de testes de desempenho aplicados a contextos bastante similares.

A partir do exposto, percebe-se uma lacuna de trabalhos focados no uso de SGBDs NoSQL dentro do contexto de IoT. Portanto, este artigo objetiva identificar dados comumente utilizados em aplicações de IoT e povoar BDs NoSQL com esses dados, possibilitando análises focadas no desempenho, integridade e consumo de recursos, de maneira que se possa traçar um comparativo. Para a realização do experimento proposto, foi desenvolvida uma ferramenta específica para *benchmarking*, a qual é descrita neste trabalho. O artigo está estruturado da seguinte forma: Na Seção 2, é apresentada a fundamentação teórica necessária para o entendimento deste trabalho; na Seção 3, são descritas as definições do ambiente de execução dos testes, bem como a ferramenta para benchmarking e a base de dados utilizada; Na Seção 4, são descritos os resultados sucintos alcançados com o experimento; e na Seção 5, são apresentadas a Conclusão e os Trabalhos Futuros.

2. Fundamentação Teórica

O embasamento teórico deste trabalho fundamenta-se, principalmente, nos conceitos de BDs NoSQL, IoT e técnicas de *benchmarking*.

2.1. Sistemas de Bancos de Dados NoSQL

Os BDs relacionais podem ser empregados na maioria das aplicações, tendo muitas vantagens em sua utilização. No entanto, um problema recorrente em seu uso é a incompatibilidade de impedância. Segundo Sadalage e Fowler (2013), a incompatibilidade de impedância ocorre quando há uma diferença entre o modelo de dados relacional e as estruturas dos dados na memória. Os BDs NoSQL são, então, mais adequados a aplicações e sistemas que utilizam dados não estruturados e que necessitam de grande desempenho em escalabilidade e no acesso aos dados, como sistemas que funcionam em tempo real. Nesse modelo, os bancos priorizam a disponibilidade dos dados em detrimento à consistência absoluta desses.

Vieira *et al*. (2012, p. 3) resumem as características dos BDs NoSQL em quatro propriedades básicas: "(i) dados na ordem de dezenas ou centenas de Terabytes; (ii) poder de

crescimento elástico horizontal; (iii) fácil distribuição dos dados e/ou processamento; e (iv) tipos de dados variados, complexos e/ou semiestruturados". Essas características tornam os BDs NoSQL recomendáveis para cenários como o da Internet das Coisas, que demanda de alto poder de processamento de dados não estruturados e de grande disponibilidade.

2.2. Internet das Coisas

Segundo Almeida (2015, p.7), a IoT "refere-se à integração de objetos físicos e virtuais em redes conectadas à Internet, permitindo que "coisas" coletem, troquem e armazenem uma enorme quantidade de dados numa nuvem, em que uma vez processados e analisados esses dados, gerem informações e serviços em escala inimaginável".

Sobre os dados das aplicações de IoT, Li *et al*. (2012) evidenciam que essas têm seus dados como ativos de valor. Tendo esses as seguintes características: (i) provém de várias fontes e são heterogêneos; (ii) têm grande escala; (iii) têm correlação temporal e espacial; (iv) necessitam de interoperabilidade; e (v) são de multidimensionalidade. Resumidamente, essas características mostram que os dados destas aplicações surgem de forma contínua e expansiva de diversas fontes (sensores, câmeras, leitores, dentre outros), precisam de referências de tempo, já que as suas informações são geralmente tratadas em tempo real e devem trabalhar independentemente de plataformas, sendo seus dados acessados por diferentes aplicações.

2.3. Testes de Desempenho

Segundo Santos e Neto (2008), os testes de desempenho verificam em um sistema: latência, vazão, escalabilidade e uso de recursos de máquina. Para a realização de diversos tipos de testes, comumente são utilizadas várias ferramentas, focadas ou especializadas em uma característica específica a ser testada em um sistema computacional. Tais ferramentas são conhecidas como ferramentas de *benchmarking*.

Existem diversas ferramentas disponíveis para avaliar *softwares*, SGBDs, elementos de *hardware*, dentre outros. No entanto, como citado por Gray (1993), nenhuma das ferramentas poderia testar o desempenho de um sistema por completo, havendo então a necessidade de se testar um sistema utilizando ferramentas adequadas ao seu domínio de aplicação.

3. Experimento

Para a realização do experimento foram analisadas diversas características de SGBDs e ferramentas para testes, com o intuito de realizar uma simulação tão próxima quanto possível da realidade quanto ao uso de aplicações de IoT integradas a BDs NoSQL. Com o intuito de prezar ao máximo pelo correto funcionamento dos SGBDs durante a realização dos testes, bem como garantir que todos os SGBDs testados estivessem sob o mesmo ambiente com a mínima variação possível, optou-se pela utilização de uma partição de disco do computador (com 17,7 GB de capacidade), com um Sistema Operacional (SO) dedicado apenas aos testes, abrigando tanto o *benchmarking* utilizado, quanto os SGBDs testados. O computador utilizado na realização dos testes foi um Notebook Acer Aspire V5 - 472-6_BR826, contando com 8 GB de memória RAM, 500 GB de armazenamento de disco rígido e processador Intel i3-3217U, que dispõe de 2 núcleos físicos e 4 *threads* com frequência base de 1,8 GHz. O SO utilizado foi o Ubuntu 16.04.1. Tal escolha se deu por ter esse SO livre disponibilidade, por ser compatível com todos os SGBDs testados e por sua grande comunidade, que auxilia na resolução de problemas. Durante a realização dos testes, optou-se pelo uso do SO apenas via linha de comando de modo a economizar recursos que seriam utilizados por sua interface gráfica.

Dada a necessidade da automatização da tarefa de submeter objeto testado a diversas cargas e condições de testes, fez-se necessário o uso de ferramentas de *benchmarking*. Após a análise das ferramentas elencadas com tal finalidade, percebeu-se uma dificuldade em utilizá-las para um contexto específico, nesse caso, o de IoT. Optou-se então pelo desenvolvimento de uma ferramenta específica para o foco deste trabalho e moldada de forma que fosse o mais simples e eficiente possível. O *software* desenvolvido se chama Inobench, uma referência à sua

finalidade (IoT-NoSQL-Benchmarking), e está disponível para livre uso sob a licença GNU General Public License V.3 em seu repositório na Internet[1]. A ferramenta em questão é composta de alguns módulos: *core* (controle geral da aplicação), *CSVReader* (leitura de arquivos externos), *files* (armazenamento e manipulação de arquivos externos), além de um módulo específico para cada um dos SGBDs abrangidos pela solução (implementação de operações para a manipulação de dados em cada SGBD).

Os BDs utilizados no experimento foram de documentos e de chave-valor. Tal escolha se deu pelo entendimento de que esses modelos de BDs permitem uma manipulação mais simples dos dados utilizados, tornando seu armazenamento mais fácil. Foram escolhidos três SGBDs: MongoDB e Couchbase para os testes com SGBDs de documentos, e Redis para os testes com SGBDs de chave-valor. Os critérios que levaram à escolha desses foram sua compatibilidade com o ambiente de teste, a possibilidade de sua utilização de forma gratuita, sua adequação à proposta deste trabalho e a sua facilidade de uso a partir da experiência do agente realizador do experimento. Durante a realização dos testes, buscou-se utilizar a versão mais recente de cada SGBD. Sendo assim, o MongoDB foi utilizado em sua versão 3.2.12, o Couchbase em sua versão 4.6.0 e o Redis em sua versão 3.2.8. É importante salientar que, em todos os SGBDs, não foram utilizadas técnicas de replicação. Sendo assim, os resultados alcançados são baseados no desempenho relativo a apenas uma instância de cada SGBD.

Para que o experimento proposto simulasse de forma fiel o ambiente real, optou-se pela utilização de uma base de dados com informações colhidas de aplicações de IoT. Assim, foi utilizada a base de dados *Air Quality Data Set*, disponibilizada publicamente no repositório[2] de aprendizagem de máquina da Universidade da Califórnia, Irvine. A base de dados em questão contém 9.358 instâncias de dados colhidos de diversos sensores que medem quimicamente a qualidade do ar. Cada instância corresponde à média dos dados colhidos dentro de uma hora, durante os anos de 2004 e 2005, em ambientes de grande concentração de poluição em uma cidade na Itália. Cada instância da base de dados contém 15 dados diferentes. Os dados são relativos à data da coleta, hora da coleta, concentração de monóxido de carbono, concentração de dióxido de estanho, concentração de benzeno, temperatura, umidade, dentre outros.

No que tange à metodologia de teste empregada, foram realizadas múltiplas operações de inserção e consulta, utilizando-se de um número crescente de usuários, com o intuito de verificar os limites máximos de operação de cada um dos SGBDs. Cada ciclo de teste foi iniciado com 1 usuário realizando 100.000 operações, subindo ao nível de 10 usuários simultâneos realizando 10.000 operações, 100 usuários realizando 5.000 operações, 500 usuários realizando 1.000 operações e finalizando-se com 750 usuários realizando 100 operações. A quantidade máxima foi definida após a observação de limites que pudessem garantir a confiabilidade dos resultados, visto que os resultados foram comprometidos pela sobrecarga em um cenário com cerca de 1.000 usuários. Nos ciclos de teste, avaliou-se: tempo de resposta; vazão; taxa de erros; consumo de CPU e memória; e espaço de armazenamento utilizado (para operações de inserção).

4. Resultados

A análise dos resultados obtidos no experimento realizado possibilita perceber a superioridade de um SGBD no cenário testado para cada uma das métricas definidas. Assim sendo, as conclusões tomadas neste trabalho são baseadas inicialmente nos fatores individuais que fizeram com que cada um dos SGBDs testados se mostrasse superior, fatores que devem, portanto, ser analisados para a escolha de um desses SGBDs para uso em ambiente real de IoT.

Tratando-se de vazão e tempo de resposta, os resultados demonstraram que os SGBDs testados têm resultados parecidos na inserção de registros, tendo o Redis alcançado resultados

[1] Inobench: https://github.com/allexandresampaio/inobench

[2] Air Quality Data Set: http://archive.ics.uci.edu/ml/datasets/Air+Quality

levemente superiores aos demais. Para a leitura de dados, o Couchbase foi mais rápido. Sobre as taxas de erros, verifica-se que estes ocorrem com maior frequência quando se inserem mais usuários simultâneos acessando os SGBDs. No entanto, a ocorrência de erros representa valores muito pequenos dentre o total de operações realizadas. Nesses cenários, o MongoDB obteve resultados exitosos nos testes de inserção e consulta, e o Redis nos testes de consultas.

A análise do consumo de CPU e memória RAM demonstra que, na maioria dos casos, os SGBDs testados não demandam de muitos recursos para realizar as operações com eficiência. Sobre o consumo de CPU, nos testes de inserção de dados, tanto o Couchbase quanto o Redis têm resultados semelhantes, sendo os mais econômicos. Para leitura de dados, o Couchbase consumiu menos processamento. Sobre o consumo de memória RAM, nenhum dos SGBDs chegou a usar sequer 10% do total de memória RAM disponível. Tanto nas inserções, quanto nas consultas de registros, o Redis se mostrou superior em relação aos demais SGBDs.

A análise de consumo de memória física para guardar dados nos SGBDs é importante visto que, no ambiente de IoT, a quantidade de dados armazenados é consideravelmente grande e crescente. Neste quesito, os testes evidenciaram superioridade do MongoDB, tendo os demais SGBDs consumido várias vezes mais espaço do que o utilizado por esse.

Na Figura 1 é exibida em formato de gráfico de radar a adequação dos SGBDs testados às métricas avaliadas. Cada vértice das linhas que representam os SGBDs está posicionado de acordo com uma escala que, do centro para fora do gráfico, vai de zero a cem. O uso deste gráfico tem como objetivo demonstrar visualmente qual dos SGBDs testados mostra-se mais recomendado ao uso de acordo com apenas uma ou por combinação de métricas. Desta forma, é fácil perceber e possível inferir que: o MongoDB é recomendado para aplicações onde há restrições de confiabilidade e de espaço para armazenamento em disco, além de ter um resultado mediano nas demais métricas; o Couchbase é recomendado para aplicações que demandam de alta velocidade; e o Redis é recomendado para aplicações onde há restrições de consumo de processamento, tendo um resultado satisfatório nas demais métricas avaliadas.

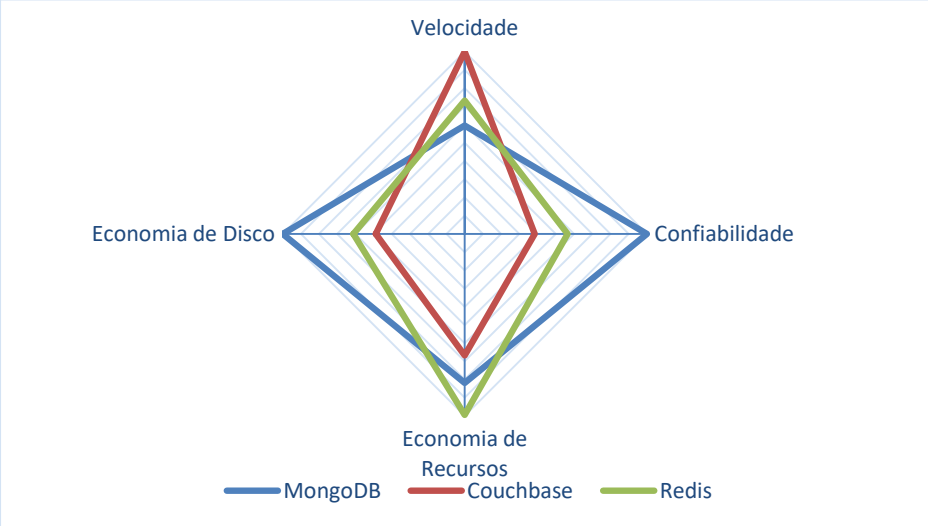


Figura 1. Gráfico Radar de métricas avaliadas. Fonte: Próprio autor

Sobre a realização dos testes, faz-se necessário frisar que, como a aplicação utilizada foi desenvolvida em Java, houve um consumo de recursos pela máquina virtual da aplicação que pode ter influenciado nos resultados. No entanto, considera-se que os resultados obtidos são válidos, uma vez que a possível influência da aplicação ocorreu para todos os testes, igualmente, independentemente do SGBD utilizado ou cenário testado.

5. Conclusão

O objetivo proposto por este trabalho foi de desenvolver uma análise comparativa entre SGBDs NoSQL quando inseridos no contexto da IoT. Para tanto, desenvolveu-se uma ferramenta capaz de realizar testes em SGBDs NoSQL utilizando bases de dados predefinidas. Durante os testes, as principais variáveis que definem o bom desempenho ou não de um determinado SGBD foram colhidas. Isso possibilitou identificar a superioridade de um determinado SGBD em situações específicas no ambiente da IoT, sendo essa superioridade condicionada ao contexto da aplicação que fará uso do SGBD. Portanto, as necessidades da aplicação devem ser avaliadas para a escolha do SGBD que melhor se adapte a elas.

Apesar de existirem alguns trabalhos correlatos que compararam SGBDs relacionais e não-relacionais, observa-se que este trabalho tem como diferencial a realização de um *benchmarking* aplicado a um contexto específico, por meio de testes de desempenho. Pode-se destacar como principais contribuições deste estudo o desenvolvimento de um *software* voltado para o *benchmarking* de SGBDs NoSQL, sendo possível contextualizar tais testes a partir da utilização de uma base de dados específica. Além disso, considera-se de grande contribuição os resultados obtidos no experimento, podendo ser utilizados como auxílio na análise da utilização de um dos SGBDs NoSQL testados em uma aplicação no contexto da IoT. Por fim, recomenda-se como trabalhos futuros a realização de testes de escalabilidade dos SGBDs NoSQL, bem como a utilização de SGBDs NoSQL temporais (*time-series*).

Referências

Almeida, H. (2015) Tudo Conectado. *Computação Brasil: Revista da Sociedade Brasileira de Computação*, Porto Alegre, v. 29, 14-18.

Carniel, A. C., Sá, A. A., Ribeiro, M. X., Bueno, R., Ciferri, C. D. A. e Ciferri, R. R. (2012) Análise Experimental de Bases de Dados Relacionais e NoSQL no Processamento de Consultas sobre Data Warehouse. In: *XXVII SBBD*, pages 113-120.

Gray, J. (1993) *The Benchmark Handbook for Database and Transaction Systems*. 2ª ed. Morgan Kaufmann Publishers.

Li, T., Liu, Y., Tian, Y., Shen, S. e Mao, W. (2012) A storage solution for massive IoT data based on NoSQL In: *GreenCom*, pages 50-57. IEEE.

Oliveira, M. I. S., Lóscio, B. F. e Gama, K. S. (2015) "Análise de Desempenho de Catálogo de Produtores de Dados para Internet das Coisas baseado em SensorML e NoSQL." In: XIV Workshop em Desempenho de Sistemas Computacionais e de Comunicação.

Pinto, A. P., Malagoli, F., Maciel, R. M., Soares, H. R. e Santana, S. A. (2013) "Testes de Performance Utilizando o DB4O e MongoDB" In: e-RAC, v. 3, n. 1.

Porto, F., Ziviani, A. e Ogasawara, E. (2015) BIG DATA: Um Novo Desafio à Nossa Porta. *Computação Brasil: RSBC*, Porto Alegre, v. 29, 23-27.

Sadalage, P. e Fowler, M. (2013) *NoSQL Essencial*: Um guia conciso para o mundo emergente da persistência poliglota. São Paulo: Novatec.

Santos, I. e Neto, P. (2008) Automação de Testes de Desempenho e Estresse com o JMeter. In: *II ERCEMAPI*.

Strauch, C. (2011) "NoSQL databases" In: Lecture Notes, Stuttgart Media University.

Tavares, A. T., Oliveira, M. I. S. e Lóscio, B. F. (2016) "Data producer catalogs for the web of things: a study on NoSQL solutions." In: 31st Annual ACM Symposium on Applied Computing.

Vieira, M., Figueiredo, J., Liberatti, G. e Viebrantz, A. (2012) Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data. In: *XXVII SBBD*, pages 439-468.

paper:174079

Uso de instâncias de dados e carga de trabalho para mineração de restrições de integridade

Eduardo Henrique Monteiro Pena[1,2], **Eduardo Cunha de Almeida**[2]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)

[2]Universidade Federal do Paraná (UFPR)

eduardopena@utfpr.edu.br,eduardo@inf.ufpr.br

Resumo. *Dependências funcionais (DFs) representam restrições de integridade amplamente estudadas no contexto de caracterização de dados. Neste trabalho, exploramos a descoberta automática de DFs e descrevemos um método para seleção daquelas que são relevantes com relação a semântica da carga de trabalho. A avaliação experimental mostra que as dependências selecionadas exibem propriedades expressivas comparadas ao espaço de busca, o que demonstra a efetividade da abordagem apresentada.*

Abstract. *Functional dependencies (FDs) are integrity constraints widely studied in the context of data profiling. In this work, we explore the automatic discovery of FDs and describe a method for selecting relevant ones regarding workload semantics. The experimental evaluation shows that the selected dependencies exhibit expressive properties compared to the search space, which demonstrates the effectiveness of the presented approach.*

1. Introdução

Relações de dependência entre atributos amparam inúmeras atividades do gerenciamento e caracterização de dados, dentre elas, otimização de consultas, integração de dados e limpeza de dados [Liu et al. 2012]. Infelizmente, o desenvolvimento de mecanismos que auxiliem na exposição de meta-informações relevantes se torna bastante desafiador uma vez que as organizações modernas apresentam constantes evoluções nos seus padrões de provisão e massificação de dados [Abedjan et al. 2015].

Uma forma de dependência bastante conhecida é a dependência funcional (DF) [Beeri et al. 1984]. Normalmente, DFs são definidas como restrições de integridade (RIs) durante o projeto e normalização de bancos de dados relacionais. No entanto, em decorrência de evoluções nas aplicações e dados, a manutenção supervisionada de novas restrições se torna uma atividade sujeita a erros, que pode muitas vezes ser deixada de lado em bancos de dados não normalizados (e.g., armazém de dados). Ademais, tarefas que recorrem à DFs em algum nível não deveriam estar limitadas às DFs estaticamente definidas por projetistas. De outro modo, deveriam explorar a dinâmica das instâncias de dados e aplicações a fim de colaborar com aquelas que apresentam maior riqueza semântica e funcional.

Devido à importância do tema, diversos algoritmos para descoberta automática de DFs foram propostos [Abedjan et al. 2015]. O processo de descoberta apresenta dois desafios proeminentes. Em primeiro lugar, a complexidade de encontrar todas as DFs

mantidas em uma instância de relação com n registros está em $\mathcal{O}(n^2(\frac{m}{2})^2 2^m)$ para uma relação com m atributos [Liu et al. 2012]. Em segundo lugar, as dependências retornadas por algoritmos de descoberta podem se tornar numerosas, ao ponto de dificultarem sua utilização em cenários reais. Um limite superior de $2^m - 1$ é considerado uma vez que as possibilidades de DFs mantidas em uma relação com m atributos é um resultado combinatório [Liu et al. 2012]. Infelizmente, muitos trabalhos desenvolvem técnicas que cooperam com a descoberta eficiente de DFs, mas deixam de lado o domínio em que as mesmas serão aplicadas [Papenbrock et al. 2015]. Por sua vez, trabalhos com domínio específico negligenciam a descoberta automática e exploram, exclusivamente, DFs definidas manualmente em suas análises experimentais [Abedjan et al. 2015].

Nossa hipótese é que é possível extrair informação semântica das cargas de trabalho para seleção de RIs relevantes. Embora diversos critérios possam ser considerados para representação de tal informação [Chaudhuri et al. 2003], estamos particularmente interessados nos atributos acessados por um conjunto histórico de consultas relacionais em suas operações de seleção, projeção e agrupamento. Assim, manipulamos a estrutura semântica exposta pela combinação de DFs e conjuntos de consultas, e formalizamos um método baseado no agrupamento por propagação de afinidade capaz de selecionar restrições que melhor representam o espaço de busca com relação ao uso da aplicação. Em nossa avaliação experimental, o método de seleção contribuiu com a redução do grande número de DFs descobertas em instâncias de dados reais. Além disso, as DFs selecionadas apresentaram níveis de correlação com a carga de trabalho aprimorados (com relação ao espaço de busca), demonstrando que a seleção de DFs pode auxiliar, ou até mesmo substituir, a intervenção humana na configuração de RIs.

2. Trabalhos relacionados

Dependências funcionais encontram espaço em diversos cenários. Sobre caminhos de acesso, um índice secundário sobre um atributo pode ser beneficiado se é funcionalmente dependente de atributos que dispõem de índices agrupados [Kimura et al. 2009]. O relacionamento semântico entre especificações de ordenação em planos de consultas é investigado com auxílio de DFs em [Szlichta et al. 2013]. No processo de integração de dados, é comum que dados provenientes de diferentes fontes produzam inconsistências e violem diversas RIs. Nesse contexto, um estudo sobre a relação entre a colaboração coletiva em atividades de limpeza e a qualidade final dos dados (violações de DFs são métricas avaliadas) é apresentado em [Chung et al. 2017]. De modo geral, o processo de limpeza de dados envolve a detecção de inconsistências e, possivelmente, reparo do banco de dados para que um conjunto de regras de qualidade seja satisfeito. Um modelo para limpeza de dados baseado no reparo de DFs, por meio da modificação de dados, é apresentado por [Bohannon et al. 2005]. Como as regras mantidas pela aplicação evoluem continuamente, a limpeza também pode ser feita pelo reparo das próprias DFs [Mazuran et al. 2016].

Um resultado direto do método descrito neste trabalho é a redução do custo de reparos de RIs, justamente porque o número de DFs consideradas após o método de seleção é muito menor. Reparar uma instância r em R significa encontrar um r' que esteja em conformidade com um conjunto Σ. Devido a natureza NP-completo do problema, modelos de custo e heurísticas são utilizados, trocando a busca do menor custo pela efetividade dos reparos. Em [Bohannon et al. 2005] uma heurística com tempo de execução em $\mathcal{O}(n \lg^2 n \cdot |\Sigma|^2)$ é descrita para relações com n registros e $|\Sigma|$ DFs. Complexida-

des análogas são mantidas por outras abordagens [Abedjan et al. 2015], o que indica a existência de cenários potencialmente beneficiados pelo método de seleção.

3. Seleção automatizada de dependências funcionais

3.1. Definições

Considere uma relação de esquema $R(A_1, ..., A_n)$ e r uma instância de R sobre os domínios de atributo $dom(A_1), ..., dom(A_n)$. Dados dois conjuntos de atributos X e Y, $X \subseteq R$ e $Y \subseteq R$, denotamos por $t_i[X]$ e $t_i[Y]$ a projeção da tupla t_i em X e Y, respectivamente. Uma DF sobre R tem a forma $f : X \rightarrow Y$, e requer que X determine funcionalmente Y. Uma DF se mantém em r, denotamos por $r \models f : X \rightarrow Y$, se $\forall t_i, t_j \in r$, $t_i[X] = t_j[X]$ então $t_i[Y] = t_j[Y]$. Consideramos X como o *left-hand side* (*lhs*) de f e Y como o *right-hand side* (*rhs*). Um conjunto de DFs Σ é não-trivial e mínimo se $\forall f \in \Sigma$, então f tem um único atributo em *rhs*, não possui atributo redundante (i.e. $Y \not\subseteq X$), e não existe Z tal que $(X - Z) \rightarrow Y$ seja uma DF válida.

3.2. Descoberta automatizada

A primeira etapa da abordagem proposta é a descoberta automática de todas as DFs mantidas em r. Uma estudo e avaliação experimental dos principais algoritmos de descoberta é visto em [Papenbrock et al. 2015]. Em nossa implementação, utilizamos o algoritmo HyFD [Papenbrock and Naumann 2016] que, atualmente, apresenta os melhores resultados em termos de tempo de execução e escalabilidade. A entrada do algoritmo é uma instância r e sua saída é o conjunto de todas DFs não-triviais e mínimas em r. De modo geral, os componentes do HyFD são estruturados para desenvolverem otimizações em nível de tuplas e colunas combinadas com técnicas de amostragem e compressão, permitindo que o algoritmo escale para grandes conjuntos de dados.

3.3. Combinando DFs e carga de trabalho

Com base na hipótese discutida na introdução, combinamos a informação semântica das DFs retornadas pela execução do algoritmo HyFD com os padrões de carga de trabalho a fim de identificar atributos de interesse. Primeiro, calculamos a frequência em que atributos são acessados por DFs e consultas. Considere uma relação $R(A_1, ..., A_n)$ e uma coleção de conjuntos $S = \{s_1, ..., s_m\}$ tal que $s_i \subseteq R$. Cada s_i representa uma lista de atributos relativos a R, extraídos diretamente dos operadores relacionais (seleção, projeção e agrupamento) de uma consulta ou da estrutura *lhs* de uma DF. Para cada $s_i \in S$, e para cada $A_j \in R$, um valor de ocorrência é atribuído como na Função 1:

$$o_{ij} = \begin{cases} 1 & \text{se } s_i \text{ possui atributo referenciando } A_j \\ 0 & \text{caso contrário.} \end{cases} \quad (1)$$

As entradas em o_{ij} formam uma matriz de ocorrência de atributos (MOA), denotada por O, e indicam atributos em R tocados por elementos em s_i. Estabelecemos duas MOAs: para um conjunto de consultas (O^q); e para um conjunto de DFs, sobre suas estruturas *lhs* (O^{lhs}). Utilizamos apenas a estrutura *lhs* pois empregamos regras de inferência [Beeri et al. 1984] sobre o conjunto Σ para obter DFs estendidas (agrupamento pelo mesmo *lhs*). Ademais, considere o vetor resultante da soma das linhas em qualquer O, calculado com $\sum_j^m o_{ij}$ e denotado por $\rho(O)$. Se assumirmos duas MOAs, O e O',

é possível ponderar o número de acessos aos atributos de R com $\mathbf{O} = \sum_i^m O(i)\rho(O')$. Estamos interessados em ponderar as DFs de acordo com os padrões de acesso na carga de trabalho, assim, assumimos $O = O^{lhs}$ e $O' = O^q$.

3.4. Agrupando DFs por meio da propagação de afinidade

Cada vetor $\mathbf{o}_i$ em $\mathbf{O}$ representa a estrutura de uma DF ponderada pela sua relação com a carga de trabalho. É possível utilizar medidas de distância para estimar a proximidade entre pares dessas estruturas e agrupá-las segundo sua semântica e densidade. Em experimentos preliminares, medidas clássicas como coeficientes de correlação e distância Euclidiana produziram resultados, em grande parte, inconclusivos. No entanto, percebemos que a distância de *Mahalanobis* (DM) funciona com precisão sobre o esquema proposto. Dados $\mathbf{o}_i$ e $\mathbf{o}_j$, $i \neq j$, estimamos a distância de *Mahalanobis* com $dm(\mathbf{o}_i, \mathbf{o}_j) = \sqrt{(\mathbf{o}_i - \mathbf{o}_j)V^{-1}(\mathbf{o}_i - \mathbf{o}_j)^T}$, onde V^{-1} é a inversa da matriz de covariância.

Nossa estratégia de seleção de DFs utiliza o algoritmo de agrupamento por propagação de afinidade (APA) [Frey and Dueck 2007]. A vantagem da APA sobre algoritmos clássicos de agrupamento (e.g, *K-means*) é que ela não exige que o número de *clusters* seja especificado a priori. Assim, dois tipos de mensagens são trocadas recursivamente entre pares de DFs até que um critério de parada seja atingido (qualidade das mensagens estáveis por um número de iterações). A primeira é chamada responsabilidade, calculada com $r(\mathbf{o}_i, \mathbf{o}_j) \leftarrow dm(\mathbf{o}_i, \mathbf{o}_j) - \max_{\forall \mathbf{o}'_j \neq \mathbf{o}_j} \left\{\alpha(\mathbf{o}_i, \mathbf{o}'_j) + dm(\mathbf{o}_i, \mathbf{o}'_j)\right\}$. A responsabilidade $r(\mathbf{o}_i, \mathbf{o}_j)$ mede a evidência acumulada que $\mathbf{o}_j$ deva representar $\mathbf{o}_i$, considerando a disponibilidade α de $\mathbf{o}_j$ sobre $\mathbf{o}_i$, a qual é calculada com $\alpha(\mathbf{o}_i, \mathbf{o}_j) \leftarrow \min\left\{0, r(\mathbf{o}_j, \mathbf{o}_j) + \sum_{\mathbf{o}'_i \text{ tal que } \mathbf{o}'_i \notin \{\mathbf{o}_i, \mathbf{o}_j\}} \max\{0, r(\mathbf{o}'_i, \mathbf{o}_j)\}\right\}$. As disponibilidades entre pares de DFs são nulas na primeira iteração. Assim, cada $r(\mathbf{o}_i, \mathbf{o}_j)$ assume a diferença entre $dm(\mathbf{o}_i, \mathbf{o}_j)$ e a maior DM entre o ponto $\mathbf{o}_i$ e os demais candidatos. A disponibilidade $\alpha(\mathbf{o}_i, \mathbf{o}_j)$ é o resultado da auto-responsabilidade $r(\mathbf{o}_j, \mathbf{o}_j)$ somada as responsabilidades que o_j recebe de outros pontos. Os elementos que acumulam maiores níveis de disponibilidade e responsabilidade são chamados exemplares, a saída da seleção de DFs. Para traduzir cada $\mathbf{o}_i$ em sua *lhs* correspondente, extraímos os atributos A_j's equivalentes em R para os elementos diferentes de zero.

4. Avaliação experimental

Avaliamos a descoberta e seleção de DFs sobre dois conjuntos de dados da UCI *Machine Learning* (*Abalone* e *Adults*) [1], a relação *lineitem* do *benchmarck* TPC-H (Fator 1GB) [2], e duas semanas de dados provenientes do Sistema Integrado de Monitoramento do Ministério da Ciência, Tecnologia, Inovações e Comunicações (SIMMC) [3]. Os detalhes de cada relação são mostrados na Tabela 1. Nosso protótipo foi implementado como um cliente em linguagem Java conectado a uma instância de banco de dados PostgreSQL. Para a definição da carga de trabalho geramos mil consultas do tipo seleção-projeção-junção com agrupamento. Tal estrutura de consulta é suficiente para expor os parâmetros considerados em nossa abordagem (i.e, frequência de atributos). A escolha dos atributos seguem distribuições enviesadas (*Zipf*) e os valores dos predicados são escolhidos a partir

[1] http://archive.ics.uci.edu/ml/

[2] http://www.tpc.org/tpch/

[3] http://simmc.c3sl.ufpr.br/

Tabela 1. Conjuntos de dados e quantidade de DFs descobertas/selecionadas.

Conjunto de dados	nº de colunas	nº de registros	nº de DFs (HyFD)	nº de DFs após agrupamento
Abalone	9	4.177	137	10
Adults	14	48.842	78	5
Lineitem	16	6 milhões	4 mil	23
SIMMC	12	2 milhões	32	8

da divisão do domínio de cada atributo de forma igualmente espaçada. Finalmente, a qualidade das DFs são avaliadas com suas DMs em relação a carga de trabalho ($\rho(O^q)$) e seu nível de desconfiança, calculada com $d = \sqrt{\left|\frac{|\pi_X(r)|}{|r|} - \frac{|\pi_Y(r)|}{|r|}\right|^2}$. Os níveis de desconfiança (adaptada de [Liu et al. 2012]) medem o nível de redundância de uma DF. Assim, a probabilidade de uma DF se manter por coincidência (desconfiança) diminui conforme as cardinalidades das projeções de *lhs* e *rhs* se aproximam no número de duplicatas. Não utilizamos tal medida como critério de seleção pois se relaciona à instância dos dados e não à carga de trabalho, além de envolver elevado custo computacional.

As quantidades de DFs retornadas pelo algoritmo HyFD e pela APA, respectivamente, são mostradas na Tabela 1. Comparamos as distribuições dos níveis de desconfiança (Figura 1) e DM com relação a carga de trabalho (Figura 2), mostradas como diagramas de extremos e quartis, para verificar as dispersões decorrentes da seleção. Conforme esperado, o agrupamento reduziu o número de DFs significativamente: em 4 vezes na menor proporção (SIMMC); e em 173,91 vezes na maior proporção (*lineitem*). Notavelmente, as distribuições dos níveis de desconfiança se mantiveram estáveis ou melhoraram (quartis centrais em *Adults* se direcionaram a valores menores). Isso é um reflexo direto da APA, pois DFs com estruturas semelhantes usualmente compartilham níveis de desconfiança próximos. As distribuições das DMs mostraram variações mais marcantes. Com exceção de SIMMC, as distribuições da seleção se estendem por valores muito menores, comparadas à distribuição das DFs iniciais. As estruturas *lhs* de SIMMC são semelhantes, acarretando variações pouco perceptíveis nas suas distribuições.

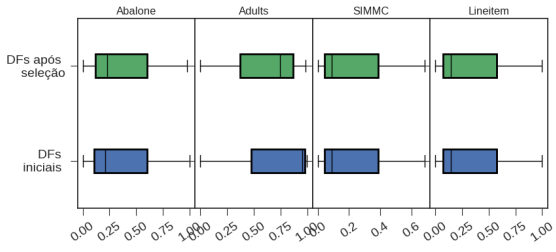


Figura 1. Níveis de desconfiança

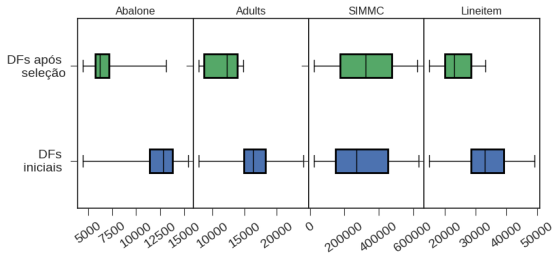


Figura 2. Distância de *Mahalanobis*

A intuição básica por trás da melhora nos níveis de DMs é que o conjunto inicial de DFs apresenta diferentes níveis de correlação com a carga de trabalho. Em suas primeiras iterações, a APA considera qualquer DF como possível exemplar, propagando mensagens uniformes e pouco representativas. Os exemplares são configurados conforme grupos de DFs, que concentrando maior equivalência semântica, avaliam seus níveis de responsabilidade e disponibilidade. Foi crucial a utilização da distância DM uma vez que

considera variâncias desiguais sobre as estruturas ponderadas de pares de DFs. Assim, os exemplares selecionados representam semântica de um grupo de DFs relacionados e, ao mesmo tempo, conferem distâncias reduzidas com relação a carga de trabalho.

5. Conclusão

Neste trabalho observamos a importância da seleção automática de DFs baseada na correspondência entre instância de dados e carga de trabalho. Ponderamos a semântica exposta pela carga de trabalho sobre as estruturas de DFs descobertas nas instâncias de dados e formalizamos um método de agrupamento para seleção de exemplares. As propriedades de correlação com a carga de trabalho e níveis de desconfiança das DFs selecionadas melhoraram com relação aos resultados da descoberta exaustiva, o que demonstra a efetividade da seleção. Trabalhos futuros incluem a integração do método descrito com a limpeza de dados baseada em reparos de RIs.

Referências

Abedjan, Z., Golab, L., and Naumann, F. (2015). Profiling relational data: A survey. *The VLDB Journal*, 24(4):557–581.

Beeri, C., Dowd, M., Fagin, R., and Statman, R. (1984). On the structure of armstrong relations for functional dependencies. *J. ACM*, 31(1):30–46.

Bohannon, P., Fan, W., Flaster, M., and Rastogi, R. (2005). A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD*, 2005, pages 143–154.

Chaudhuri, S., Ganesan, P., and Narasayya, V. (2003). Primitives for workload summarization and implications for sql. In *VLDB*.

Chung, Y., Krishnan, S., and Kraska, T. (2017). A data quality metric (DQM): how to estimate the number of undetected errors in data sets. *PVLDB*, 10(10):1094–1105.

Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *Science*, 315:972–976.

Kimura, H., Huo, G., Rasin, A., Madden, S., and Zdonik, S. B. (2009). Correlation maps: A compressed access method for exploiting soft functional dependencies. *PVLDB*, 2(1):1222–1233.

Liu, J., Li, J., Liu, C., and Chen, Y. (2012). Discover dependencies from data - a review. *IEEE Trans. on Knowl. and Data Eng.*, 24(2):251–264.

Mazuran, M., Quintarelli, E., Tanca, L., and Ugolini, S. (2016). Semi-automatic support for evolving functional dependencies. In *EDBT, 2016*.

Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J.-P., Schönberg, M., Zwiener, J., and Naumann, F. (2015). Functional dependency discovery: An experimental evaluation of seven algorithms. *PVLDB*, 8(10):1082–1093.

Papenbrock, T. and Naumann, F. (2016). A hybrid approach to functional dependency discovery. In *SIGMOD*, 2016, pages 821–833.

Szlichta, J., Godfrey, P., Gryz, J., and Zuzarte, C. (2013). Expressiveness and complexity of order dependencies. *PVLDB*, 6:1858–1869.

tutorials

32th Brazilian Symposium on Databases

October 2nd to 5th, 2017
Uberlândia – MG – Brazil

TUTORIALS

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Uberlândia – UFU

Tutorials Chair

Ana Carolina Salgado, UFPE

32nd Brazilian Symposium on Databases

October 2nd to 5th, 2017
Uberlândia – MG – Brazil

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Uberlândia – UFU

SBBD Steering Commitee

Agma Juci Machado Traina, USP
Bernadette Lóscio, UFPE
Caetano Traina Jr., USP
Carmem Hara, UFPR
Javam Machado, UFC
Mirella M. Moro, UFMG
Vanessa Braganholo, UFF

SBBD 2017 Commitee

Steering Committee Chair
Javam Machado, UFC

Local Organization Chairs
Maria Camila N. Barioni, UFU and Humberto L. Razente, UFU

Program Committee Chair
Carmem S. Hara, UFPR

Short papers Chairs
Bernadette Lóscio, UFPE and Damires Souza, IFPB

Demos and Applications Session Chair
Daniel de Oliveira, UFF

Short Courses Chair
Vaninha Vieira, UFBA

Workshop on Thesis and Dissertations in Databases Chair
Carina Dorneles, UFSC

Tutorials Chair
Ana Carolina Salgado, UFPE

Thesis and Dissertation Contest Chair
Vânia Vidal, UFC

Workshops Chair
Fernanda Baião (UNIRIO)

Local Organization Committee

Maria Camila N. Barioni, UFU
Humberto L. Razente, UFU
José Gustavo de Souza Paiva, UFU
Marcelo Zanchetta do Nascimento, UFU
Elaine Ribeiro de Faria Paiva, UFU
João Henrique de Souza Pereira, UFU

Tutorials Program Committee

Agma Traina (ICMC - USP)
Caetano Traina Jr. (ICMC - USP)
Carmem S. Hara (UFPR)
Vanessa Braganholo (UFF)

Editorial

Tutorials at the Brazilian Symposium on Databases (SBBD) have the goal to present introductory and advanced discussions on topics within the area of databases. Introductory tutorials target an audience consisting of advanced undergraduate and graduate students, as well as attendees from industry. Advanced tutorials, on the other hand, cover a state-of-the-art topic, motivating and exposing potential research paths.

The three accepted tutorials this year are related to relevant nowadays topics. The first one is entitled "Mobile Crowdsourcing for Smart City Applications" and will discuss mobile crowdsourcing issues for smart city applications, including topics like data quality, data analytics and trust issues. It will be presented by Sanjay Madria, a full tenured Professor at the Missouri University of Science and Technology.

The second tutorial, "Adaptivity in Database Kernels", will be presented by Javam Machado (Full Professor), Paulo Amora (MSc. student) and Elvis Teixeira (MSc. student) from Federal University of Ceara (UFC). They will talk about adaptivity, i.e., problems related to database physical design optimization for scenarios where the workload is unknown and immediate availability is a requirement. Adaptivity can also be applied on data storage layout in order to optimize relevant data exchange between memory hierarchy layers.

The third tutorial is entitled "Social Professional Networks: Taxonomy, Metrics and Analyses of Relationship Strength". It will present a deeper understanding of the social professional networks types, definitions, features, analysis and applications while providing a useful taxonomy about their use. It also discusses the strength of ties, a central aspect that allows studying the roles of relationships. This tutorial will be presented by Mirella Moro (Associate Professor) and Michele Brandão ("professor substituto") from the Computer Science department at UFMG.

This year we had six excellent submissions and I would like to thank all the authors of submitted proposals. Also, I would like to invite all of you to attend and take advantage of the selected tutorials.

Ana Carolina Salgado, UFPE
SBBD 2017 Tutorials Chair

Table of Contents (Tutorials)

Mobile Crowdsourcing for Smart City Applications 323
Sanjay Kumar Madria

Adaptivity in Database Kernels 326
Javam C. Machado, Elvis Marques Teixeira, Paulo Roberto Pessoa Amora

Social Professional Networks: Taxonomy, Metrics and Analyses of Relationship Strength 331
Michele A. Brandão, Mirella M. Moro

per:tutorial1

Mobile Crowdsourcing for Smart City Applications

Sanjay Kumar Madria
Professor, Department of Computer Science and Director, Web and Wireless Computing Lab, Missouri Institute of Science and Technology, USA
E-mail: madrias@mst.edu

Abstract: This tutorial will discuss mobile crowdsourcing issues for smart city applications. In particular, it will motivate the use of incentives for better crowd participation from people carrying mobile devices, data quality, data analytics and trust issues among others. It will also discuss some open research problems in that domain.

Introduction: Ever-increasing prevalence of social networking using mobile devices has catalyzed the growth of interesting and innovative new-age mobile crowdsourcing applications, which work at the intersection of human-centric computation (e.g., economic incentive management and social computing) and dynamic management of information and content in wireless networks. The prevalence and proliferation of mobile devices coupled with popularity of social media and increasingly technology-savvy users have fuelled the growth of mobile crowdsourcing and participatory sensing. In particular, participatory sensing can occur in various ways by means of devices (e.g., mobile phones, PDAs, laptops and various types of sensors) or by including humans in the loop or both. Notably, participatory sensing can also potentially act as a key enabling technology for various applications involving smarter cities initiatives.

Incidentally, large-scale collection of city-related event data is crucial to effective planning and decision-making for improving city management. Examples of city-related event data include traffic congestion, illegal parking, accidents, dysfunctional streetlights, broken pavements, potholes, planned road construction works, public rallies, waterlogging, uncleared garbage, and the like. Notably, existing sensor-based data collection mechanisms cannot always take human judgment and the context of the event into consideration, and the costs of deploying them across all city locations would be prohibitively expensive. Hence, event data collection by users can be used to complement sensor-based data collection. Observe that a vast majority of the users can be reasonably expected to carry mobile devices. Since mobile devices often come equipped with various kinds of sensors, resident-driven data collection is also well-aligned with current technological trends. However, note that incentives need to be provided to users for encouraging them to contribute event data.

In this environment, research issues include incentives & economic models for crowd participation, large-scale data management, resource discovery, replica allocation and consistency, mobile cloud, analytics on the collected data, resource access, indexing & query processing, mobile computing, and trust. These issues have generated a significant amount of interest in academia as well as in industry. We will discuss some of the open research issues in this area and provide our perspectives on those issues. This tutorial

intends to foster discussions on the key research challenges as well as the design issues of key enabling technologies that need to be addressed to make scalable next-generation mobile crowdsourcing and human computation effective in a real-world application.

1. Mobile Crowd sourcing architectures; Mobile P2P, Mobile Cloud, etc.
2. Incentive Models for Mobile Crowdsourcing
3. Dynamic data management and querying in Crowdsourcing
4. Sensor data management for smart city applications
5. Content management in smart transportation
6. Privacy of Trajectory Data for Smart Transportation

References

Ilarri, Sergio, Ouri Wolfson, and Thierry Delot. "Collaborative sensing for urban transportation." IEEE Data Engineering Bulletin 37 (2014): 3-14.

Nilesh Padhariya, Anirban Mondal, Sanjay Madria, Efficient Processing of Mobile Crowdsourcing Queries with Multiple Sub-tasks for Facilitating Smart Cities, in SmartCities workshop with ACM Middleware, 2016, Italy.

Nilesh Padhariya, Anirban Mondal, Sanjay Kumar Madria: Top-k query processing in mobile-P2P networks using economic incentive schemes. Peer-to-Peer Networking and Applications 9(4): 731-751 (2016)

Nilesh Padhariya, Ouri Wolfson, Anirban Mondal, Varun Gandhi, Sanjay Kumar Madria: E-VeT: Economic Reward/Penalty-Based System for Vehicular Traffic Management. MDM (1) 2014: 99-102

Dejun Yang, Guoliang (Larry) Xue, Xi Fang and Jian Tang, Crowdsourcing to Smartphones: Incentive Mechanism Design for Mobile Phone Sensing, Conference: Proceedings of the 18th annual international conference on Mobile computing and networking, 2012.

Shuo Ma, Yu Zheng, Ouri Wolfson: Real-Time City-Scale Taxi Ridesharing. IEEE Trans. Knowl. Data Eng. 27(7): 1782-1795 (2015)

Y. Wang, X. Jia, Q. Jin, and J. Ma, Mobile crowdsourcing: Architecture, applications, and challenges, Concurrency and Computation: Practice and Experience, 2016.

Kan Yang, Kuan Zhang, Ju Ren, and Xuemin Shen. Security and privacy in mobile crowdsourcing networks: challenges and opportunities. Communications Magazine, IEEE, 53(8):75–81, 2015.

Katrina Ward, Dan Lin and Sanjay Madria, "MELT: Mapreduce-based Efficient Large-scale Trajectory Anonymization", to appear in 29th International Conference on Scientific and Statistical Database Management (SSDBM, 2017), USA.

Biography: Sanjay Kumar Madria received his Ph.D. in Computer Science from Indian Institute of Technology, Delhi, India in 1995. He is a full tenured Professor, Department of Computer Science, at the Missouri University of Science and Technology (formerly, University of Missouri-Rolla), USA. Earlier he was Visiting Assistant Professor in the Department of Computer Science, Purdue University, West Lafayette, USA. He has published more than 230 Journal and conference papers in the areas of mobile computing, sensor networks, security, cloud computing, etc. He won five IEEE best paper awards in conferences including IEEE SRDS 2015, IEEE MDM in 2011 and 2012. He co-authored a book entitled "Web Data Management: A Warehouse Approach" published by Springer-verlag. He guest edited WWW Journal, several Data and Knowledge Engineering Sp. Issues on Web data management and Data warehousing. He was founding Program Chair for EC&WEB conference series. He served as a general co-chair of Mobile Data Management conference in 2010, IEEE Symposium on Reliability in Distributed Systems in 2012 and PC co-chair of MDM 2015. He serves in steering committees of IEEE SRDS and IEEE MDM. He is serving/served as PC member of various conferences such as VLDB, MDM, CIKM, ICDCS, and reviewer for many reputed journals such as IEEE TKDE, IEEE Computer, ACM Internet Computing, IEEE TMC etc. Dr. Madria has given tutorials on mobile data management in many international conferences like Middleware, MDM and SRDS. He is regular invited panelist in NSF, NSERC (Canada), Hong Kong Research Council and Sweden Council of Research. He received UMR faculty excellence award in 2007, 2009, 2011, 2013 and 2015, Japanese Society for Promotion of Science invitational fellowship in 2006, and Air Force Research Lab's visiting faculty fellowship from 2008 to 2016. He was honored with NRC fellowship in 2012-2013 by National Academies. His research is supported by multiple grants from several agencies such as NSF, DOE, NIST, AFRL, ARL, UM research board and from industries such as Boeing. He is IEEE Senior Member, served as IEEE Distinguished speaker and currently he is a speaker under ACM Distinguished Visitor program. He is ACM Distinguished Scientist and IEEE Golden Core Awardee.

per:tutorial2

Adaptivity in Database Kernels

Javam Machado[1], **Elvis Teixeira**[1], **Paulo Amora**[1]

[1]LSBD/DC – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brazil

{javam.machado,elvis.teixeira,paulo.amora}@lsbd.ufc.br

Abstract

Adaptivity addresses a class of problems related to database physical design optimization for scenarios where the workload is unknown and immediate availability is a requirement. The general strategy is to improve physical design by means of incremental changes, each guided by the current workload request. For instance, adaptive indexing builds partial indexes through steps during query processing rather than building full indexes. Adaptivity can also be applied on data storage layout in order to optimize relevant data exchange between memory hierarchy layers. Instead of having a fixed layout, adaptive storage redesigns data organization to answer incoming queries incrementally based on the current requests or recent workload pattern.

Introduction

Analytical processing of large datasets is currently a daily task for many business and for the scientific community. For these tasks it is often the case that users and data scientists do not have a well defined set of queries and data modification procedures, that is, the workload is not known or modeled. This scenario is typical of data exploration environments [Idreos et al. 2015] for data mining or scientific work found in industry or academia.

Another issue on modern data exploration that can be addressed with adaptive approaches is the treatment of hot and cold data. Hot data is characterized as data which is being currently used or accessed by insert and update commands, as well as recent data, which is more suitable to suffer those changes, is usually the data present in an OLTP database system. Cold data is defined as historical, usually accessed by aggregation queries and used for information extraction, like the data accessed by an OLAP database system and stored in data warehouses. So, the more recent the data is, the more valuable it is. This means that the amount of time it takes to load the dataset and optimize its physical layout is a concern, and that traditional database system design is not well suited.

New database setups for such tasks are needed for two main reasons: first, schema, physical design tuning and index selection and creation take time. Second, even if there is time available for database tuning the lack of knowledge of the workload turns hard problems like data model and index selection into even greater challenges since an inappropriate set of indexes or a storage layout unfit for the access patterns provide performance well below optimal, rendering data exploration impractical.

Adaptive systems address both issues by making layout operations less atomic so that they are able to change direction if required by the actual requests, thus optimizing the access to data that is currently being requested. Initially the system is made available

to process user queries as soon as new data arrives [Idreos et al. 2007] with a physical design composed of simple and plain data structures and without any optimization. Again, this decision is made based on the premises that immediate availability is desired and no knowledge is available to lead to good decisions on how to tune the physical design.

This tutorial touches the most recent literature on adaptivity in various aspects of database architecture discussing and comparing their strengths and weaknesses, and presents the architecture of some well established adaptive indexing and storage techniques. Focus will be directed towards the lower levels of a DBMS, that is, the storage and data access layers. Additionally, storage and data access is where most of the important recent developments occurred and pioneering works on adaptivity in higher level database components, such as transaction management [Graefe et al. 2014] will be presented as guidelines for valuable future work.

Adaptive Storage

The way data is stored is usually defined by the system administrator in a manual process, with the objective of optimizing query response times. This data organization takes into account the application domain, how that data is accessed, and other factors. These constraints have one common factor: they all necessitate previous knowledge to be fulfilled, be it about query workload or semantic relationships between attributes of a schema. The DBA is assumed to have this knowledge, but most database systems rely on a static data layout. Therefore, adapting these layouts as queries are executed to allow optimal data organization without previous knowledge is a desirable feature, enabling the database to answer queries more quickly, without external interference and without requiring any knowledge about the data or how it is accessed.

In this tutorial we focus on three adaptive storage approaches: PelotonDB [Arulraj et al. 2016], H_2O [Alagiannis et al. 2014] and HyPer [Lang et al. 2016].

H_2O is a OLAP storage system which leverages queries access patterns to reorganize data layouts, incurring in several layouts to the same data, each optimized to a different access pattern. This self-designing approach requires different physical operators to access the different data organizations, and this is done by creating these operators on-the-fly, through code templates.

PelotonDB is a HTAP (Hybrid Transactional Analytic Processing) oriented database that fragments data into horizontal and vertical partitions called physical tiles as data becomes colder, for example, when it ceases to be accessed in a transactional way. A table can keep several data layouts, called tile groups. Data access is done by a single execution engine, through the abstraction of logical tiles, which contain pointers to the data in physical tiles. It also employs special techniques for recovery and logging aimed at novel Non-Volatile Memory hardwares.

HyPer is another HTAP oriented database that provides hardware manipulation techniques. The main process handles the OLTP workload and when a OLAP workload shows up, the main process forks itself. The forked process then have access to the data through a virtual memory snapshot. At first, data was categorized within four categories, Hot, Cooling, Cold and Frozen. Hot, Cooling and Cold items had only conceptual differences among them, all of them were stored in small memory pages and were uncompressed. The Frozen items, however, were compressed and stored in huge pages. To track

the temperature of items, special in-memory flags, such as *dirty* and *young* are manipulated by the database, overriding the OS, through a special memory allocation. Nowadays, HyPer proposes a structure named Data Blocks, which is based on the same principle of splitting data in hot, uncompressed parts and cold, immutable, compressed parts. It makes use of Positioned Small Materialized Aggregates (PSMA) as an index to speed up scan operations. By having a maximum and a minimum for each block, it can save *scan* operations on blocks where the value searched isn't present for sure.

Adaptive Indexing

Adaptive indexing came up as the first promise that a workload-driven, self-tuning database system architecture is possible. The fundamental idea is tuning the access structures to provide faster response times for key ranges which have already been queried, and its vicinity, by indexing them at query processing time. The underlying assumption is that some key ranges will be of more interest than others so indexing them only when queried would be better since it is faster than building full indexes and the focus of indexing can change as soon as the query focus changes.

The first technique to be extensively tested was Database Cracking [Idreos et al. 2007], an approach that applies an incremental quicksort-like partitioning scheme to optimize the access for key ranges which are being focus of current query attention. This technique was conceived and implemented in the context of Column Store database systems. The process on indexing consists of choosing the boundaries of range query predicates as pivots for the next partition, so the expected behavior of the response time is to decrease from that of a linear scan towards that of a full index scan.

Adaptive Merging [Graefe and Kuno 2010b] is other distinctive technique focused on traditional tuple-based storage layouts, and it harnesses the well known behavior of B+-trees or, more specifically, partitioned B+-trees for index data structure. The first query instructs the system about index selection, the heuristic used is creating the index for the attribute that mostly benefits this first query. The process of index building starts in this first query by performing a full linear scan in the table and sorting it piece by piece in main memory. After the first query is done, the data will be again in disk but in the form of sorted runs ready to be turned into the leaves of the tree by an external merge sort process. Subsequent queries are processed by searching within each run and merging their key range into the final structure.

The main advantage of adaptive merging is the fact that its underlying data structure is cache friendly, that is, units of data storage and transfer are naturally paged, as in any B-tree related structure. This is ideal for databases stored in block devices like hard disks or flash drives, or even for the exchange of data between higher cache hierarchies like main memory and processor caches. Further developments that point out ways to go beyond adaptivity [Halim et al. 2012] [Petraki et al. 2015] Will be briefly touched in the tutorial.

About the authors:

Javam C. Machado is a full professor at the Federal University of Ceara (UFC), Brazil. He obtained a MsC in Computer Science from the Federal University of Rio Grande do Sul, Brazil and a PhD degree in Computer Science from the University of Grenoble,

France. For 8 years, he was the manager of the UFC's IT infrastructure and, in 2011, he became the vice-director of the College of Science at the same University. Since 1995, he has coordinated research projects on the area of Database and Distributed Systems and he has advised MsC and PhD candidates. Javam is the coordinator of the Database Special Committee of the Brazilian Computer Society (2017) and he was the chair of the 2016 edition of the Brazilian Databases Systems Symposium. He has served as a member of the program committee of different conferences on database and on cloud computing. In 2010 Javam founded the Databases and Systems Laboratory (LSBD) of the Computer Science Department which he coordinates since than. Lately he is interested more particularly in data management and data privacy but also in adaptive database systems and cloud computing.

Paulo Roberto Pessoa Amora is a Computer Science MSc. student at UFC - Federal University of Ceará under Prof. Javam Machado. He is a member of the Databases and Systems Laboratory (LSBD) currently working on adaptive storages in main-memory databases. Paulo earned a BSc. in Computer Engineering at IFCE - Instituto Federal do Ceará with a sandwich year at University of Pittsburgh. He has interest in the following research topics: Adaptive Databases, Main-Memory Databases, Storage management.

Elvis Marques Teixeira is a Computer Science MSc. student at UFC - Federal University of Ceará under Prof. Javam Machado. He is also a member of the Databases and Systems Laboratory (LSBD) currently working on adaptive indexing techniques. Elvis earned a BSc. in Physics at UFC - Federal University of Ceará, meanwhile conducted research on mineralogy data processing. He has interest in the following research topics: Adaptive Indexes, data analysis and applications of machine learning on the design and adaptation of data access methods.

Referências

Alagiannis, I., Idreos, S., and Ailamaki, A. (2014). H2O: a hands-free adaptive store - read. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, 2014*, pages 1103–1114.

Arulraj, J., Pavlo, A., and Menon, P. (2016). Bridging the archipelago between row-stores and column-stores for hybrid workloads. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, 2016*, pages 583–598.

Graefe, G., Halim, F., Idreos, S., Kuno, H. A., Manegold, S., and Seeger, B. (2014). Transactional support for adaptive indexing. *VLDB J.*, 23(2):303–328.

Graefe, G. and Kuno, H. A. (2010a). Adaptive indexing for relational keys. In *Workshops Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, Long Beach, California, USA*, pages 69–74.

Graefe, G. and Kuno, H. A. (2010b). Self-selecting, self-tuning, incrementally optimized indexes. In *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, 2010, Proceedings*, pages 371–381.

Halim, F., Idreos, S., Karras, P., and Yap, R. H. C. (2012). Stochastic database cracking: Towards robust adaptive indexing in main-memory column-stores. *PVLDB*, 5(6):502–513.

Idreos, S., Kersten, M. L., and Manegold, S. (2007). Database cracking. In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, 2007, Online Proceedings*, pages 68–78.

Idreos, S., Papaemmanouil, O., and Chaudhuri, S. (2015). Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, 2015*, pages 277–281.

Lang, H., Mühlbauer, T., Funke, F., Boncz, P. A., Neumann, T., and Kemper, A. (2016). Data blocks: Hybrid OLTP and OLAP on compressed storage using both vectorization and compilation. In *SIGMOD Conference*, pages 311–326. ACM.

Petraki, E., Idreos, S., and Manegold, S. (2015). Holistic indexing in main-memory column-stores. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, 2015*, pages 1153–1166.

per:tutorial3

Social Professional Networks:

Taxonomy, Metrics and Analyses of Relationship Strength

Michele A. Brandão, UFMG
micheleabrandao@dcc.ufmg.br
www.dcc.ufmg.br/~micheleabrandao

Mirella M. Moro, UFMG
mirella@dcc.ufmg.br
www.dcc.ufmg.br/~mirella

Abstract

Social professional networks provide features not available in other networks. For example, LinkedIn facilitate professional networking, and GitHub enables committing and sharing code. Such networks also provide data on users, behaviors and interactions. Here, we foster a deeper understanding of the social professional networks types, definitions, features, analyses and applications while providing a useful taxonomy about their use. We also study the strength of ties, a central aspect that allows studying the roles of relationships. Therefore, besides analyzing the strength of co-authorship ties, we also present a set of metrics and algorithms to measure such strength in different contexts.

Outline

1. Introduction

Social Networks Analysis has evolved from a Social Sciences area to a Computer Science-based Multidisciplinary research area. Despite the many analyses possible, there are two main aspects to any research at both perspectives (Social and Computer Science): (*i*) how to collect and manage social data, and (*ii*) how to build and analyze the social networks derived from such data.

Furthermore, extracting and analyzing relevant information from social networks provide many challenges for developers, user, and technology. For developers, after collecting data from collaborators, it is necessary to model, store and manage them within databases with proper interface to whatever application uses them. For users, when they need to obtain relevant knowledge from these networks. For technology, which should provide the necessary support for implementation of methodologies. For instance, exploring collaborative relations can improve the accuracy and quality of existing methods that combine bibliometry, altmetrics and academic social analysis.

A specific perspective of evaluation is given by *academic social networks*, in which nodes represent researchers and edges their co-authorships and academic relations. Building the structure of such networks is relatively simple, as the nodes are given by any set of researchers who are connected through their common published work, for example. However, one central aspect of more complex analysis is the *strength of the ties* among researchers, as pairs of researchers have stronger or weaker connections depending on the degree of academic relationship. Such degree of relationship (or *tie strength*) may be defined according to Granovetter's theory: the ties are *weak* when they serve as

bridges in the network by connecting users from different groups, and *strong* when they link individuals in the same group (community) [1].

Formally, tie strength may be measured by a combination of the amount of time, the cooperation intensity and the reciprocal services that characterize the tie [1]. Such strength may also be measured by using the neighborhood overlap metric (also known as topological overlap), a numerical quantity that captures the total number of collaborations between the two ends of each edge. Note that neighborhood overlap has been used for uncovering the community structure and analyzing structural properties of a large network of mobile phone users, besides measuring tie strength [2].

As for practical applications, the study of social ties has lead to building rigorous models that reveal the evolution of social networks and their dynamism. Indeed, the strength of ties has been largely explored in different contexts, such as information diffusion, analyses of patterns in communication logs and evaluation of scientific researchers productivity [1-6]. Moreover, analyzing tie strength allows investigating how distinct relationships play different roles and identifying impact at micro-macro levels in a network.

Finally, properly measuring the strength of co-authorship ties may help to identify which collaborations are more influent to each researcher. For example, if a researcher *A* collaborates with other researchers *B* and *C*, the strength of ties reveals which one is more important to *A*, then allowing different studies, such as team formation analyses. Also, researchers that form mostly weak (or strong) ties in the social network may indicate different collaboration patterns. Among others, a researcher who has many collaborators through single papers, i.e., that person has collaborated only once with many people.

2. Background and Main Definitions

This part of the tutorial goes over the background necessary to understand the tutorial. Specifically: an overview of general concepts for social networks (building process, modeling, metrics and properties) and social professional networks examples and features.

3. General Taxonomy for Professional Networks

We present a taxonomy based on the tasks and issues of social networks. By analyzing the publications on the area, we have identified (*i*) two main tasks: analysis and applications; and (*ii*) two main issues: data acquisition and preparation, and data storage. After covering such a taxonomy, we further describe a specific analysis (clustering) and two important applications (recommendation and ranking). This part of the tutorial is mostly based on the paper [3].

4. Tie Strength: Concepts and Existing Metrics

Tie strength in social networks has been addressed with diverse goals such as measuring the strength of weak ties [1], co-authorship ties [2], contact ties [4], friendship ties [5] and work ties [6]. Such studies contextualize the importance of measuring tie strength in an appropriate manner: relationships play different roles and should be distinctly qualified as well through (for example) their strength. Indeed, studies show that a relationship has large impact at micro-macro levels in the network, depending on its strength, and influences the patterns of communications [1, 2, 5]. Tie strength can be calculated by considering topological and/or semantic properties. This part of the

tutorial goes over such properties and presents an analysis over co-authorships social networks of three different research areas (Medicine, Computer Science and Sociology), mostly based on the paper [2].

5. New Metrics for Tie Strength and their Applications

This part goes over new metrics for calculating tie strength over static networks as well as temporal networks, and is divided in (mostly based on the papers [7, 8, 9, 10]):

a. An analysis of problems of measuring tie strength using solely neighborhood overlap or co-authorship frequency.

b. A new tie strength metric to non-temporal social networks.

c. An analysis of tie persistence and transformation in temporal social networks by using an existing algorithm and a new one.

6. Open Problems and Possible Future Work

Finally, besides open problems and possible future work, we go over our current research over social coding networks. This part includes the paper *Collaboration Strength Metrics and Analyses on GitHub*, recently accepted for publication at IEEE/WIC/ACM Web Intelligence.

References

[1] M. Granovetter. The strength of weak ties. american joumal af sociology, 78:1360-1380. Dietz. Pugh. and Wiley, 91(2004):423--433. 1973.

[2] Michele A. Brandão and Mirella M. Moro. Analyzing the strength of co-authorship ties with neighborhood overlap. In *Proceedings of the International Conference on Database and Expert Systems Applications* (DEXA), pages 527-542. 2015.

[3] Michele A. Brandão and Mirella M. Moro. Social professional networks: A survey and taxonomy. *Computer Communications* 100: 20–31. 2017.

[4] J. Wiese et al., You never call, you never write: Call and sms logs do not always indicate tie strength. In *Procs. of CSCW*, pages 765--774. 2015.

[5] M. Zignani, S. Gaito, and G. P. Rossi. Predicting the link strength of newborn links. In *Proceedings of International Conference on World Wide Web - Companion Volume*, pages 147--148. 2016.

[6] D. Castilho, P. O. Vaz de Melo, and F. Benevenuto. The strength of the work ties. *Information Sciences*, 375:155--170. 2017.

[7] Michele A. Brandão and Mirella M. Moro. The strength of co-authorship ties through different topological properties. *Journal of the Brazilian Computer Society* 23:5. 2017.

[8] Michele A. Brandão, Pedro O. S. Vaz de Melo and Mirella M. Moro. Tie Strength Persistence and Transformation. In *Proceedings of the Alberto Mendelzon Workshop* (AMW), 2017.

[9] Michele A. Brandão and Mirella M. Moro. Strength of Co-authorship Ties in Clusters: a Comparative Analysis. In *Proceedings of the Alberto Mendelzon Workshop* (AMW), 2017.

[10] Michele A. Brandão, Pedro O. S. Vaz de Melo, Mirella M. Moro. Tie Strength Dynamics over Temporal Co-authorship Social Networks. In *Proceedings of IEEE/WIC/ACM Web Intelligence* (accepted for publication), 2017.

Authors' Bio

Michele A. Brandão has recently finished her PhD in Computer Science at UFMG (Belo Horizonte, Brazil), with title *Tie strength in co-authorship social networks: analyses, metrics and a new computational model*. She holds a Masters degree from UFMG, and a Bachelor from Universidade Estadual de Santa Cruz, both in Computer Science. She is currently at UFMG as staff (*"professor substituto"*). Her research interests are mostly in data science, social networks, recommender systems and link prediction. Recent publications include papers: at journals Computer Communications, Journal of the Brazilian Computer Society, and JIDM; as well as at conferences IEEE/WIC/ACM Web Intelligence, BRASNAM, AMW, DEXA, and SBBD.

Mirella M. Moro is associate professor at the Computer Science department at UFMG (Belo Horizonte, Brazil). She holds a Ph.D. in Computer Science (University of California Riverside - UCR, 2007), and MSc and BSc in Computer Science as well (UFRGS, Brazil). She is a member of the ACM Education Council, ACM SIGMOD, ACM SIGCSE, ACM-W, IEEE, IEEE WIE, SBC, and MentorNet. She was the Education Director of SBC (Brazilian Computer Society, 2009-2015), the editor-in-chief of the electrocnic magazine SBC Horizontes (2008-2012), and associated editor of JIDM (2010-2012). Mirella has been working with research in Computer Science in the area of Databases since 1997. Her research interests include social networks analysis, query optimization, and hybrid SQL/NoSQL modeling. Her recent publications include papers on prestigious venues such as Scientometrics, Data & Knowledge Engineering, ACM Hypertext, IEEE/WIC/ACM Web Intelligence, TPDL, JCDL as well as JIDM and SBBD.

talks

32th Brazilian Symposium on Databases

October 2nd to 5th, 2017
Uberlândia – MG – Brazil

INVITED TALKS

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Uberlândia – UFU

Table of Contents (Invited Talks)

A Holistic View of Human Factors in Crowdsourcing 337
Sihem Amer-Yahia

What Non-Volatile Memory Means for the Future of Database Management Systems 338
Andy Pavlo

Pesquisador Homenageado do SBBD 2017 339
José Palazzo Moreira de Oliveira

paper:talk1

A Holistic View of Human Factors in Crowdsourcing

Sihem Amer-Yahia

For over 40 years, organization studies have examined human factors in physical workplaces and their influence on the ability of an individual to perform a task, or a set of tasks, alone or in collaboration with others. In a virtual marketplace, the crowd is typically volatile, its arrival and departure asynchronous, and its levels of attention and accuracy diverse. This has generated a wealth of new research ranging from studying workers' fatigue in task completion to examining the role of motivation in task assignment. I will review such work and argue that we need a holistic view to take full advantage of human factors such as skills, expected wage and motivation, in improving the performance of a crowdsourcing platform.

Sihem Amer-Yahia is a CNRS Research Director in Grenoble where she leads the SLIDE team. Her interests are at the intersection of large-scale data management and data analytics. Before joining CNRS, she was Principal Scientist at QCRI, Senior Scientist at Yahoo! Research and Member of Technical Staff at at&t Labs. Sihem served on the SIGMOD Executive Board, the VLDB Endowment, and the EDBT Board. She is Editor-in-Chief of the VLDB Journal. Sihem is PC co-chair for VLDB 2018. Sihem received her Ph.D. in CS from Paris-Orsay and INRIA in 1999, and her Diplôme d'Ingénieur from INI, Algeria.

paper:talk2

What Non-Volatile Memory Means for the Future of Database Management Systems

Andy Pavlo

The advent of non-volatile memory (NVM) will fundamentally change the dichotomy between memory and durable storage in database management systems (DBMSs). These new NVM devices are almost as fast as DRAM, but all writes to it are potentially persistent even after power loss. Existing DBMSs are unable to take full advantage of this technology because their internal architectures are predicated on the assumption that memory is volatile. That means when NVM finally arrives, just like when you finally passed that kidney stone after three weeks, everyone will be relieved but the transition will be painful. Many of the components of legacy DBMSs will become unnecessary and will degrade the performance of data intensive applications.

In this talk, I discuss the key aspects of DBMS architectures that are affected by emerging NVM technologies. I then describe how to adapt in-memory DBMS architectures for NVM. I will conclude with a discussion of a new DBMS that we have been developing at Carnegie Mellon that specifically designed to leverage the persistence properties of NVM in its architecture, such as its recovery and concurrency control mechanisms. Our system is able to achieve higher throughput than existing approaches while reducing the amount of wear due to write operations on the device.

Andy Pavlo is an Assistant Professor of Databaseology in the Computer Science Department at Carnegie Mellon University. At CMU, he is a member of the Database Group and the Parallel Data Laboratory. His work is also in collaboration with the Intel Science and Technology Center for Big Data and Visual Computing Systems. He was a recipient of the 2014 ACM SIGMOD Jim Gray Dissertation Award.

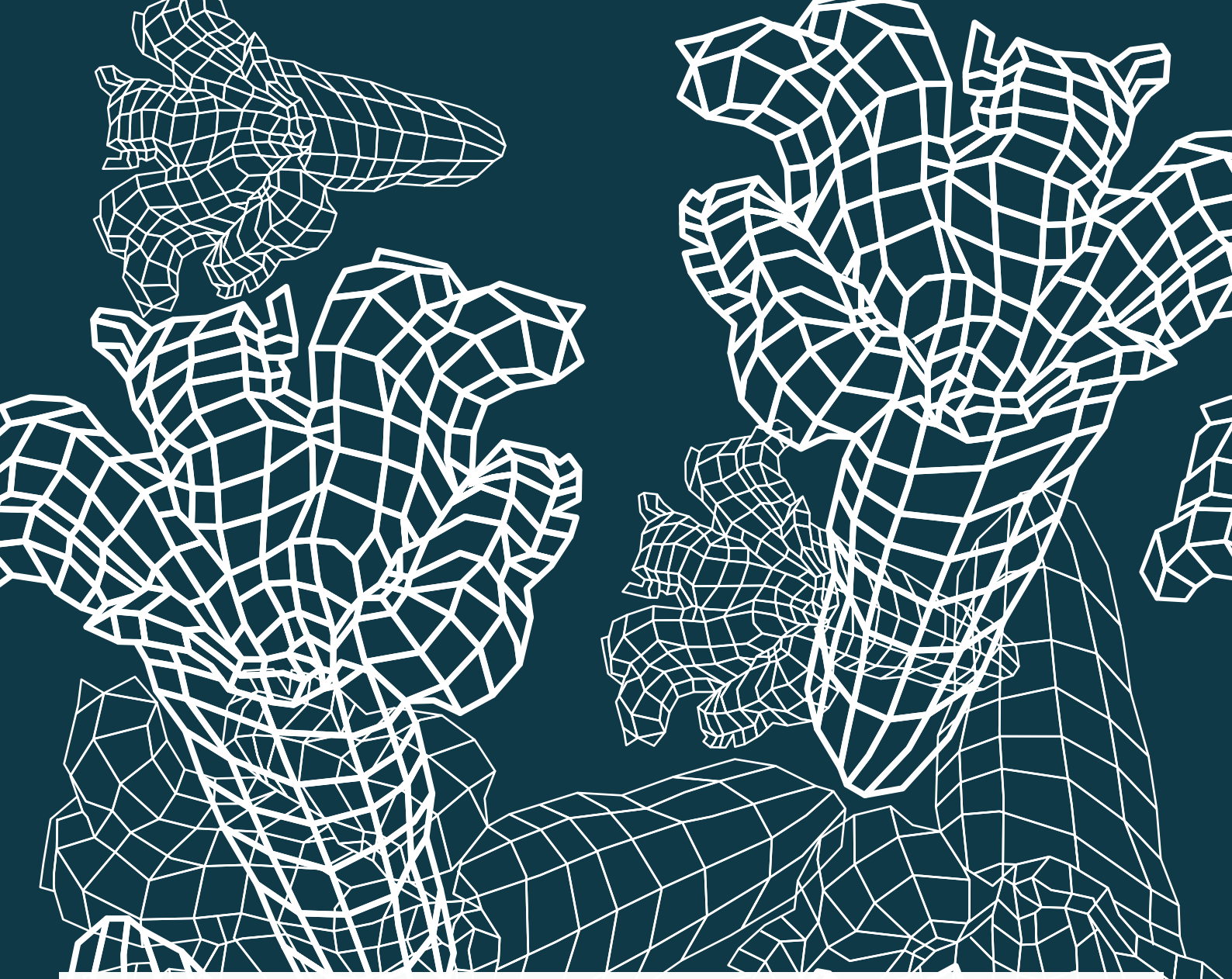
paper:talk3

Pesquisador Homenageado do SBBD 2017

José Palazzo Moreira de Oliveira

Como professor e pesquisador, desde minha formatura na Escola de Engenharia da UFRGS, tive a rara sorte de acompanhar o desenvolvimento da Computação e do ensino de Banco de Dados nas universidades brasileiras. Minha ontogênese acadêmica acompanhou o percurso da história do SBBD. Esta distinção foi uma grande alegria e surpresa, em uma época em que a avaliação de um pesquisador é constituída quase exclusivamente por índices bibliométricos, em receber um reconhecimento pelo *conjunto da obra*. Algo muito relevante para mim foi que os jovens colegas se lembraram de uma carreira de 48 anos com forte dedicação à área de Sistemas de Informação e Banco de Dados. Ao longo da carreira desenvolvi atividades em múltiplas dimensões, 81 alunos de pós-graduação já orientados, muitas disciplinas ministradas, forte interação internacional e um consistente número de boas publicações. Tinha que decidir o formato desta apresentação, uma alternativa seria descrever tecnicamente minhas pesquisas, representadas pelas publicações, isto seria enfadonho e traria pouca contribuição para os jovens membros a comunidade. Pensei melhor e então resolvi apresentar as áreas de pesquisa em que tenho trabalhado e sua evolução ao longo destes anos, sem entrar em profundos detalhes técnicos. Este andamento seguiu muito de perto a evolução do SBBD. Após apresento uma perspectiva do futuro dos Bancos de Dados e os perigos que corremos. Uma das atividades realizadas na Comissão Especial de BD e que considero importante foi a implementação do 1° Concurso de Teses e Dissertações em Banco de Dados. Desejo que a apresentação seja útil para os jovens pesquisadores conhecerem melhor o caminho percorrido até aqui pela nossa comunidade e para que entrevejam o possível futuro e seus desafios. A vida acadêmica não pode ser uma *Torre de Marfim*, a preocupação e engajamento com a comunidade é essencial. Nesta apresentação vocês terão a oportunidade de conhecer, de forma agradável, o desenvolvimento de nossa área no Brasil em paralelo com uma análise do que considero essencial para uma carreira equilibrada no ensino e na pesquisa. A história dos Bancos de Dados inicia com a estruturação de arquivos tradicionais e chega aos complexos sistemas atuais. As noções de transação, recuperação e outras são essenciais para a maioria das aplicações transacionais. Hoje há uma revolta contra tudo isto propondo alternativas como o NoSQL, mas diferentes aplicações exigem diversos modelos de SGBDs. Talvez estejamos exagerando nas *customizações*. O que nos reserva o futuro? Como vamos estruturar nossas carreiras em um período turbulento?

José Palazzo Moreira de Oliveira é Professor Titular por concurso público do Instituto de Informática da UFRGS, atualmente atuando como docente convidado. Possui graduação em Engenharia Elétrica pela Universidade Federal do Rio Grande do Sul (1968), mestrado em Ciências da Computação pela Universidade Federal do Rio Grande do Sul (1976) e doutorado em Informática pelo Instituto Nacional Politécnico de Grenoble (1984). Tem experiência na área de Ciência da Computação, com ênfase em Sistemas de Informação, atuando principalmente nos seguintes temas: ontologia, modelagem conceitual, ensino a distância, banco de dados, sistemas de informação e sistemas na Web. É conselheiro da Sociedade Brasileira de Computação. Foi Coordenador do PPGC/UFRGS, participou da criação dos programas de doutorado em Computação e Administração da UFRGS, foi vice-presidente e membro da Câmara de PG da UFRGS, membro do Comitê Assessor em Ciência da Computação do CNPq – CA-CC, coordenador do Comitê de Matemática, Estatística e Computação – MEC – da Fundação de Amparo à Pesquisa do RS – FAPERGS, implantou o Curso de Informática Instrumental para professores do Ensino Médio oferecido pela UFRGS para a UAB. Tendo sido coordenador e membro da Comissão Especial de Banco de dados da SBC e membro da Comissão de Educação da SBC.



REALIZAÇÃO

SBC
Sociedade Brasileira de Computação

ORGANIZAÇÃO

UFU

Faculdade de Computação

APOIO

ProPP

UBERLÂNDIA
CONVENTION & VISITORS BUREAU

FOMENTO

CAPES

FAPEMIG

CNPq

PATROCÍNIO DIAMANTE

Itaú

NVIDIA

PATROCÍNIO OURO

Algar Telecom

PATROCÍNIO PRATA

Google

IBM Research | Brasil

PATROCÍNIO BRONZE

SEBRAE

neppo

SANKHYA
GESTÃO DE NEGÓCIOS

ClickPerformance®