October 7-10 • Ceará • Brazil

34th Brazilian Symposium on DATABASES

SBBD|2019

PROCEEDINGS

October 7-10 • Ceará • Brazil

34th Brazilian Symposium on DATABASES

PROCEEDINGS

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

Organization
Departamento de Computação, Universidade Federal de Ceará– UFC

Steering Committee Chair
Bernadete Farias Lóscio (UFPE, Brazil)

Local Chair
José Maria da Silva Monteiro Filho (UFC, Brazil)

Program Committee Chairs
Full Paper: Carina F. Dorneles (UFSC, Brazil)
Short Paper: Fábio Porto (LNCC, Brazil)
Demos and Applications Chair: Robson L. F. Cordeiro (ICMC-USP, Brazil)
Thesis and Dissertation Workshop Chair: Jonice Oliveira (UFRJ, Brazil)
Tutorials Chair: Altigran Soares da Silva (UFAM, Brazil)
Short course Chair: Maria Cláudia Cavalcanti (IME, Brazil)
Workshop Chair: José Antônio Macedo (UFC, Brazil)
Thesis and Dissertation Contest Chair: Caetano Traina Jr. (USP, Brazil)
Graduation Student Workshop Chair: Ticiana Linhares (UFC, Brazil)

B839
Brazilian Symposium on Databases (SBBD 2019) (25.: 2019 october 07-10, 2019 – Fortaleza, CE)
Proceedings of 34nd Brazilan Symposium on Databases - SBBD 2019 [recurso eletrônico] / Organização: Carina Friedrich Dorneles e Fabio Andre Machado Porto – Fortaleza: SBC, 2019.

355p. v.1

Access Mode: http://sbbd.org.br/2019/

ISSN: 2016-5170

1. Computação - Congressos. 2. Base de Dados – Congressos. I. Dorneles, Carina Fiedrich. II. Sociedade Brasileira de Computação. III. Título.

CDD: 005

Elaborada pela Biblioteca da Universidade Federal do Ceará Campus Crateús (BCCR).

Message from the Local Organization Committee Chairsf

Welcome to the 34th Brazilian Symposium on Databases and to Fortaleza, Ceará! 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 34th edition of the symposium (SBBD 2019) was held in Fortaleza, in the state of Ceará, from October 7th to 10th, 2019. The local organization was performed by the Federal University of Ceará (UFC) through the Computer Science Department (DC). This year, for the first time, SBBD had the Symposium on Knowledge Discovery, Mining and Learning (KDMiLe); the Brazilian Symposium on Bioinformatics (BSB) and the ACM Latin American School on Recommender Systems (LARS) as co-located events providing a rich environment for the discussion of researches of their interrelated areas.

The SBBD 2019 program offers a wide 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 2 tutorials, presented by distinguished speakers from Brazil, Chile and Germany; 9 technical sessions; 4 short courses about hot topics in the area, presented by specialists in their research fields; demos and applications session; posters sessions; industrial session, thesis and dissertations workshop; the biannual thesis and dissertations contest; 2 co-located workshops; the 3rd KDDBR (Brazilian Knowledge Discovery in Databases) competition; and a panel.

The excellence of SBBD 2019 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 2019 edition. We thank the symposium chairs and our colleagues of the local organization committee who donated their precious time to made SBBD 2019 a reality. We also thank the Computer Science Department (DC) of the Federal University of Ceará (UFC) and its Post-graduation Program (MDCC), which allowed their staff and students to help on the many tasks of the event preparation. 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 2019. 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 2019 in Fortaleza, Ceará!

José Maria da Silva Monteiro Filho, UFC
SBBD 2019 Local Organization Committee Chair

Table of Contents

Editorial

It is a great pleasure to introduce the Proceedings of the Brazilian Symposium on Databases (SBBD) with the full, short, vision and industry papers accepted for presentation at the 34rd edition of the symposium. SBBD 2019 was held in Fortaleza, in the state of Ceará, Brazil, from October 7th to 10th, 2019. This edition was organized by Universidade Federal do Ceará (UFC, Brazil) and Centro Universitário 7 de Setembro (UNI7, Brazil), both located in the state of Ceará. SBBD is the official database event of the Brazilian Computer Society (SBC) and the largest venue in Latin America forum for presenting and discussing research results and applications on data management. The main areas of interest include Core Database Foundations and Technology, Knowledge Modeling and Management, Information and Data Management, Knowledge Discovery in Databases and Machine Learning and Data Mining. Along with technical sessions, SBBD includes invited talks and tutorials given by distinguished speakers from the national and international research communities. SBBD regularly promotes a demos and applications session, a thesis and dissertations workshop, as well as a thesis and dissertation contest as co-located events. 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 2019 edition, SBBD accepted six categories of submissions: JIDM articles, full papers, short papers, industrial papers, vision papers, and distinguished published papers. Submissions to the JIDM category could be made throughout the year.

The full papers track had one cycle of submissions, and the review process was performed in one 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 47 submitted papers, 14 were accepted as full papers (acceptance rate of 29,7%). The best full papers received award certificates during the event, and their authors will be invited to submit extended versions of their papers to JIDM.

Distinguished Published Papers is a category of submission introduced in SBBD 2017, aiming at attracting the best papers of the Brazilian researchers, published or accepted for publication by a first class database conference, and give the authors the opportunity to

present their work during the event. In this year, five submissions were accepted for presentation by the SBBD Steering Committee.

The topics with more full papers submissions (according to the author's indication from the Topics of Interest) were: Algorithms and Techniques for Data Mining (21 submissions), Data Analytics and Data Visualization (21, submissions), Data Cleaning, Information Filtering and Dissemination (16 submissions), Performance Evaluation and Benchmarking (12 submissions), Data Management for Machine Learning (11 submissions), Database Design and Data Semantics (11 submissions), Information Retrieval Models and Techniques (11 submissions), Information Integration and Interoperability (10 submissions), Knowledge Bases and Modeling (10 submissions).

The short, vision and industry tracks of SBBD 2019 received 38 submissions. All papers were reviewed by at least 3 reviewers and, after a discussion period, 23 were accepted. The set of accepted papers include one from each of vision and industry tracks. The papers were organized into three technical sessions for oral presentation and a poster session. The technical session gathered papers in three themes: DBMS and Big Data; Data Science and Data Model, Integration and Data Structure.

This year we organized an invited industry session counting with three keynotes and a panel from experiences in data management and analysis in large companies, such as Facebook, Petrobras and LeanXcale.

The Proceedings of SBBD are the result of the collective effort of a large community, which we gratefully acknowledge. We thank the SBBD 2019 local organization committee and its symposium chairs, who worked hard to guarantee an outstanding event. We do not have enough words to thank all committee members and external reviewers for their commitment and high-quality reviews. We are also grateful to the steering committee members for their help, advice and support. Finally, we are grateful to the authors who submitted their work to SBBD 2019.

Carina F. Dorneles (UFSC)
Program Chair – SBBD 2019 – Full Papers

Fábio Porto (LNCC)
Program Chair – SBBD 2019 – Short Papers

34th Brazilian Symposium on Databases

October 7-10, 2019
Fortaleza - CE - Brazil

SBBD PROCEEDINGS
FULL PAPERS

Promotion

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

Organization

Departamento de Computação, Universidade Federal de Ceará– UFC

Program Chair

Carina F. Dorneles, UFSC

34th Brazilian Symposium on Databases

October 7-10, 2019
Fortaleza - CE - Brazil

Promotion

Brazilian Computer Society – SBC
SBD Database Steering Committee

Organization

Departamento de Computação, Universidade Federal de Ceará– UFC

SBBD Steering Commitee

Ângelo Brayner (UFC)
Bernadette Lóscio (UFPE) Steering Committee Chair
Carina Dorneles (UFSC)
Sérgio Lifschitz (PUC-Rio)
Fábio Porto (LNCC)
Carmem Hara (UFPR)

SBBD 2019 Commitee

Steering Committee Chair
Bernadette Lóscio (UFPE)

Local Chair
José Maria da Silva Monteiro Filho (UFC)

Full Paper Chair
Carina F. Dorneles (UFSC, Brazil)

Short Paper Chair
Fábio Porto (LNCC, Brazil)

Demos and Applications Chair
Robson L. F. Cordeiro (ICMC-USP, Brazil)

Thesis and Dissertation Workshop Chair
Jonice Oliveira (UFRJ, Brazil)

Tutorials Chair

Altigran Soares da Silva (UFAM, Brazil)

Short course Chair
Maria Cláudia Cavalcanti (IME, Brazil)

Workshop Chair
José Antônio Macedo (UFC, Brazil)

Thesis and Dissertation Contest Chair
Caetano Traina Jr. (USP, Brazil)

Graduation Student Workshop Chair
Ticiana Linhares (UFC, Brazil)

Local Organization Committee
SBBD Local Chair: José Maria da Silva Monteiro Filho (DC/UFC)
Leonardo Oliveira Moreira (Instituto UFC Virtual/UFC)
Marum Simão Filho (UNI7)
Angelo Roncalli de Alencar Brayner (DC/UFC)
Javam de Castro Machado (DC/UFC)

Full Papers Program Committee
Agma Traina (ICMC-USP, Brazil)
Alberto Laender (UFMG, Brazil)
Alexandre Plastino (UFF, Brazil)
Altigran Soares da Silva (UFAM, Brazil)
Ana Carolina Salgado (UFPE, Brazil)
André Santanchè (UNICAMP, Brazil)
Angelo Brayner (UFC, Brazil)
Bernadette Loscio (UFPE, Brazil)
Caetano Traina Júnior (ICMC – USP, Brazil)
Carina F. Dorneles (UFSC, Brazil)
Carmem Hara (UFPR, Brazil)
Celso Hirata (ITA, Brazil)
Clodoveu Davis (UFMG, Brazil)
Cristina Ciferri (USP, Brazil)
Damires Souza (IFPB, Brazil)
Daniel de Oliveira (UFF, Brazil)
Daniel Kaster (UEL, Brazil)
Denio Duarte (UFFS, Brazil)
Dimas C. Nascimento (UFRPE, Brazil)
Duncan Ruiz (PUCRS, Brazil)
Edleno Moura (UFAM, Brazil)
Eduardo de Almeida (UFPR, Brazil)
Eduardo Ogasawara (CEFET/RJ, Brazil)
Elaine Sousa (USP, Brazil)
Fabio Porto (LNCC, Brazil)
Fernanda Baião (PUC-Rio, Brazil)
Flavio Horita (UFABC, Brazil)

Genoveva Vargas-Solar – French Council on Scientific Research
Humberto Razente (UFU, Brazil)
Javam Machado (UFC, Brazil)
João Eduardo Ferreira (IME/USP, Brazil)
José Palazzo Moreira de Oliveira (UFRGS, Brazil)
José Antonio Macêdo (UFC, Brazil)
Julio Dos Reis (IC-UNICAMP, Brazil)
Karin Becker (UFRGS, Brazil)
Kelly Braghetto (IME/USP, Brazil)
Khalid Belhajjame (Université Paris Dauphine, France)
Lucas Augusto Carvalho (UNICAMP, Brazil)
Luciano Barbosa (UFPE, Brazil)
Marco Antonio Casanova (PUC-Rio, Brazil)
Marcos Gonçalves (UFMG, Brazil)
Maria Camila Nardini Barioni (UFU, Brazil)
Maria Claudia Cavalcanti (IME, Brazil)
Mario Nascimento (Univ. Alberta, Edmonton, Canada)
Maristela Holanda (UnB, Brazil)
Mirella Moro (UFMG, Brazil)
Mirian Halfeld-Ferrari (Université d'Orleans – LIFO, France)
Olivier Corby (INRIA, France)
Pedro Eugenio Rocha Pedreira (Facebook Inc.)
Renata Galante (UFRGS, Brazil)
Renato Fileto (UFSC, Brazil)
Ricardo Ciferri (UFSCar, Brazil)
Ricardo Torres (Unicamp, Brazil)
Ronaldo Mello (UFSC, Brazil)
Sergio Lifschitz (PUC-Rio, Brazil)
Theo Haerder (Univ. Kaiserlautern, Germany)
Tiago de Melo (UFAM, Brazil)
Valéria C. Times (UFPE, Brazil)
Vanessa Braganholo (UFF, Brazil)
Vania Bogorny (UFSC, Brazil)
Vania Vidal (UFC, Brazil)
Vaninha Vieira (UFBA, Brazil)
Wagner Meira Jr. (UFMG, Brazil)
Zoubida Kedad (UVSQ, France)

Table of Contents (Full Papers)

TRiER: A Fast and Scalable Method for Mining Temporal Exception Rules

Thábata Amaral and Elaine P. M. de Sousa

[1]Institute of Mathematics and Computer Science (ICMC)
University of São Paulo (USP)
São Carlos, SP - Brazil

thabataamaral@usp.br, parros@icmc.usp.br

Abstract. *Association rules are a common task to discover useful and comprehensive relationships among items. Our interest is to find exception rules, i.e. patterns that rarely occur but have critical consequences. Existing approaches for exception rules usually handle Itemset databases and are unfeasible for mining large ones due to high computational complexity. We thus propose* `TRiER` *(TempoRal Exception Ruler), an efficient method for mining temporal exception rules that not only discover unusual behaviors and their causative agents, but also identifies how long consequences take to appear. We performed an extensive experimental analysis in real data and results show* `TRiER` *is faster and more scalable than existing approaches while finding meaningful rules.*

1. Introduction

Time series occur in countless domains including economy, health and agribusiness. They are continuously produced and stored, generating a large volume of data. This scenario motivates the development of effective and efficient data mining approaches to process and analyze such data. We are particularly interested in association rules mining [Agrawal et al. 1993]. This task has many practical applications due to its simplicity in expressing how humans learn new knowledge. The common purpose is to discover frequent and reliable relationships, called strong rules. An example of strong rule is "with the help of antibiotics, the patient tends to recover" and it is noted as $antibiotics \rightarrow recovery$. Also, We call $antibiotics$ as antecedent of the rule and $recovery$ as consequent.

Although it may be of interest when the expert wants to find unobserved frequent patterns, it is not applicable to detect hidden infrequent ones. Few approaches deal with the extraction of infrequent knowledge. We focus on those proposals that allow obtaining unusual and unexpected information, called exception rules. This class of rule express patterns that contradict the common belief [Suzuki 1996, Hussain et al. 2000]. It means that for searching an exception we have to find an attribute, i.e. an agent, changing the consequent of a strong rule. An example of exception rule is "antibiotics combined with staphylococci may lead to death" and can be noted as $antibiotics\ and\ staphylococcus \rightarrow death$, where $staphylococcus$ is the agent causing the exception.

Mining both rules explain the agents perturbing typical behaviors. As a result, previous control actions can be implemented in scenarios where unexpected agents have critical consequences. An example of prior control action to avoid staphylococcus is to properly maintain the patient hygiene in the hospital. Existing approaches for exception rules usually handle Itemset databases, where transactions have no temporal organization.

However, temporality may be inherent to some real contexts and should be considered to improve the semantic quality of results. Moreover, these approaches have high computational cost (of exponential order), becoming ineffective for mining large datasets.

Aiming to overcome these drawbacks, we propose TRiER (TempoRal Exception Ruler), a fast and scalable method for mining temporal exception rules that not only discover unusual behaviors and their causative agents, but also identifies how long consequences take to appear. TRiER differs from related methods since it handles multivariate time series, proposes a more significant support measure to limit the number of generated rules and discovers rules with greater semantic relevance. We performed an extensive experimental analysis in real data to verify the practical applicability of TRiER. Our results reveal our method is faster and more scalable than existing approaches while finding meaningful temporal exception rules.

The remainder of the paper is organized as follows. Section 2 summarizes background concepts. Section 3 presents related work and existing formalizations for associations rules, sequence mining and exception rules. In Section 4 we describe TRiER. Section 5 presents our experimental study comparing TRiER with related methods. Finally, Section 6 concludes with the contributions of the paper.

2. Background

2.1. Time Series

Time series is a sequence of time-ordered observations with regular time intervals between each pair of observations [Mitsa 2010]. A time series is defined as $S = \{s_1, s_2, \ldots, s_m\}$, where s_i for $i \in \{1, \ldots, m\}$ corresponds to the occurrence of s_i at time t_i. A univariate time series is created by only one underlying variable. A multivariate time series, on the other hand, is created by observations of several variables at each time t_i. An examples is a weather time series with variables of rainfall, maximum temperature and air humidity. We formally represent each observation s_i of a multivariate time series as $s_i = \{s_{i1}, \ldots, s_{iD}\}$, where s_i is a vector with m values that corresponds to D dimensions of the time series.

Time series discretization is the process of mapping continuous values into discrete ones, aiming to provide a suitable and concise representation of the time series for data mining tasks. The simplest discretization methods apply the equal-width and equal-frequency approaches. Equal-width methods divide the values so that all ranges have the same size. In contrast, equal-frequency ones divide the values so that ranges have the same frequency of values. Intuitively, to mine infrequent patterns, the equal-width approach is more appropriate. The reason is equal-frequency discretization creates irregular size ranges, grouping rare items to produce ranges with the same frequency of values. This grouping assumes all values in the range have the same probability of occurring, thus diluting unusual patterns. Our preliminary empirical studies corroborated this intuition.

2.2. Association Rules

Given a set I (set of items) and a database DB composed of a set of transactions T, each one being a subset of I, association rules are implications in the form $X \rightarrow Y$ that relate the presence of itemsets X and Y in transactions of DB, assuming that $X, Y \in I$, $X \cap Y = \varnothing$ and $X, Y \neq \varnothing$. We call X as **antecedent** and Y as **consequent** of the rule. The classical measures to assess association rules are support ($supp$) and confidence

($conf$). Support calculates the number of times the rule occurred ($freq$) considering the total number of transactions T in the database. Equation 1 defines the support of $X \rightarrow Y$.

$$supp(X \rightarrow Y) = \frac{freq(X \cup Y)}{T} \tag{1}$$

Confidence measures the probability of the consequent occurs in transactions that also contain the antecedent. Equation 2 represents the confidence of $X \rightarrow Y$.

$$conf(X \rightarrow Y) = \frac{freq(X \cup Y)}{freq(X)} \tag{2}$$

Given the minimum thresholds of support ($minsupp$) and confidence ($minconf$) informed by the user, we say $X \rightarrow Y$ is frequent if $supp(X \rightarrow Y) \geq minsupp$ and confident if $conf(X \rightarrow Y) \geq minconf$. Moreover, $X \rightarrow Y$ is **strong** if it is frequent and confident. An alternative framework to support-confidence was proposed in [Berzal et al. 2002] where the accuracy is measured by means of certainty factor (cf).

Certainty factor solves some of the confidence drawbacks. In particular, the support-certainty factor framework reduces the number of obtained rules, filtering those corresponding to statistical independence or negative dependence. As a consequence, extracted rules are stronger than those obtained with support-confidence. Certainty factor ranges from -1 to 1 and measures how our belief that Y is in a transaction changes when we are told that X also is. Positive values indicate our belief increases, negative values mean it decreases and 0 means no change. Analogously, we say $X \rightarrow Y$ is certain if $cf(X \rightarrow Y) \geq mincf$, where $mincf$ is the minimum threshold for certainty factor informed by the user. Equation 3 defines the certainty factor of $X \rightarrow Y$.

$$cf(X \rightarrow Y) \begin{cases} \dfrac{conf(X \rightarrow Y) - supp(Y)}{1 - supp(Y)} & \text{if } conf(X \rightarrow Y) > supp(Y) \\ \dfrac{conf(X \rightarrow Y) - supp(Y)}{supp(Y)} & \text{if } conf(X \rightarrow Y) < supp(Y) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

3. Related Work

Traditional methods for association rules relate items disregarding their occurrence orders [Agrawal et al. 1993]. However, time information is relevant to some applications and should be analyzed to better understand semantic issues. Thus, sequence mining was introduced in Agrawal and Srikant 1995 to find sequences of items considering the order they occur. There are several approaches for sequence mining in literature. We thus discuss some of the most explored ones and those closely related to our work.

Classic sequence mining algorithms were based on *Apriori*, like *GSP* (Generalized Sequential Patterns) [Srikant and Agrawal 1996]. Apriori-like algorithms usually include candidate generation and validation steps, performed with support and confidence counts. A limitation of Apriori-like approaches is the need for support calculations through a complete database scan, comparing each item with all transactions. Aiming to reduce

the processing time, approaches based on vertical format, as *ECLAT* [Zaki 2000] and *SPADE* [Zaki 2001], and pattern growth, as *Prefix-Span* [Pei et al. 2001] were proposed.

Prefix-Span (Prefix-projected Sequential Pattern Growth) [Pei et al. 2001], for example, does not require a candidate generation step. The database is recursively projected into smaller parts and frequent sequences are directly extended to create larger ones. Sequence mining algorithms, in general, look for sequences considering the order items occurred but do not establish cause and consequence relationships. In addition, they are pseudo-polynomial in time complexity [Dong 2009] and implement a non-discriminatory support measure. Therefore, when applied to mine sequences with low support thresholds, these algorithms generate an overwhelming amount of sequences in unfeasible time.

Low support thresholds concern the mining of infrequent patterns, that might be relevant to some applications. We focus on those proposals that allow obtaining some unexpected and uncommon information, called exception rules. In general, these approaches are able to manage rules that, being infrequent, provide a specific domain usually delimited by a strong rule. Exception rules were first defined as rules that contradict the user's common belief. It means that for searching an exception we have to find an attribute (also called agent) that changes the consequent of a strong rule. In general terms, the kind of knowledge an exception rule discovers can be interpreted as follows.

*"X **strongly** implies Y (and not E), but in conjunction with E, X **does not** imply Y"*.

Classical works on this research topic are presented in Suzuki 1996 and Hussain et al. 2000. According to Suzuki 1996, an exception rule is formally defined as:

$$X \rightarrow Y \text{ (high } supp \text{ and high } conf - \text{strong rule).}$$
$$X \wedge E \rightarrow \neg Y \text{ (low } supp \text{ and high } conf - \text{exception rule).}$$
$$X \nrightarrow E \text{ (high } supp \text{ and high } conf - \text{reference rule).}$$

$X \rightarrow Y$ is a strong rule and indicates a common behavior. $X \wedge E \rightarrow \neg Y$ is an exception rule and shows that the presence of item E has modified the expected consequent of the strong rule. $X \nrightarrow E$ is a reference rule and determines the antecedent of the rule should have no association with the agent causing the exception. A similar definition is presented in Hussain et al. 2000. The difference lies in the reference rule, where the agent causing the exception may have association with the unexpected behavior ($E \rightarrow \neg Y$).

Some different definitions for exception rules do not follow the schema of mining exceptions using the three previous properties, as presented in Daly and Taniar 2008. These approaches focus on finding unusual and contradictory behavior while Suzuki 1996 and Hussain et al. 2000 also allow inferring which agent caused the exceptional behavior.

The Exception Rule Search Algorithm (*ERSA*) [Calvo-Flores et al. 2011] is based on definition introduced in Suzuki 1996 but it does not employ a reference rule. Authors argue this rule does not offer a semantic enrichment when defining exceptions and reformulate the definition as the pair of common sense and exception rules as shown below:

$$X \rightarrow Y \text{ (frequent and certain} - \text{strong rule).}$$
$$E \rightarrow \neg Y \text{ (certain in the domain of the strong rule antecedent} - \text{exception rule).}$$

ERSA basic operation is given as follows. The database is first converted into a binary format to search the set of frequent items and select strong rules. For each strong rule,

the algorithm generates exceptions possibilities, applying confidence or certainty factor to filter relevant rules. Its complexity is $O(Trl2^l)$, where T is the number of transactions in the database, l is the quantity of obtained items and r is the number of rules. Most recent methods are based on fuzzy logic, as *FERSA* (Fuzzy Exceptional Rule Search Algorithm) [Ruiz et al. 2016]. This method is similar to *ERSA* in terms of time complexity and definition, applying fuzzy logic to avoid the problem of data imprecision.

To the best of our knowledge, there is no method in literature to find exception rules caused by more than one agent. Recall staphylococcus example: although a combination of agents may cause the patient death, related methods only recognize staphylococcus as causative agent. Moreover, those methods do not consider temporality in the rules discovering process. We thus propose `TRiER`, a fast and scalable method for mining temporal exception rules that may include more than one agent causing the exception. We also propose a more selective support measure which reduces the number of sequences.

4. Proposed Method

We propose `TRiER` (TempoRal Exception Ruler), a fast and scalable method for mining temporal exceptions and their corresponding strong rules. Mining both rules allows a better understanding of the agents that perturb the strong rule's usual behavior. In general terms, the kind of knowledge `TRiER` discovers is described as follows.

"X strongly implies Y (and not E) after a while. But X and E (even if E occurs later) implies something different from Y after another while".

`TRiER` effectively deals with univariate and multivariate time series and implements the concepts of sliding window and time lag. A **window** (w) of a time series S is a block of events that occurs in a continuous interval, starting at time t_b and ending at time t_e, such that events t_b and t_e belong to S. A **sliding window** ($W[i]$) at position i of time series S is a window starting at time i and ending at time $i + w_s$, where w_s is the number of consecutive time instants composing $W[i]$ (sliding window size).

Time lag (t_{lag}) is a delay between the beginning of the antecedent and the end of the consequent. Time lag follows the temporal granularity of time series, such as days, months or years. We divide our method in three main steps: (1) *Sequences Generation*, (2) *Rules Discovery* and (3) *Exception Rules Mining*, as illustrated in Figure 1.

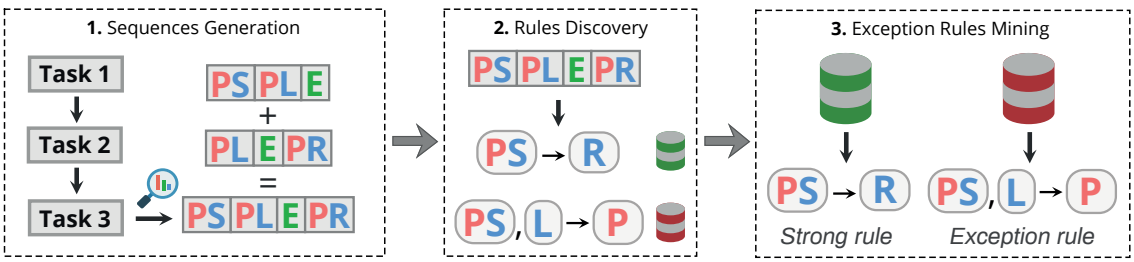


Figure 1. `TRiER` overview.

In *Sequences Generation* we create sequences that will form strong and infrequent rules. For this, we apply two parameters in the discretized dataset: the sliding window size ($W[i]$) and the minimum support for a sequence to be considered frequent ($minsupp$).

This step is composed of three main tasks. In the first one, we calculate the support of each discretization symbol, selecting those that meet the $minsupp$ threshold. In second task, we generate sequences with one observation and D dimensions. To generate a frequent sequence with D dimensions, we join two frequent sequences with $D-1$ dimensions. In last task, we obtain sequences with multiple observations and dimensions. We thus join two frequent sequences with $k-1$ observations to form a candidate with k observations, considering the order of the observations. This task ends when we reach the sliding window size. In Figure 1, to form the sequence PS, PL, E, PR we join two sequences with three observations, as PS, PL, E and PL, E, PR, so that the last observations of the first sequence are exactly the same as the first observations of the second sequence.

Classical support measures only consider the number of series in which a sequence occurs. In other words, there is no difference if a sequence occurs innumerable times or only once in a series, since they have the same support. Therefore, we propose a more selective support measure to better estimate the contribution of a sequence. Our measure considers the number of times a sequence occurs and the number of different series containing that sequence. Equation 4 formalizes our measure: L is the number of time series in the database, m is the number of observations in a series, n is the number of time series in which a sequence occurs and f is the number of times a sequence appears.

$$supp = \frac{f}{L \cdot m} \cdot e^{\left(\frac{n}{L}-1\right)} \tag{4}$$

`TRiER` was implemented in Java and our strategy to improve the efficiency of sequences generation step is described as follows. Initially, time series are stored in a map structure, called **Hash Map**. This structure is composed of key and value sets that permit to return objects quickly, with constant complexity. Keys are formed by items and an item indicates an observation in a time series. The value of each key is a **Tree Set** composed of pairs in the format $(i{:}j)$, where j is a time series identifier containing the key in the instant i. Tree Set implements a red black tree to sort $(i{:}j)$ pairs and performs the most common operations in logarithmic complexity.

In *Rules Discovery* sequences are analyzed according to the time lag possibilities they can assume. In Figure 1 we can extract rules with time lag 1, 2 and 3 from PS, PL, E, PR. Sequence $PS, L \rightarrow PR$, for instance, has time lag 2 because PR occurs two units of time after L. Rules possibilities are evaluated with the following parameters: (I) minimum support of the strong rule $(minsupp_{str})$, (II) minimum support of the infrequent rule $(minsupp_{inf})$, (III) maximum support of the infrequent rule $(maxsupp)$, (IV) minimum confidence $(minconf)$ and minimum certainty factor $(mincf)$. We classify a rule as **strong** if its support-confidence or support-certainty factor meets these minimum thresholds. To classify **infrequent rules** we apply a $maxsupp$ threshold to assure the maximum support of infrequent rules does not exceed the $minsupp$ of strong rules.

We use certainty factor because it is more restrictive than confidence and reduces the number of rules that may be huge in some cases. We do not exclude the confidence because it composes certainty factor, besides basing comparison with other methods. Once the rules are classified in strong and infrequent we start *Exception Rules Mining*, considering a slightly different definition of exception rules, as follows.

$$X \rightarrow Y \text{ (frequent and confident/certain} - \text{strong rule)}$$

$X \wedge E \rightarrow \neg Y$ (infrequent and confident/certain $-$ exception rule)

The main difference of our definition to literature ones is that we consider an exception rule as infrequent. We argue those rules are naturally atypical and indicates an unusual behavior, so the maximum support of exception rules should not exceed the minimum support of a strong rule. In *Exception Rules Mining* we analyze strong rules and look for infrequent rules that meet the following restrictions: (I) the antecedent of the infrequent rule must contain the antecedent of the strong rule and (II) the consequent of the infrequent rule must not contain or be included in the consequent of the strong rule. The infrequent rule satisfying those constraints will constitute an exception to the analyzed strong rule. As the infrequent rule $PS, L \rightarrow P$, illustrated in Figure 1, meets the previous constraints, it constitutes an exception to the strong rule $PS \rightarrow R$.

We remark the agent causing the exception can occur at a different time from the antecedent, as long as it occurs before the rule consequent. We also enable exceptions to be caused by more than one item. Algorithm 1 summarizes the main idea of our method.

Algorithm 1: TRiER exception rules mining

Input : Frequent sequences, $minsupp_{str}$, $minsupp_{inf}$, $maxsupp$, $minconf$ or $mincf$

Output: Temporal exceptions and their corresponding strong rules

```
1  begin
2    foreach sequence seq ∈ freqSequences do
3      for (p = 0 to p ≤ seqsize − 2) do
4      generate rules from the current sequence with time lag t_lag
5      filter generated rules using mincf or minconf
6      if (rule support ≥ minsupp_str) then
7        strong rules ← rule
8      else if (rule support ≥ minsupp_inf ∧ rule support ≤ maxsupp) then
9        infrequent rules ← rule
10   foreach strong rule str ∈ strong rules do
11     foreach infrequent rule inf ∈ infrequent rules do
12       if (str antecedent ⊂ inf antecedent ∧ str consequent ⊄ inf consequent) then
13         exception rules ← strong rule and its exception rule
```

TRiER complexity is divided in sequences generation and exception rules. Sequences generation is the most costly step and is $O((w_s L m D^{w_s})^2 log(Lm))$, where w_s is the sliding window size, L is the number of time series in the database, m is the number of observations in a time series and D is the number of dimensions in each observation. The exception rules complexity is $O(yw_s)^2$, where y is the number of generated sequences.

Although sequences generation has a high complexity, when mining larger sequences with low support our results are obtained at feasible times. TRiER overcomes drawbacks of related methods as it handles multivariate time series, proposes a more significant support measure, identifies exceptions caused by more than one item, applies a sliding window in the mining process and detects rules occurring in a time lag.

5. Experimental Analysis

Our experimental results on real data show the usefulness, effectiveness and efficiency of TRiER when compared to related methods. We first analyze the relevance of sequences

generated by TRiER and classical algorithms, as *GSP* and *Prefix-Span*. Finally, we compare TRiER with the baseline for exception rules, *ERSA*. Experiments were performed on an Intel Core i7, 3.40 GHz computer with 16 GB of RAM and a SATA hard disk.

5.1. Agriculture Data

Our experimental study on real data aims to investigate how computational methods can consolidate automatic means of discovering frequent patterns and exceptions in agriculture under climate influence. We chose agriculture because it plays a fundamental role in the economy of several countries. In Brazil, for example, agribusiness accounts for 23% of GDP and 40% of the labor. Mining exception rules in agriculture may help us to identify which climatic conditions most affect crops and how quickly implications arise. As a result, previous control actions could be carried out in regions with similar characteristics to minimize severe impacts caused by extreme weather, as droughts. Our analysis focus on sugarcane, due to its importance for ethanol production and Brazil's economy.

We constructed a multivariate time series dataset including observations of climate variable and vegetation index. Vegetation index data were obtained with SATVeg[1], a tool that extracts *NDVI* time series from TERRA/MODIS satellite images. *NDVI* (Normalized Difference Vegetation Index) is a widely used index in agricultural research that represents the soil vegetative vigor. *NDVI* ranges from -1 to 1: values close to 1 indicate strong vegetative activity while negative or close to 0 values describe regions where there is weak or no chlorophyll activity. We collected *NDVI* time series of sugarcane from 2014 to 2018. Time series are composed by 60 monthly observations from the state of São Paulo, the largest sugarcane producing state in Brazil.

We also collected climate time series from National Institute of Meteorology[2] (INMET). Time series are multivariate, composed of monthly observations of rainfall, maximum and minimum temperature. We then associated climate observations with *NDVI* data within a range of 70 km (geodesic distance) from the corresponding weather station. Figure 2 illustrates sugarcane regions we analyze (green points) and the weather stations monitoring them (purple stars).

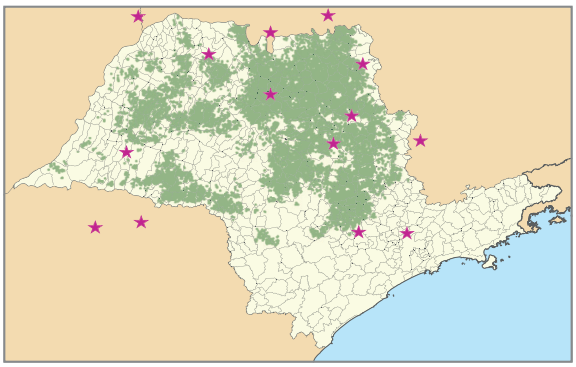


Figure 2. Weather stations and sugarcane areas in São Paulo

Table 1 describes the resulting dataset. $MaxTemp$ and $MinTemp$ are abbreviations of maximum and minimum temperature, respectively.

[1]https://www.satveg.cnptia.embrapa.br/satveg/

[2]http://www.inmet.gov.br/portal/

Table 1. Dataset description

Time series size (m)	Dimensionality (D)	Dataset size (L)
60 monthly observations	$MaxTemp$, $MinTemp$, *NDVI* and rainfall	$24,420$ time series

5.2. Sequences Generation

This experiment aims to investigate the efficiency of *GSP*, *Prefix-Span* and TRiER in the most costly phase of the mining process, i.e. sequences generation. Accordingly, we measure running time as we increase the window size (w_s) and the $minsupp$ threshold. As the temporal granularity of series is monthly, the window size is given in months. $w_s = 1$, for instance, means one month, $w_s = 2$ indicates two months and so on. Each algorithm was executed 5 times and Table 2 summarizes their average time, in minutes. We also tested *GSP* with $w_s = 2$ and the algorithm took on average 346 minutes in sequence mining. Thus, testing other window sizes would be impractical.

Table 2. Time for generating sequences in minutes

$minsupp$	$w_s = 2$		$w_s = 3$		$w_s = 4$		$w_s = 5$	
	Prefix-Span	TRiER	*Prefix-Span*	TRiER	*Prefix-Span*	TRiER	*Prefix-Span*	TRiER
0.05	**0.067**	7.416	**1.494**	19.166	34.324	**26.5**	811.613	**29.75**
0.1	**0.066**	3.5	**1.428**	6.916	34.319	**8.583**	806.415	**10.2**
0.15	**0.066**	2.25	**1.427**	3.833	33.906	**4.333**	784.452	**6.3**
0.2	**0.065**	1.116	**1.421**	1.916	33.346	**1.935**	735.166	**3.383**
0.3	**0.065**	0.75	1.352	**0.8**	32.744	**0.816**	648.352	**1.25**
0.4	**0.064**	0.383	1.319	**0.416**	32.380	**0.466**	565.446	**0.65**

Our analysis showed that in the mining of infrequent patterns, the equal-width approach is more appropriate. The reason is that equal-frequency discretization creates irregular size ranges, grouping rare items to produce ranges with the same frequency. This grouping assumes all values in the range have equal probability of occurring. As a result, equal-frequency discretization dilutes unusual patterns.

As we increase the window size, the performance of existing works degrades. The reason is that these algorithms implement a non-discriminatory support measure. It means that if an item occurs only once or countless times, it will have the same contribution in support calculation. As a consequence, these algorithms generate an overwhelming number of sequences, which include a large amount of irrelevant and unnecessary information

Figure 3 illustrates the significant growth in the number of sequences as the window size increases. For small windows (2 and 3) the number of sequences is feasible, 1397 and 50515 respectively. When we work with larger windows, 4 and 5, the number of sequences is immense, 1693404 and 52520544 respectively. Since the rules will come from this volume, the number of rules will also be huge. Thus, instead of creating knowledge to facilitate expert analysis, this amount of information will require a second-order data mining task to weed out irrelevant patterns and filters necessary ones.

Although our method is not superior to *Prefix-Span* for smaller windows (2 and 3), sequences are obtained at feasible times, unlike *GSP*. However, when we increase the minimum support thresholds, TRiER has the best performance in most window sizes. For larger windows the gain of our approach is meaningful. The reason is that we generate

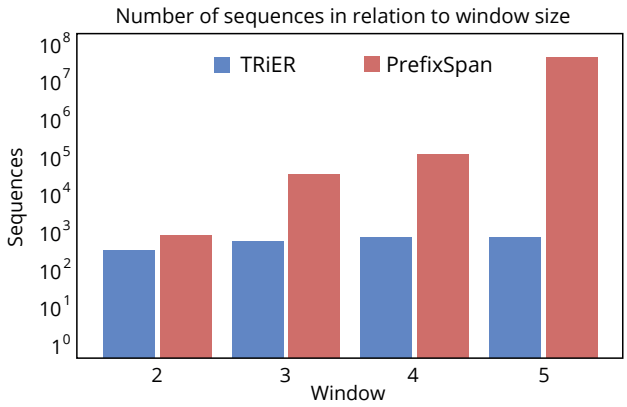


Figure 3. Sequences generated by `TRiER` and *Prefix-Span* for 5% of minsupp

far fewer sequences than previous approaches since we apply a more selective support measure. As a result, we reduce the time required to create larger sequences.

5.3. Rules Discovery

This experiment aims to investigate the effectiveness and usefulness of TRiER and the baseline method for mining exceptions, *ERSA*. We analyze results from two perspectives: the time taken to discover rules and their semantic relevance. Figure 4 illustrates time spent by TRiER and *ERSA* to discover exception rules. TRiER was tested with different sliding windows $W[i] = \{2, 3, 4, 5\}$ to analyze how this size influences on rules discovery step. Although *ERSA* generates all the possibilities of frequent itemsets, we restricted the maximum itemset size to 5 so that the time taken was not impractical.

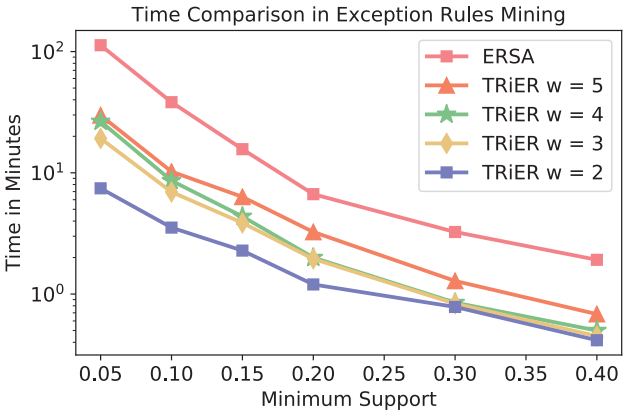


Figure 4. Time spent by *ERSA* and `TRiER` to discover exception rules

As *ERSA* generates a huge number of rules, we measured times with 60% of certainty factor for both algorithms. Rules generated only by confidence or lower values of certainty factor exceeded the computational resources available to perform the experiment. Unlike TRiER that deals with multivariate data and understands a time series is composed of 60 observations and each observation has 4 variables (dimensions), *ERSA* does not make this distinction and considers a time series as one itemset with 240 items, that is, 60 observations of each variable (dimension).

To discover exception rules, *ERSA* first generates frequent itemsets. The way it was planned *ERSA* would generate 2^{31} candidates. For that reason we simplified the maximum itemset size to 5. Then, the number of possible candidates is the number of sets with up to 5 elements we can form with the number of symbols used in discretization. This simplification reduces the number of candidates to approximately $200,000$ itemsets.

Another problem that degrades *ERSA* performance is the way it searches exceptions. The rules are stored in a single set, without distinction of strong and infrequent rules. Thus, for each strong rule, the set is scanned again to identify an exception rule. This step is one of *ERSA's* efficiency and effectiveness bottlenecks. Efficiency bottleneck because the set tends to be large and is wholly scanned every time a strong rule is found. Effectiveness bottleneck because many unnecessary and irrelevant rules can be interpreted as an exception, which can hinder and confuse the expert's analysis.

`TRiER`, in contrast, classifies discovered rules between strong and infrequent. Thus, for each strong rule the set of infrequent rules is analyzed to discover an exception that meets the premises we defined in Section 4. In other words, our method does not scan the whole rules set, but only the ones classified as infrequent, a much smaller set. In order to base the semantic analysis of the discovered rules, we present some examples of rules generated by `TRiER` and *ERSA*. Table 3 presents examples of rules mined by `TRiER`. Values are presented in intervals, due to the discretization process. Temperatures are measured in degree Celsius, rainfall in millimeters and time lag in months. Support, confidence and certainty factor relate to exception rules. Recall exception rules definition: X is the rule antecedent, Z is the agent causing the exception, $\neg Y$ is the exceptional consequent and Y is the expected consequent.

Table 3. Rules discovered by `TRiER`

X	Z	$\neg Y$	Y	t_{lag}	$supp$	$conf$	cf
$MinTemp[18, 20]$	$rainfall[0, 20]$	*NDVI*$[0.2,0.4]$	*NDVI*$[0.6, 0.8]$	1	5.09%	74.38%	69.17%
$MaxTemp[28, 30]$	$rainfall[450, 500]$	*NDVI*$[0.6, 0.8]$	*NDVI*$[0.4, 0.6]$	1	9.82%	84.53%	70.96%
$MinTemp[20, 22]$	$MaxTemp[32, 34]$ and $rainfall[100, 150]$	*NDVI*$[0.8, 1.0]$	*NDVI*$[0.6, 0.8]$	1	14.25%	85.47%	79.63%

These rules can be validated by studies conducted by Embrapa[3]. They state periods of low rainfall damage sugarcane crops and are responsible for reducing NDVI values. In contrast, intense rainfall are related to the increase in *NDVI*. Finally, high temperatures and regular rainfall are ideal for sugarcane plantations.

Rules generated by *ERSA* are semantically restricted because they only allow inferring about relationships and do not consider the temporal aspect. Table 4 presents examples of rules mined by *ERSA*.

Table 4. Rules discovered by *ERSA*

X	Z	$\neg Y$	Y	$supp$	$conf$	cf
$MinTemp[18, 20]$	$rainfall[200, 250]$	*NDVI*$[0.6,0.8]$	*NDVI*$[0.4, 0.6]$	54.13%	65.78%	59.58%
$MaxTemp[30, 32]$	$MinTemp[8, 10]$	*NDVI*$[0.4, 0.6]$	*NDVI*$[0.6, 0.8]$	39.25%	78.30%	62.25%

6. Conclusion

In this paper we proposed `TRiER`, a method for mining temporal exception rules. Our method handles multivariate time series and applies the concept of sliding window as a

[3]https://www.embrapa.br/

constraint in the sequence mining process. Sliding window enables to discover rules and exceptions that do not necessarily occur instantly. We also developed a more restrictive support measure, which considers the importance of an item in a sequence or rule. Besides support-confidence framework `TRiER` implements the support-certainty factor one. We apply certainty factor as an alternative to confidence to filter rules corresponding to statistical independence or negative dependence.

7. Acknowledgements

We thank National Council for Scientific and Technological Development (CNPq), National Council for the Improvement of Higher Education (CAPES) and São Paulo Research Foundation (FAPESP) for financial support.

References

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of SIGMOD*, pages 207–216.

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of ICDE*, pages 3–14.

Berzal, F., Blanco, I., Sánchez, D., and Vila, M. (2002). Measuring the accuracy and interest of association rules: a new framework. *Intelligent Data Analysis*, pages 221–235.

Calvo-Flores, M., Ruiz, M., and Sánchez, D. (2011). New approaches for discovering exception and anomalous rules. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 19:361–399.

Daly, O. and Taniar, D. (2008). Exception rules in data mining. *Applied Mathematics and Computation*, pages 735–750.

Dong, G. (2009). *Sequence data mining*. Springer-Verlag, Berlin, Germany.

Hussain, F., Liu, H., Suzuki, E., and Lu, H. (2000). Exception rule mining with a relative interestingness measure. In *Proceedings of KDD*, pages 86–97.

Mitsa, T. (2010). *Temporal data mining*. Chapman & Hall/CRC, Minneapolis, U.S.A.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2001). Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of ICDE*, pages 215–224.

Ruiz, M. D., Sánchez, D., Delgado, M., and Martin-Bautista, M. J. (2016). Discovering fuzzy exception and anomalous rules. *IEEE Transactions on Fuzzy Systems*, 24(4):930–944.

Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In Apers, P., Bouzeghoub, M., and Gardarin, G., editors, *Advances in Database Technology — EDBT '96*, pages 1–17, Berlin, Heidelberg. Springer Berlin Heidelberg.

Suzuki, E. (1996). Discovering unexpected exceptions : a stochastic approach. *Proceedings of RSFD 1996*, pages 259–262.

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, pages 372–390.

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, pages 31–60.

A Multi-Strategy Approach to Overcoming Bias in Community Detection Evaluation

Jeancarlo C. Leão[1], Alberto H. F. Laender[2], Pedro O. S. Vaz de Melo[2]

[1] Instituto Federal do Norte de Minas (IFNMG)
39600-000 – Araçuaí – MG – Brazil.

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
31270-901 – Belo Horizonte – MG – Brazil.

jeancarlo.leao@ifnmg.edu.br, laender@dcc.ufmg.br, olmo@dcc.ufmg.br

Abstract. *Community detection is key to understand the structure of complex networks. However, the lack of appropriate evaluation strategies for this specific task may produce biased and incorrect results that might invalidate further analyses or applications based on such networks. In this context, the main contribution of this paper is an approach that supports a robust quality evaluation when detecting communities in real-world networks. In our approach, we use multiple strategies that capture distinct aspects of the communities. The conclusion on the quality of these communities is based on the consensus among the strategies adopted for the structural evaluation, as well as on the comparison with communities detected by different methods and with their existing ground truths. In this way, our approach allows one to overcome biases in network data, detection algorithms and evaluation metrics, thus providing more consistent conclusions about the quality of the detected communities. Experiments conducted with several real and synthetic networks provided results that show the effectiveness of our approach.*

1. Introduction

The community detection problem has been much studied in the context of social networks due to its wide application in many domains, giving rise to many methods to address it [Almeida et al. 2012, Fortunato 2010, Yang and Leskovec 2015]. However, one of the major challenges related to this problem is the difficulty to evaluate the detected communities with respect to the various methods proposed in the literature. Part of this difficulty lies on the fact that there is still no universally accepted definition for the concept of community [Fortunato 2010], as well as for what we understand as being the quality of a community [Hric et al. 2014]. In general, this evaluation is done without explicitly dealing with bias on data, methods and metrics, which may lead to inconsistent results.

In order to illustrate this, let us consider the example shown in Figure 1. Specifically, Figure 1a shows a social network formed by 34 members (vertices) of a karate club interconnected by edges representing interactions between them outside the club. Originally, this network was divided into two non-overlapping communities labeled by Zachary [1977] with 16 and 18 members, respectively, each one supervised by a specific instructor. Figure 1b, on the other hand, shows the communities detected

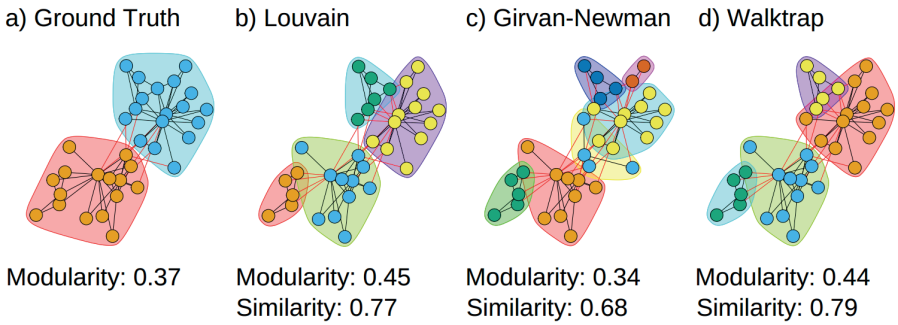


Figure 1. Example of how bias can affect community detection in social networks.

in this same network by the Louvain algorithm [Blondel et al. 2008]. Note that the community structure revealed by the Louvain algorithm is different from those presented in Figure 1c and in Figure 1d, which were respectively obtained by the Girvan-Newman [Newman and Girvan 2004] and Walktrap [Pons and Latapy 2005] algorithms, both of them considered very effective. In addition, the value of the modularity metric indicates that the communities shown in Figure 1b present a better quality with respect to their modular structure (i.e., it presents the highest modularity value). On the other hand, when comparing the detected communities with the ground truth (Figure 1a) using the Rand Index similarity metric [Rand 1971], it indicates that the network in Figure 1d is the best one, since its communities are more similar to the ones shown in Figure 1a, even though there is not a perfect match. Finally, due to some specific bias in the original network, such as sporadic interactions [Leão et al. 2018], the Girvan-Newman algorithm has not been able to identify good communities (Figure 1c), as shown by the values of the two metrics considered.

Thus, in face of these pieces of evidence pointing to opposite directions with respect to the quality of the communities in our example, a question arises on which one presents the best structure and what causes this divergence. One possible explanation to this fact is the lack of a comprehensive evaluation approach that considers multiple strategies and allows one to find out which one provides the best interpretation. More importantly, as we detail later, due to their own biases it is not always possible to find a consensus among different metrics and community detection methods on which community structure presents the highest quality. This requires a cross-checking approach involving more than two distinct evaluation strategies in order to indicate a consensus and estimate a possible bias with respect to the quality of the revealed communities.

A method that is generally employed to increase data reliability and validity is triangulation[1], which consists in using multiple methods to test a same hypothesis [Leão 2018]. Based on this idea, the main contribution of this paper is a robust approach for community quality evaluation that allows one to obtain results less prone to bias when detecting communities in synthetic and real networks.

Thus, given a network, its set of ground truth communities and a set of its communities to be evaluated, our approach allows one to overcome biases in network data,

[1]According to O'Donoghue and Punch [2003], triangulation is a "method of cross-checking data from multiple sources to search for regularities in the research data."

Table 1. Methods for community detection.

Main Method	Algorithm	ξ	References
Modularity maximization	Louvain Modularity (LM)	D	[Blondel et al. 2008]
	Greedy Optimization of Modularity (GM)	D	[Clauset et al. 2004]
	Leading Eigenvector (LE)	D	[Newman 2006]
Dynamic node labeling	Label Propagation (LP)	N	[Raghavan et al. 2007]
Removal of edges between communities	Girvan–Newman (GN)	D	[Newman and Girvan 2004]
Node closeness given by random walks	Walktrap (WT)	N	[Pons and Latapy 2005]
	Infomap (IM)	N	[Rosvall and Bergstrom 2011]

ξ: State model (D-Determinístic/N-Non determinístic).

detection methods and evaluation metrics by using distinct evaluation strategies when analyzing the quality of such communities. For this, each strategy must strongly highlight a distinct aspect of a community's quality by considering multiple metrics, detection methods and distinct datasets. For example, in Figure 1 the structural and functional aspects of the communities are represented, respectively, by their modularity and similarity with the respective ground truths. Notice that, for our purpose, the choice of the best metrics, detection methods and datasets is not important, since we are not trying to identify the best existing community, but the best one among those being compared.

The rest of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 describes our approach to community detection evaluation. Then, Section 4 analyzes the experimental results obtained by applying our proposed approach to real and simulated networks. Finally, Section 5 presents our conclusions and some considerations on future work.

2. Related Work

Although community detection has become one of the most popular and best-studied research topics in network science [Zhao 2017], the problem of validating the quality of a community derived from a real network has not received the due attention, since there is no consensus on what is meant by a good community. For example, the methods listed in Table 1 usually extract distinct communities from a given network, which are usually considered of good quality by different metrics.

In this context, there is no best metric to assess the quality of a community [Almeida et al. 2012]. Moreover, community detection algorithms are usually evaluated by correlated metrics or by the same metrics used by their optimization function, such as modularity [Fortunato 2010, Yang and Leskovec 2015]. Thus, existing work usually considers only specific aspects to assess the quality of a community, for example by measuring the structure derived from its connectivity [Newman and Girvan 2004], comparing its similarity with a ground truth community [Peel et al. 2017] or performing a comparison with a good baseline [Hric et al. 2014].

Regarding the structural aspect, popular quality metrics present strong bias when applied to networks with different sizes or number of clusters [Almeida et al. 2012, Coscia et al. 2011, Pons and Latapy 2005]. In particular cases, it is possible to as-

sess the functional aspect of a detected community by comparing it with its respective ground truths [Hric et al. 2014, Peel et al. 2017, Zaki and Meira Jr. 2014]. For Fortunato et al. [2010], this kind of evaluation involves the definition of a criterion to establish how "similar" is a community provided by an algorithm with respect to the ground truth. To address this, the authors adopt some specific indexes such as *Rand Index* and *Normalized Mutual Information*.

In addition, community detection algorithms are sensitive to different community structures, topologies or instances of a network [Coscia et al. 2011]. In this context, different approaches have been proposed with the aim of reducing the effect of biases and improving the detection of communities. For instance, Lancichinetti et al. [2012] show how to combine the communities obtained from various detection methods into a consensual one, statistically more stable and with a better structure. In a previous work, Rocha et al. [2017] described how the representation of real temporal interactions can result in biased data. More recently, Leão et al. [2018] proposed a solution to the biased data problem by directly removing noisy produced by sporadic relationships[2] found in a social network. In addition, they also showed that this kind of noise may cause errors when detecting communities.

To the best of our knowledge, the closest work to ours is the one by Yang and Leskovec [2015]. In their work, they use the correlation between distinct community definitions to evaluate their structure in large social networks. On the other hand, Lancichinetti and Fortunato [2012] seek consensus only on the structural aspect of the communities. In both works, the authors evaluate the quality of a community without aiming at a consensus involving distinct aspects or addressing any bias.

Thus, by analyzing the above works, we have not been able to identify any approach that deals with different types of bias for assessing the quality of a community. Moreover, differently from our work, the above ones do not provide a systematic and consistent strategy to produce a robust conclusion about a community's quality.

3. Proposed Approach

Figure 2 summarizes our approach. First, we provide as input a network, its ground truth communities and the set of its communities that we wish to evaluate. Next, in addition to a set of ground truth communities, we consider as further evidence the communities detected by multiple algorithms, for example, those listed in Table 1. Then, in the quantitative evaluation step, all communities are assessed by multiple structural and functional metrics, and then compared to each other to provide a set of combined evidence. Finally, we group the results produced by each algorithm in a new set of pieces of evidence in order to highlight structural and functional aspects related to the quality of each community, and compare them with those of the communities obtained by the other algorithms, as described next.

[2]In the history of interactions of a social network there are those that represent a strong relationship between two people in a community (e.g., a teacher and a student in a school) and others, result of chance, that represent intercations between people from different communities and most likely will not occur in the future (e.g., a phone call from a telemarketer) [Leão et al. 2018].

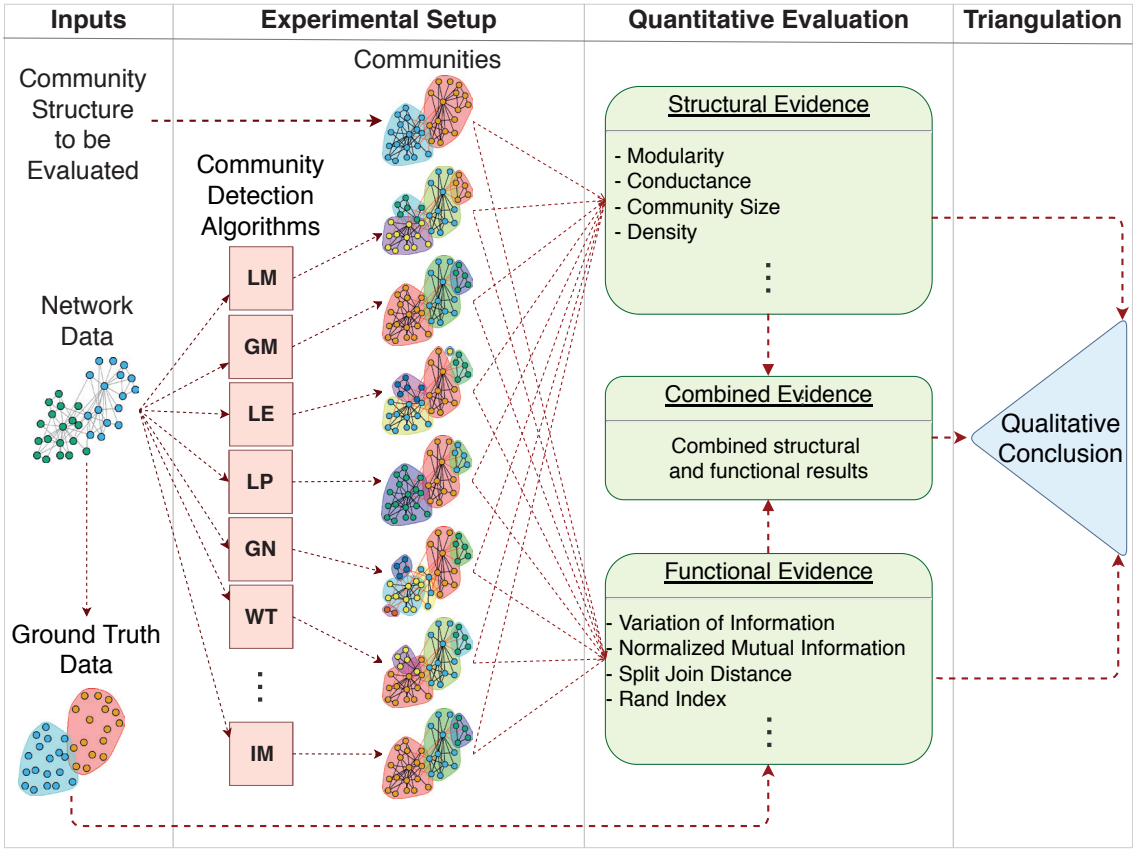


Figure 2. Overview of the proposed approach to community detection evaluation.

By using structural metrics, we quantify how much the connectivity of specific sets of nodes in the network expresses structural characteristics that are typical of real-world communities [Yang and Leskovec 2015]. For this, we take into account multiple pieces of evidence on the quality of a community expressed by metrics such as modularity, conductance and density [Newman and Girvan 2004, Yang and Leskovec 2015]. We also use specific statistics, such as the number and size of the detected communities, the variance of these values and network measures to help analyzing the results. In addition, we consider similarity (or functional) metrics such as *Variation of Information* (VI), *Normalized Mutual Information* (NMI), *Split Join Distance* (SJD) and *Rand Index* (RI) to provide some functional evidence.

We then combine structural and functional measures to compare the quality of the communities obtained by different community detection algorithms applied to the same network. For this, we assess the agreement between these measures by using the standard one-dimensional Euclidean distance. By crossing all types of evidence considered (structural, functional and combined), we capture distinct aspects of the communities' quality and conclude on the quality of the their structures by means of the consensus among all pieces of evidence. In addition, we check the effect of bias in data by controlling its source. More specifically, to minimize and estimate the effects of bias caused by noisy data, we filter the networks by using the framework proposed in our previous work [Leão et al. 2018]. Note that here "data bias" is any error generated by a community detection algorithm that might be associated with noise in the network.

Table 2. Caracterization of the networks.

Appl. Domain	Network	$\|V\|$	$\|E\|$	Δ	D	CC
Scientific Collaboration	APS	181k	852k	305	0.5	0.33
	PubMed	444k	5.5M	4869	0.6	0.36
	arXiv	33k	180k	424	3.3	-
Diseace Propagation	High School	327	5818	87	1.1k	0.44
Simulated Nets	Sinthetic	$\approx$1k	$\approx$13k	$\approx$78	$\approx$267	$\approx$0.29

$|V|$: set of vertices; $|E|$: set of edges; Δ: max degree; D: density (x10^{-4}); CC: cluster coeficient. The min degree is 1 in all networks.

4. Experimental Results

To evaluate our proposed approach, we run a series of experiments to assess the structural and functional aspects of the communities derived from five networks by applying a combination of seven algorithms based on state-of-the-art detection methods. Note that in these experiments we analyze the communities generated by each algorithm separately, considering the other ones as their baselines. In addition, experiments involving non-deterministic algorithms (see Table 1) were performed several times to ensure the reliability of the results.

4.1. Networks

Initially, we modeled as aggregate edge graphs the scientific collaboration networks (here identified by their respective datasets, namely APS, PubMed and arXiv)[3] and the contact network of secondary school students[4], which were used in previous works by Gemmetto et al. [2014] and Leão et al. [2018], respectively. Table 2 presents a general characterization of these networks.

Notice that these networks represent distinct social relationships. Thus, in the scientific collaboration networks, vertices represent researchers and there is an edge connecting two researchers if they are coauthors of a same article. In the contact network, vertices represent members of a school (for example, students or teachers) and there is an edge between two individuals if they are close to each other. We also used synthetic networks for which we have created their respective ground truth communities. These synthetic simulated networks are based on the GRM model [Nunes et al. 2017], which allows the representation of mobility networks with group (community) characteristics.

4.2. Evidence Considered

The combination of functional and structural evidence in our experiments allowed us to corroborate the quality of the ground truths as well as of the communities detected in all networks. This also made it possible to indicate the algorithm that identified the best

[3]Datasets obtained from http://homepages.dcc.ufmg.br/~mirella/projs/apoena/. APS: coauthorship network of members of the American Physical Society; PubMed: coauthorship network derived from scientific articles available on MEDLINE; arXiv: coauthorship network derived from scientific articles obtained from https://www.kaggle.com/neelshah18/arxivdataset/

[4]Datasets obtained from http://www.sociopatterns.org/datasets/.

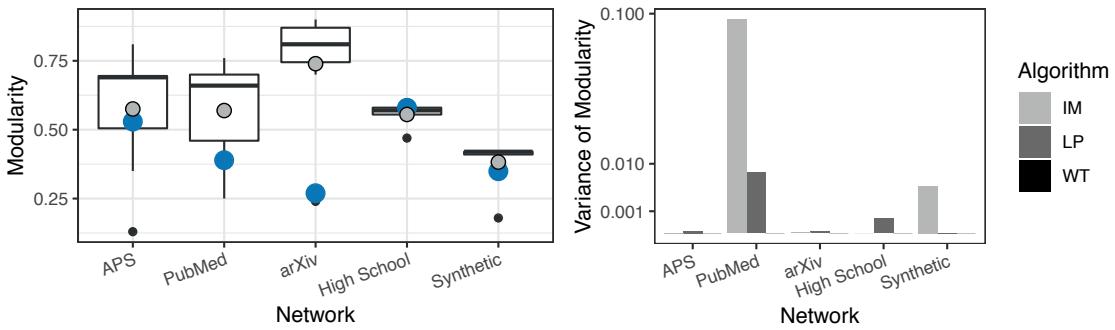


Figure 3. Modularity values for communities detected by all considered algorithms (boxplot) and for the ground truth (blue dot). $Var(Modularity)$: Variance of the modularity between replications of the detection experiments by non-deterministic algorithms (deterministic algorithms have zero variance and therefore are not presented in the right graph).

communities on the networks. For this, we first analyzed the results of each strategy individually, providing hypotheses about the quality of the communities. Then, we combined these results, verifying the consensus among the communities. In this way, we verified which hypotheses were refuted, as well as the biases identified.

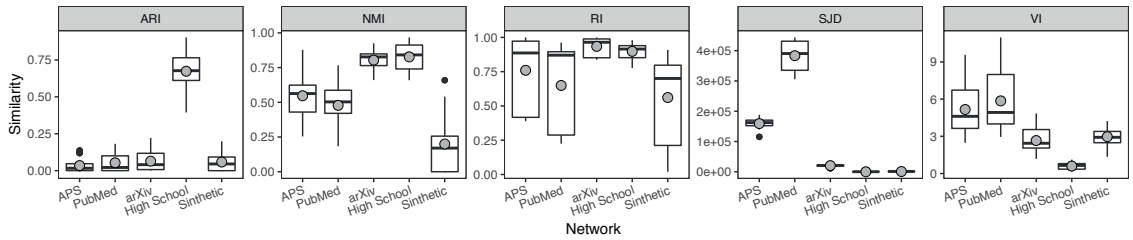


Figure 4. Similarity values between communities detected by different algorithms and measured by different metrics. Note that, especially for the metrics VI and SJD, the lower their values, the greater the similarity indicated.

4.2.1. Identifying the Best Communities

First, we analyze the structure of the communities detected by the different algorithms. Here, we note that the communities derived from the High School and arXiv networks have the best well defined characteristics. For this, we consider the following evidence: high average modularity (Figure 3, left), greater consensus on the structure of the communities (interquartil of the similarity between them, presented in Figure 4), greater confidence of the modularity value obtained in different experiments with the same non-deterministic algorithm (Figure 3, right) and small variation in the number of communities detected by these algorithms (Figure 6). However, as we shall see below, although such pieces of evidence indicate that the communities from these two networks have the same characteristics, we have not come to the same conclusion about their quality.

From a functional viewpoint, unlike the High School network, in the arXiv one there is no convergence of evidence to confirm the quality of its communities when compared with the ground truth (Figure 5). This can be considered as a disagreement with

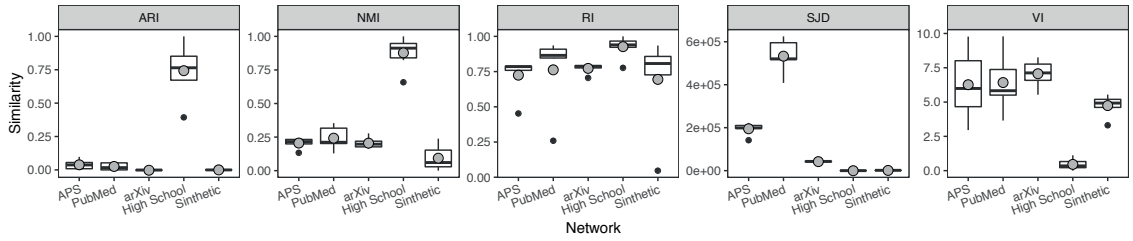


Figure 5. Values of the similarity between the ground truth and the communities detected by distinct algorithms and expressed by different metrics.

respect to the structural aspect when we compare the distance between the modularity values of the arXiv network with those of the other networks.

Note in Figure 3 (left), for example, that there is a large difference between the modularity values of the ground truths and those estimated for the detected communities. In addition, according to Figure 6, the number of communities in the ground truths is far from the number of communities actually detected in the networks. Therefore, the strength of this initial evidence has led us to the conviction that the communities detected in the actual networks are the correct ones. In addition, the confidence and the structural evidence that strongly disagree with the functional one corroborate the interpretation that the detected communities are the real ones and not those shown by the ground truths.

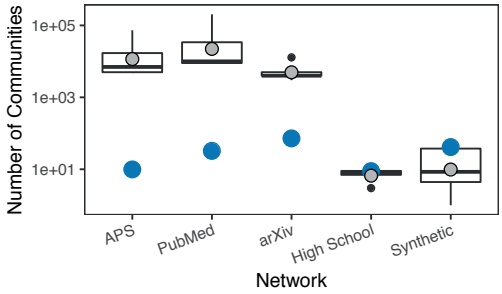


Figure 6. Number of communities detected by all considered algorithms in each network (boxplot) and in the respective ground truth (blue dot).

Note that the two sets of evidence, structural and functional, contradict each other on which communities in the arXiv network are the best ones and also on which evidence in the first set is the strongest one. Based on our approach, the possibility of bias being the cause of these divergences makes it necessary to evaluate them by using a third set of evidence (on a particular aspect) to support one of the two contradictory pieces of evidence. For this, we identified the predominant source of bias and showed that it interferes with the final conclusion.

It occurs that in the arXiv network the bias caused by the detection methods does not considerably interfere in their results, since the communities suggested by them are structurally similar. In addition, this was evidenced in this network by all structural metrics considered, whose values corroborate a high-quality community structure, as already shown in Figures 3, 4 and 6. Thus, we hypothesized that the interference bias is predominantly in the data, provoking a disagreement between structural and functional evidence, as well as the identification of false communities of high quality.

For this, we first analyze the ground truth communities of the arXiv network and then consider the meaning of these communities in this network, i.e., they are groups of researchers that publish together and predominantly in the same area of knowledge. However, it should be noted that this definition is not absolute since there may be a multi-disciplinary community with sporadic co-authorships or a community of researchers that work in the same area, but do not collaborate with each other. In both cases, ground truth communities are not very well captured by detection methods that rely on network connectivity. Thus, we consider the hypothesis that the bias that obscures the real community structure of the arXiv network is a consequence of the existence of edges and nodes that represent, respectively, sporadic collaborations and researchers that work in the same knowledge area, but do not significantly interact with each other.

Then, to test our hypothesis, we run the following bias control experiment: we removed such skewed edges and nodes using the filtering framework proposed by Leão et al. [2018] and then applied the same detection algorithm on the filtered version of that network[5]. This time we obtained new communities in which the convergence between the structural and functional metrics occurred, as indicated by its greater similarity with the ground truth communities, and having a better structural aspect, as indicated by all metrics used for this purpose. This way, we have been able to identify that the most significant source of bias in the arXiv network was its data, which shows that considering few pieces of evidence can lead to apparently convincing results, but unreliable.

In the APS and PubMed networks, data bias is also the main source of divergence between the respective ground truths and the detected communities. However, two characteristics of these two ground truths are among those that most interfere in the quality of their respective communities: the high overlap among the communities [Leão et al. 2018] and the large number of communities formed by multiple components (see Figure 6). This shows how pieces of structural evidence, such as those obtained by modularity, are insufficient to characterize this disagreement, even though the modularity of the ground truth and the communities detected in these networks have relatively high values and are close to each other. We also verify that the bias in the data caused by sporadic relationships does not considerably interfere in the detection of the communities since there was no significant improvement after filtering the networks. On the other hand, as shown in Figure 4, there was little consensus among the communities detected in these networks by the different algorithms.

In this context, in addition to identifying significant bias in the structure of the arXiv and High School networks, this phenomenon was also verified in a smaller scale among the detection methods. For example, as demonstrated for the arXiv network, its consensual community, despite its high structural quality, presents results that are considerably skewed. On the other hand, in the PubMed network and more clearly in the APS network, the average quality of the communities detected by different algorithms stands out when compared with the ground truths. In the synthetic networks and in the High School one bias had no significant interference on the convergence of the structural, functional and combined pieces of evidence of the communities' quality.

[5]The generated datasets are available by request at http://cnet.jcloud.net.br repository.

Table 3. Best detection algorithm according to distinct experiments.

Best Algorithm	LM	GM	LE	LP	WT	IM
Metrics	ARI, SJD	ARI	SJD*, VI*	RI, SJD, VI	ARI, RI	ARI, NMI*, RI, SJD, VI
Networks	PubMed, arXiv	APS	APS, PubMed	APS, Sinth.	arXiv, Sinth.	All but HS

*Metrics with the best overall value.

4.2.2. Best Detection Algorithms

Although the most modular communities are those detected by the Louvain algorithm (upper bounds shown in Figure 3, left), the modularity values of the communities detected by the Infomap algorithm are generally closer (there is a greater agreement) to those of the ground truth. In addition, Infomap provided a number of cases in which there was an agreement between the modularity of the detected communities and that of the ground truth. On the other hand, these same metrics achieved smaller values for the Louvain algorithm. In addition, the modularity of the communities extracted by different algorithms and that of the functional communities have considerably varied for most networks. We also verified how different community detection algorithms agree with each other and with the network ground truths with respect to their communities. Despite the variation in the structure of the detected communities (Figure 4), we verified a higher consensus among them than with the ground truth communities, as shown in Figure 5.

In addition to obtaining a consensus among different algorithms, our approach also identified some algorithms with distinct behavior, such as Infomap, that detected less modular communities, but in general more similar to their ground truths. Despite such divergences among the strategies, most pieces of evidence indicate the Louvain algorithm as the least biased and the one that obtained the best values for most of the structural metrics, particularly modularity. We also identified some algorithms that presented the best score when considering a specific metric. This is the case of the Louvain algorithm (LM) for modularity, and of the Infomap (IM) and Leading Eigenvector (LE) algorithms for the similarity metrics Normalized Mutual Information (NMI), and Split Join Distance (SJD) and Variation of Information (VI), respectively (see Table 3). Notice that our proposed approach is able to analyze distinct alternative solutions for the task at the hand, thus being able to identify those that provide the best trade-off.

5. Conclusions and Future Work

The main contribution of this paper is an approach to identify and reduce effect of biases when assessing the quality of communities detected by distinct algorithms. Specifically, we use multiple and diversified measurement strategies designed to capture different aspects of the quality of a community structure. For its evaluation, we carried a set of experiments using five networks (four real ones and one synthetic) and compared the results obtained by seven community detection algorithms considered the state-of-the-art in the area. In addition, we also evaluated the quality of the communities by using different strategies. In this context, our evaluation evidentiated the bias of each strategy, thus providing some consensus among them.

By doing so, we were able to sustain our hypothesis by showing that the quality evaluation of communities detected from a network must be supported by multiple pieces

of evidence. That is, given the discrepancy between the quality indicated by distinct evaluation strategies, we evidentiate that the use of a single quality metric, be it structural or functional, makes the results biased and unreliable. On the other hand, our multi-strategy evaluation approach made it possible to explain extreme values for some of the metrics considered. For example, we were able to verify the existence of bias in modularity metrics, some ground truths and network data, and some detection algorithms.

A current limitation of our proposed approach is the use of a predefined set of evaluation metrics and community detection algorithms. However, this limitation can be easily overcome by providing a configurable platform in which such features could be defined according to specific characteristics of the networks being considered. Thus, as future work, we intend to conduct a study to characterize the diversity of algorithms and metrics usually used for community detection, in order to provide insights for improving our approach. Finally, it is worth noting that the approach proposed in this paper can be adapted to other tasks besides community detection. Thus, another line of future work could be, for example, adapting this approach to assess the task of link prediction in social networks in order to provide more robust results.

Acknowledgements

Work supported by project MASWeb (FAPEMIG/PRONEX grant APQ-01400-14) and by the authors' individual grants from CNPq and FAPEMIG. Particularly, the first author would like to thank LBD/UFMG, JCLoud.net.br and LabSiCCx - Laboratório de Sistemas Computacionais Complexos (PROPPI/IFNMG, project Nr. 209/2019) for the infrastructure provided.

References

Almeida, H., Guedes, D., Meira Jr, W., and Zaki, M. J. (2012). Towards a Better Quality Metric for Graph Cluster Evaluation. *Journal of Information and Data Management*, 3(3):378–393.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Clauset, A., Newman, M. E. J., and Moore, C. (2004). Finding community structure in very large networks. *Phys. Rev. E*, 70:066111.

Coscia, M., Giannotti, F., and Pedreschi, D. (2011). A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4(5):512–546.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3–5):75–174.

Gemmetto, V., Barrat, A., and Cattuto, C. (2014). Mitigation of infectious disease at school: targeted class closure vs school closure. *BMC Infectious Diseases*, 14(1):695.

Hric, D., Darst, R. K., and Fortunato, S. (2014). Community detection in networks: Structural communities versus ground truth. *Phys. Rev. E*, 90(6):62805.

Lancichinetti, A. and Fortunato, S. (2012). Consensus clustering in complex networks. *Scientific Reports*, 2:336.

Leão, J. C. (2018). An Approach for Detecting Communities from Sequences of Social Interactions. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil (in Portuguese).

Leão, J. C., Brandão, M. A., Vaz de Melo, P. O. S., and Laender, A. H. F. (2018). Who is really in my social circle? Mining social relationships to improve detection of real communities. *Journal of Internet Services and Applications*, 9(1):20:1–20:17.

Newman, M. E. (2006). Modularity and community structure in networks. *Proc. Nat. Acad. Sci.*, 103(23):8577–8582.

Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):26113.

Nunes, I. O., Celes, C., Silva, M., Vaz de Melo, P. O. S., and Loureiro, A. A. F. (2017). GRM: Group Regularity Mobility Model. In *Proceedings of the 20th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 85–89, New York, NY, USA.

O'Donoghue, T. and K., P. (2003). *Qualitative Educational Research in Action: Doing and Reflecting*. Routledge, Abingdon, UK.

Peel, L., Larremore, D. B., and Clauset, A. (2017). The ground truth about metadata and community detection in networks. *Science Advances*, 3(5):1–8.

Pons, P. and Latapy, M. (2005). *Computing Communities in Large Networks Using Random Walks*, pages 284–293. Springer Berlin Heidelberg, Berlin, Heidelberg.

Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):1–12.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846—850.

Rocha, L. E. C., Masuda, N., and Holme, P. (2017). Sampling of temporal networks: Methods and biases. *Phys. Rev. E*, 96(5):52302.

Rosvall, M. and Bergstrom, C. T. (2011). Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLOS ONE*, 6(4):1–10.

Yang, J. and Leskovec, J. (2015). Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213.

Zachary, W. W. (1977). An Information Flow Model for Conflict and Fission in Small Groups. *Journal of Anthropological Research*, 33(4):452–473.

Zaki, M. J. and Meira Jr., W. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, Cambridge, UK.

Zhao, Y. (2017). A survey on theoretical advances of community detection in networks. *Computational Statistics*, 9(5):e1403.

In-class social networks and academic performance: how good connections can improve grades

Luiz Gomes-Jr[1]

[1]DAINF – UTFPR – Curitiba – PR – Brazil

gomesjr@dainf.ct.utfpr.edu.br

Abstract. *Understanding how different variables affect student performance is an important requirement for improving educational practices. Since humans are highly social beings, social factors should play a significant role in the academic context. This paper analyzes the impact on academic performance of social indicators such as students friendship circle and in-class clustering. The analysis is based on data from six different classes of the topic Databases taken by students of computing-related majors. We assessed students' friendship circle in terms of density (sociability) and also quality (grades) of their friends. The paper shows results with strong, statistically relevant relationships between the social factors and student performance. Among other results, the analysis indicates that (i) students with higher social capital tend to perform better, and (ii) students with friends with higher grades have better chances of recovering from a low exam grade.*

1. Introduction

Understanding the factors that influence academic performance is a challenging endeavor. There are multiple variables in different levels, from genetics to individual history, from instructor's traits to group dynamics. Determining how these variables affect each other and contribute to academic performance is a requirement for better educational methods and policies.

This paper analyzes the relationship between social connections and academic performance. The main challenge in social network analysis is to gather the data representing social connections between people. This is especially challenging in the academic environment, since there is no established repository for this type of information. Previous studies have derived this information from smart-card use around a campus or from course enrollment records. In this paper we use a more direct approach, employing self-reported relationships to build the social network graph for several classes. We argue that this approach allows for the creation of more reliable social graphs, improving the accuracy of the inferences.

This paper analyzes the impact on academic performance of social indicators such as student's friendship circle and group dynamics factors such as in-class clustering. We employ several complex network measurements to quantify these indicators (Section 2). The analysis is based on data ($n = 148$) from six different classes on the topic Databases taken by students of computing-related majors (Section 3). The paper shows results with strong, statistically relevant correlations between the social factors and student performance (Section 4). We employ a linear regression model and several statistical inferences to support the following findings:

- Classes with more connections (denser graphs) tend to have higher average grade
- Students with stronger social capital (network centrality) tend to perform better
- Students that have friends with higher average grade have better chances of improving their own grades as the term progresses

We expect that these results can help educators evaluate the role of social interactions in academic performance. The results could provide evidence to support the investment in practices that foster a healthy social environment in the academic context.

2. Related work

The lack of reliable data to build social network graphs in the academic contexts limits the research on the topic. One approach to circumvent the problem is to infer social relationships from other data sources. Yao et al. [Yao et al. 2017] resorted to time and location-stamped data of students using their smart-cards on campus facilities. From the collected data, the social relationships were inferred based on co-occurrence of events between students. The researchers analyzed data from multiple locations (e.g. cafeteria, library etc.). The analysis showed associations between the grades of students and those of their inferred social circles. The data was used to build a label propagation model to predict student grades based on the grades of their peers, achieving around 40% accuracy. The researchers do not present analysis on social network measurements and their impacts on academic performance.

Gasevic et al. [Gašević et al. 2013] assess the relationship between performance and social circles in an online education scenario. The researchers used co-enrollment in courses as a proxy for social connection. The authors emphasize that this type of data is easy to obtain. However, it is highly questionable that co-enrollment data is correlated with social bonds, especially in an online university. The research fails to show significant influence on grades of even basic centrality measures such as node degree. In this paper we use self-reported information on social bonds between students which, we believe, produces a more realistic social graph.

Castilho et al. [Castilho et al. 2014] focus on students' preferences when forming intra-class groups for course assignments. The authors use group formation data from an undergraduate course and combine it with social interaction data from Facebook. The analysis shows the importance of social status (popularity) and social interactions when students select their assignment peers. In this paper we do not focus on analysing group formation, but this could be a future research direction.

A large body of research investigates the impact of the *use* of social network sites on academic performance (see [Doleck and Lajoie 2018] for a review). The goal is to determine whether the time spent in such sites could affect student performance. Even though the majority of papers show that site usage has a negative impact on performance, the field is still far from reaching a consensus. Here we do not consider the use of social network sites and instead focus on real, self-reported relationships between students of each class.

The field of complex networks have provided tools and models for the analysis of social networks in general [Borgatti et al. 2009, da F. Costa et al. 2011]. The analysis of these networks is based on algorithms that derive measurements that capture properties

of the underlying graph (see [da F. Costa et al. 2007] for a review on network measurements). Usually, these measurements quantify properties for the nodes or for the entire graph. In this paper we apply several measurements to quantify social characteristics of students (node level) and classes (graph level). Table 1 provides a short description of the main measurements used in this paper. For formal definitions we refer the reader to [da F. Costa et al. 2007].

Table 1. Descriptions of the main measurements used

Measurement	Informal description	Level
degree	number of neighbors of a node	node
eigenvector centrality	influence/connectedness based on a node's degree and, recursively, the influence of its neighbors	node
betweenness centrality	ratio of shortest paths that pass through a node; tends to be higher for nodes that are bridges between clusters	node
closeness centrality	the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph; the more central a node is, the closer it is to all other nodes	node
average neighbor degree	average degree of the neighbors of a node	node
clustering	tendency of neighbors of a node to be connected among themselves	node
assortativity	tendency of a network to have its nodes connected to similar nodes (in general in terms of degree)	graph
average shortest path length	average of the number of nodes in the shortest paths between all pairs of nodes	graph
global efficiency	the efficiency of a pair of nodes in a graph is the multiplicative inverse of the shortest path distance between the nodes; the average global efficiency of a graph is the average efficiency of all pairs of nodes	graph

3. Data collection and cleaning

The dataset used for the analysis is based on data from six classes on the topic Databases taken by undergraduate students between 2016 and 2018. The students are from computing-related majors, namely Computer Engineering and Information Systems.

Collecting social data is a challenging task, but in this case it was simplified by the nature of the practical assignment given by the instructor to all of these Databases classes: to build and analyze the class social network. To provide students with the data needed to complete the assignment, the instructor implemented a simple social network application on which the students have to enter their data in the beginning of the term. The application collects data on friendship connections and also personal preferences (music and movies). This analysis only used data on friendship connections. The students were aware that the

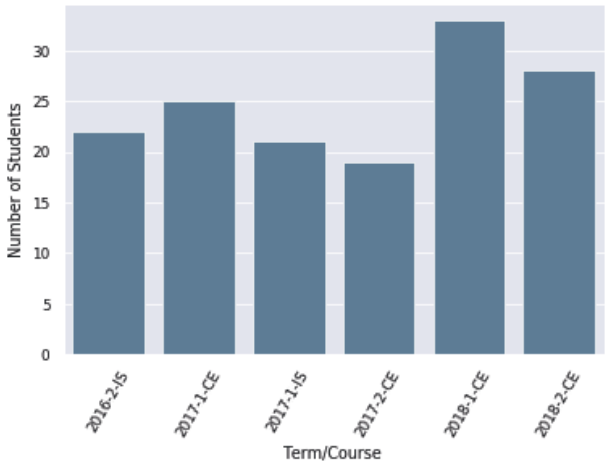


Figure 1. Distribution of students per class

data would be used for analysis. Only the instructor had access to individual data and only aggregated/anonymized data is reported here.

For each of the classes, the social graph was build and only the largest connected component was retained to represent the class social network. The graph was used to compute several complex network measurements (Table 1). Figure 2 shows an example of network for a class. Each node is a student (name anonymized) with the size of the node representing its *eigenvector centrality* score.

Figure 2. Class network example

The graph measurements were then integrated with the student performance data provided by the instructor. Students missing performance or social data (most likely class drop-outs) were excluded from the dataset. The final dataset contains 148 students distributed in classes as shown in Figure 1.

4. Data analysis

The data analysis reported here comprises four main steps: (i) exploratory analysis of class-level variables to assess the influence of social graph measurements on the average

performance of the classes; (ii) exploratory analysis of student-level variables to assess the influence of social capital measurements in the performance of students; (iii) building of a linear regression model with variables from the previous steps to quantify the influence of the variables; and (iv) analysis of the quality of connections (in term of grades) and how they help students recover from low grades. These steps are described in details in the following sections.

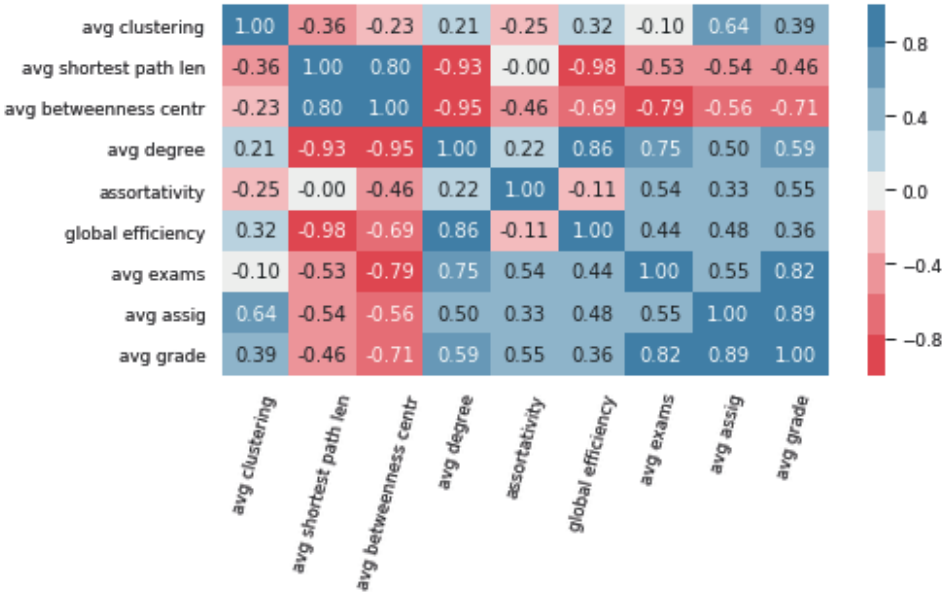


Figure 3. Correlations between class-level variables

4.1. Class-level variables

The class-level variables are intended to capture social characteristics of a given class, such as average number of friends, tendency to form clusters, etc. The class-level variables included are: average clustering, average shortest path length, average betweenness centrality, average degree, assortativity, global efficiency. The class-level variables for the grades are: average exams (average grades considering the two exams), average assignments (grade for the course assignment), and average grade (average final grade, considering exams, assignment, quizzes and exercises).

Results from the class level analysis have lower statistical relevance since there is less data (6 classes) for the inferences. The dataset shows very interesting patterns that should be interpreted carefully due to the lack of data. The heatmap in Figure 3 shows the correlation between the variables. The correlations show that, in general, the more connected a class is, the better its grades. The correlation between average degree and average grade on exams is 0.75 ($p = 0.087$), plotted in Figure 4. There are positive correlations between other variables that capture graph density, such as global efficiency and average clustering. The average shortest path length is negatively correlated with the grades for a similar reason: longer shortest paths mean less dense graphs. Average betweenness centrality is also negatively correlated with grades, which may be due to higher numbers of bridge nodes indicating isolated communities in the graph.

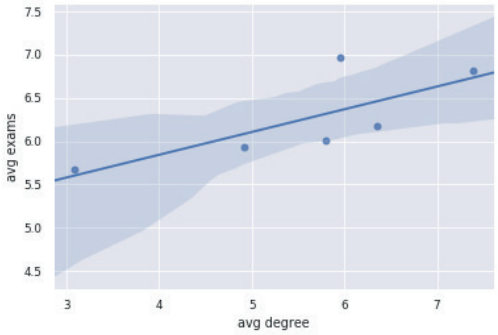


Figure 4. Class average degree vs. average grade for exams

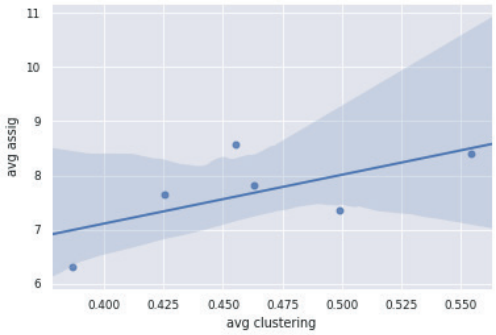


Figure 5. Class average clustering vs. average grade for assignments

Interestingly, average clustering is positively correlated with assignment grades (correlation: 0.64, $p = 0.167$, shown in Figure 5) but negatively correlated with exam grades (correlation: -0.10, $p = 0.849$). Despite the low statistical significance, it might be true that strongly connected social circles are more important for assignment grade (as explained previously, the assignments in the course are group assignments).

4.2. Student-level variables

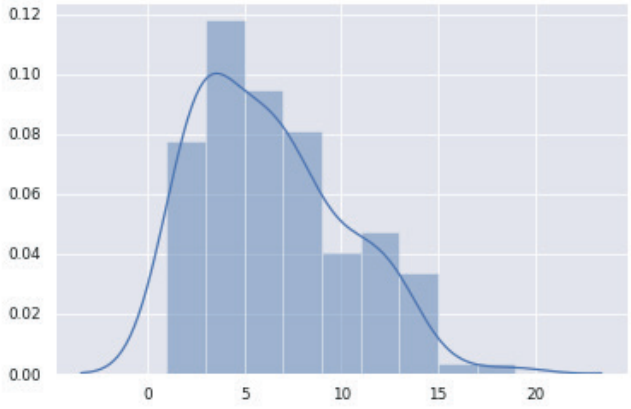


Figure 6. Degree distribution for students in the dataset

The student-level variables are intended to capture characteristics of individual's social network in the context of the class. The main student-level variables included are: average neighbor degree, betweenness centrality, closeness centrality, clustering, degree, eigenvector centrality. Results from the student level analysis have better statistical relevance since there is more data for the inferences.

We also included the class-level variables in the student data to assess whether being in a class with certain characteristics would influence individual performance. Overall, the correlations with class-level variables were weak. The strongest correlations were

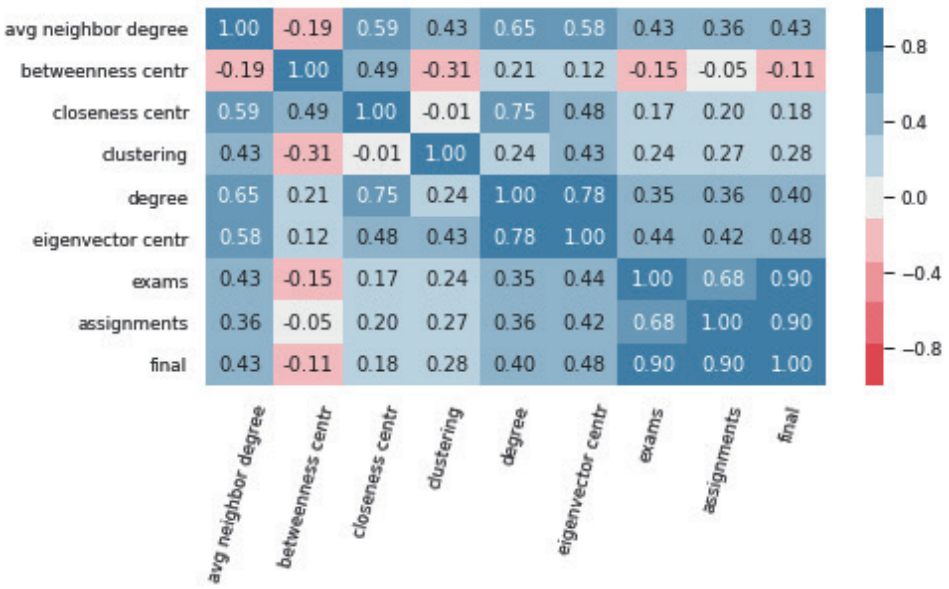


Figure 7. Correlations between student-level variables

with assortativity (0.21) and average betweenness centrality (-0.23), both considering final grades.

Figure 6 shows the degree distribution for the dataset (with the x axis representing the number of connections and the y axis their probabilities). The classes are small for any speculation about the shape of the distribution, but the figure shows the general trend in social networks of many nodes having few connections and few nodes having many connections.

The heatmap in Figure 7 shows the correlation between the student-level variables. There is a significant correlation ($p < 0,01$) between students' final grade and both *eigenvector centrality* (correlation: 0.48) and *average neighbor degree* (correlation: 0.43), suggesting that well connected students tend to have higher grades. The (weak) negative correlation between grades and *betweenness centrality* could be due to students that are in-between clusters feeling left out or having difficulties relating or being affiliated to friend groups.

Figure 8 shows a plot of individual students according to *eigenvector centrality* (normalized) and final grade. A regression line has been added for reference. The overall correlation between the variables seem pertinent. Some interesting patterns can be seen in the plot, such as a line of individuals around grade 2, probably indicating students that took too many credits or did not intend to take the course seriously from the beginning (common problems in this university). It is reasonable to assume that social factors should not play an important role in these cases. Another pattern emerges for students with very low *eigenvector centrality*, which are distributed regularly among the grades. These are probably students from other years or courses that did not know many of their peers. It is also reasonable to expect that these students would have different factors influencing their performance. Removing these two patterns from the dataset would make the correlation stronger, but since it is hard to gather data to determine the underlying reasons, we kept

all data points in the analysis.

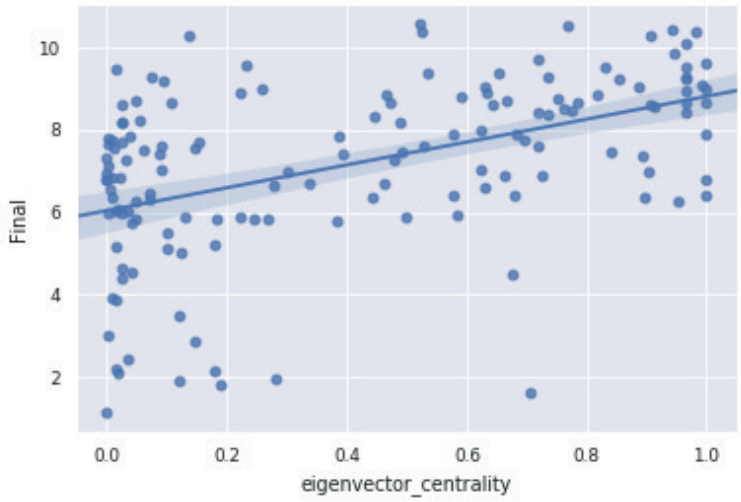


Figure 8. Student eigenvector centrality vs. final grade

4.3. Linear Regression Model

To assess the relationship between the social network variables and student performance, we built three linear regression models using *ordinary least squares* for parameter estimation. We built multiple models to evaluate whether the variables would have a different impact depending on the type of performance evaluation (grades for exams, assignments, or final grades).

To allow for comparison between models, we had the use a single criteria for feature selection. We opted to select variables with strongest combined correlation with our target variables. Therefore, for each variable, we aggregated its correlation with all targets (exams, assignments, final). We then ordered the list of variables according to strength of combined correlation. In the next step we eliminated redundant variables (e.g. for eigenvector centrality and degree, which are highly correlated, we kept only the most correlated with the target). The final list of independent variables used to build all models is: *eigenvector centrality*, *average neighbor degree*, *clustering*, *assortativity*, *average degree*, *average shortest path length*, *betweenness centrality*, *avgerage clustering*, *global efficiency*.

We used the Python module StatsModels[1] for model construction. All models achieved an R-squared value of over 0.3. Table 2 shows the results with parameter coefficients and p-values for each variable and for each target.

Several variables achieved statistical relevance for $p < 0.05$, but we focus the discussion at the more strict $p < 0.01$ level. The models confirm the influence of *eigenvector centrality* in all target variables. The 2.16 coefficient influencing final grade represents a difference of 1.45 points between students in the 25 and 75 percentile for eigenvector centrality.

[1] www.statsmodels.org

Table 2. Variables and coefficients for the three linear regression models

	Exams		Assignments		Final	
Variable	**Coef.**	**p**	**Coef.**	**p**	**Coef.**	**p**
Eigenvector Centrality	**1.76**	**0.00**	**1.98**	**0.00**	**2.16**	**0.00**
Average Neighbor Degree	**0.24**	**0.01**	0.24	0.05	0.24	0.02
Clustering Coefficient	-0.63	0.31	-0.29	0.70	-0.48	0.44
Assortativity	8.81	0.04	**18.11**	**0.00**	**15.83**	**0.00**
Average Degree	1.39	0.08	1.51	0.11	**1.97**	**0.01**
Avg. Shortest Path Length	28.56	0.04	39.87	0.02	**40.86**	**0.00**
Betweenness Centrality	-3.83	0.03	-1.76	0.40	-3.28	0.06
Average Clustering	12.81	0.05	**27.30**	**0.00**	**24.45**	**0.00**
Global Efficiency	131.35	0.05	**200.53**	**0.01**	**190.39**	**0.01**

Average neighbor degree is also significantly associated with all targets. The 0.24 coefficient represents a 0.24 increase in grade for each extra averaged degree score in a student's friendship circle.

The regression models did not confirm the impact of *clustering*. The reason might be the strong correlation between *clustering* and other variables such as *eigenvector centrality* and *average neighbor degree*.

Assortativity, *average degree* and *average clustering* are class-level variables that seem associated with academic performance. More connected classes with more homogeneous connections seem to lead to better grades.

4.4. Assessing friend's grades influence

The linear regression model showed a clear relationship between students' performance and the strength of their social connections. We now focus on analyzing the influence of the quality (in terms of grades) of the relationships.

Figure 9 shows a heatmap with the correlations between students grades (exam 1, exam 2 and final grades) and the grades of their friends (mean, maximum and minimum grades). The most important trend that can be observed is how the grades of the students tend to become more correlated as the term progresses: grades for Exam 1 have a correlation of 0.37 with friend's average for Exam 1, for Exam 2 this correlation increases to 0.52, and for the Final grade (including assignments) the correlation is 0.59. All correlations have significance level $p < 0.01$. This clearly shows that the grades of friends tend to influence a student and that influence becomes more prevalent as the term progresses (either because the student knows more about the performance of their peers or because they interact and share more knowledge).

It can also be seen from the heatmap that the strongest correlations are with the mean of the friends' grades, followed by the maximum grade among friends. This suggests that friends with lower grades tend to have a lower influence in performance.

We also calculated the correlation between the magnitude of improvement for a student $(E2 - E1)$ and his/her deficit compared to friends average $(E1\_friends\_mean - E1)$. The correlation is 0.45 $(p < 0.01)$, indicating that the larger the difference in

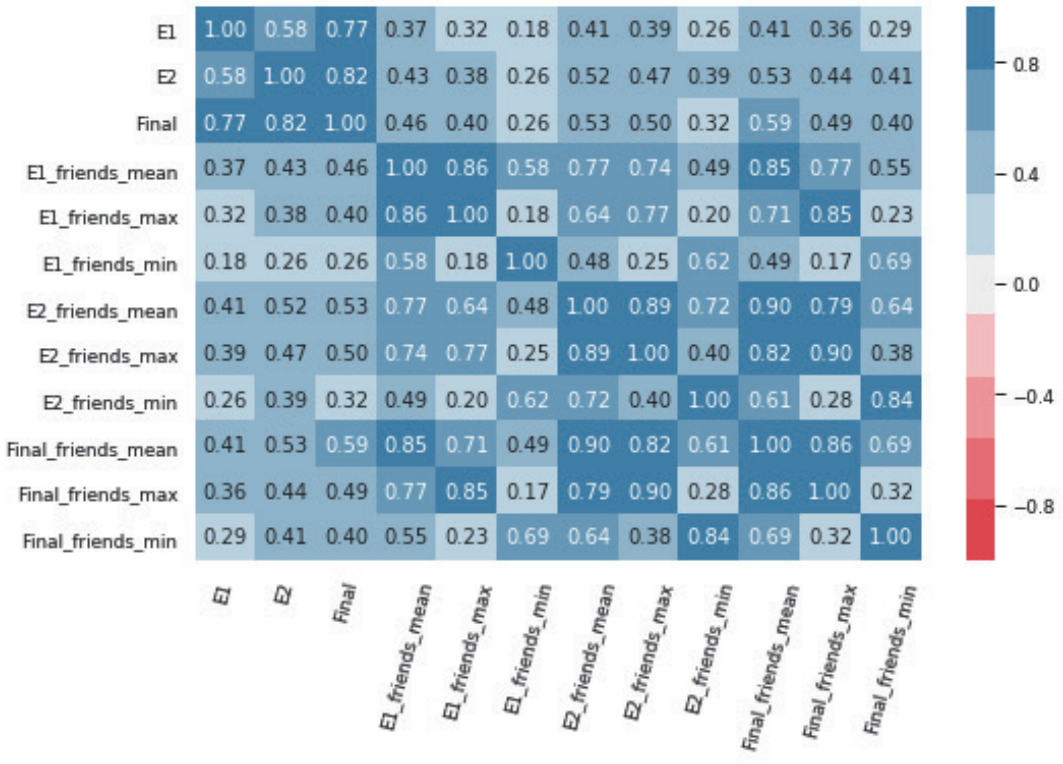


Figure 9. Correlations between student performance and their friends' performance

performance in E1, the larger the improvement for a given student.

Finally, we wanted to measure how much having a circle of friends with good academic performance can help a student in practice. We selected the students with lower grades in Exam 1 (lower than the passing grade of 6). This new dataset has $n = 62$. We then compared their performance based on the performance of their friends. Figure 10 shows the performance of the students with a low Exam 1 grade. Among these students, those with friends that performed well in the Exam 1 are represented in blue lines. It can be seen in the graph that these students tend to improve their grades. The students with friends that performed poorly in the Exam 1 (yellow lines) show no pattern for the other evaluations (some improve, some do not).

We run ANOVA tests on the averages of the students' improvement on Exam 2 for the two groups: (i) students with friends that did well on Exam 1 (grade larger than 6) and (ii) students with friends that also performed poorly in Exam 1 (< 6). Students with friends that performed poorly on Exam 1 only improved by 0.5 on average on Exam 2 ($p < 0.01$). Students with friends that performed well, in contrast, improved by 1.9 points on average – almost a 4 fold gain when compared with the other group. This suggests that having friends with good academic performance has a direct impact on students grade.

5. Discussion and Conclusion

Previous works had already established associations between students' friendship circles and academic performance. In this paper we explored similar questions using more precise data on students' social graphs. This allowed us to capture subtle but important

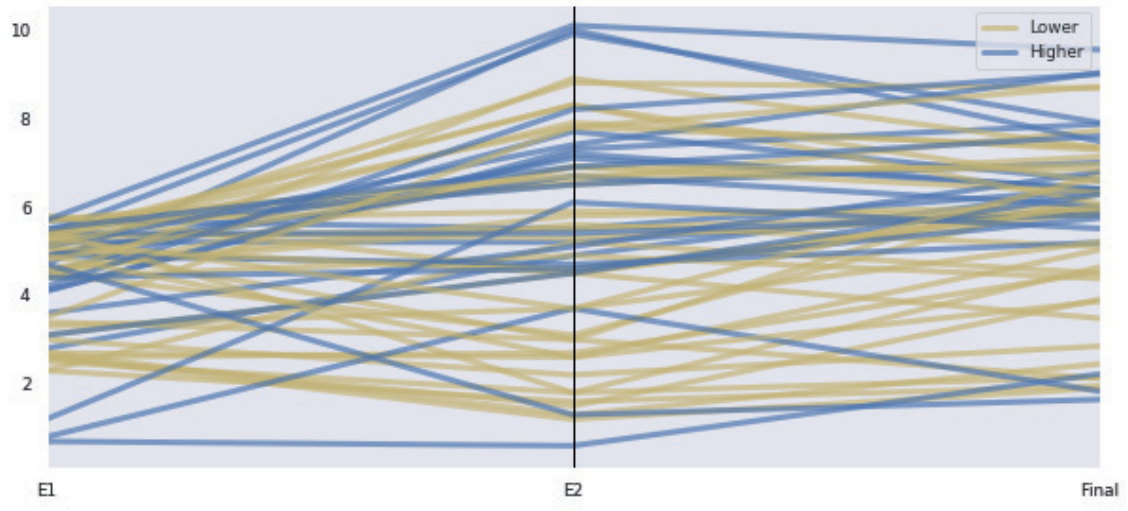


Figure 10. Performance of students that had a low grade in Exam 1

aspects of the relationship between social capital (in terms of graph measurements) and performance. In our opinion, the two most important new patterns detected are (i) the importance of complex network dynamics, and (ii) that friendship influence in performance seems to be a direct and time-dependent factor.

The importance of network dynamics in social metrics is suggested by the analysis of the correlation of social measurements with grades. The correlation between node degree (number of friends) and final grade was 0.43, which is smaller than the correlation between eigenvector centrality and final grade (0.48). Eigenvector centrality is a measurement that captures connectedness in a recursive fashion, in a model that encompasses random walk dynamics. Intuitively, the measurement produces higher values for nodes that are connected to highly connected nodes (recursively). This indicates that the relationship between social capital and grades is indeed a property of the complex social dynamics represented in the graph.

The other important aspect captured was that the influence of friends in a student's performance in not a constant. The analysis of the correlations and of the improvements from low grades show that the influence tends to grow as the term progresses. The data shows that students tend to normalize their grades with those of their friends with better performance. This does not seem to occur in the other direction – meaning that friends with poor performance have less influence on their peers.

This paper also confirms the relationship between social capital and performance. To quantify the influence we built three linear regression models, accessing each of the three grades (exams, assignment, final). All models achieved a R-squared value of over 0.3, which we consider a significant explanatory strength given that social variables are only marginal factors in student performance. In general, most of the performance of a student is more likely to be explained by variables like his/her previous performance, which is in turn associated with social, developmental and genetic factors. Of course, sociability is also highly influenced by the same factors. These variables are, however, virtually impossible to control for.

We believe that the findings presented here can help education professionals in assessing the importance of social factors in academic performance. This could then be used to guide policies for improving social interactions in the academic context.

In future work we expect to gather more data from other classes and universities to enable better inferences for class-level variables (the data collection application is available for other instructors). We also intend to analyze the factors influencing group formation and performance in the practical assignment. In terms of improving the model, we expect to integrate better criteria to identify drop-outs and correlated variables.

References

Borgatti, S. P., Mehra, A., Brass, D. J., and Labianca, G. (2009). Network analysis in the social sciences. *Science*, 323(5916):892–895.

Castilho, D., de Melo, P. O. S. V., Quercia, D., and Benevenuto, F. (2014). Working with friends: Unveiling working affinity features from facebook data. In Adar, E., Resnick, P., Choudhury, M. D., Hogan, B., and Oh, A. H., editors, *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014*. The AAAI Press.

da F. Costa, L., Oliveira Jr, O., Travieso, G., Rodrigues, F., Boas, P., Antiqueira, L., Viana, M., and Rocha, L. (2011). Analyzing and modeling real-world phenomena with complex networks: A survey of applications. *Advances in Physics*, 60:329–412.

da F. Costa, L., Rodrigues, F. A., Travieso, G., and Boas, P. R. V. (2007). Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242.

Doleck, T. and Lajoie, S. P. (2018). Social networking and academic performance: A review. *EAIT*, 23(1):435–465.

Gašević, D., Zouaq, A., and Janzen, R. (2013). "choose your classmates, your gpa is at stake!": The association of cross-class social ties and academic performance. *American Behavioral Scientist*, 57(10):1460–1479.

Yao, H., Nie, M., Su, H., Xia, H., and Lian, D. (2017). Predicting academic performance via semi-supervised learning with constructed campus social network. In Candan, K. S., 0002, L. C., Pedersen, T. B., Chang, L., and Hua, W., editors, *Database Systems for Advanced Applications - 22nd International Conference, DASFAA 2017, Suzhou, China, March 27-30, 2017, Proceedings, Part II*, volume 10178 of *Lecture Notes in Computer Science*, pages 597–609. Springer.

Uma Análise Experimental do Impacto da Seleção de Atributos em Processos de Resolução de Entidades

Levy de Souza Silva[1,2], **Gabrielle Karine Canalle**[1], **Ana Carolina Salgado**[1], **Bernadette Farias Lóscio**[1], **Mirella M. Moro**[2]

[1]Programa de Pós Graduação em Ciência da Computação - Centro de Informática (CIn) Universidade Federal de Pernambuco (UFPE)

[2]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais

{lss9, gkc, acs, bfl}@cin.ufpe.br, mirella@dcc.ufmg.br

Resumo. *Resolução de Entidades (RE) é a tarefa de identificar instâncias duplicadas em conjuntos de dados por meio de um processo de várias etapas. Um ponto em comum entre suas etapas é a seleção de atributos. Apesar de existirem trabalhos de seleção de atributos na RE, há uma falta de estudos experimentais que analisem o impacto da seleção de atributos no processo completo. Esta análise é importante pois a eficácia da RE varia conforme os atributos adotados. Assim, este trabalho aborda tal lacuna por meio de experimentos em dados reais e sintéticos de vários domínios. Por fim, os resultados mostram que a seleção de atributos afeta a eficácia da RE em até* 92%.

Abstract. *Entity Resolution is the task of identifying duplicate records in datasets by a multi-step process. A common aspect involving its steps is the attribute selection, and there is no experimental work evaluating the attribute selection impact over the complete ER process. Such an evaluation is important because the ER effectiveness varies according to the selected attributes. Therefore, we cover this gap by performing experiments over real and synthetic datasets from different domains. Finally, the results show attribute selection affects the ER effectiveness by up to* 92%.

1. Introdução

Resolução de Entidades (RE) é a tarefa de identificar instâncias duplicadas de entidades em conjuntos de dados. É um problema muito estudado com várias aplicações. Por exemplo, no contexto de Integração de Dados, a RE é aplicada para encontrar ofertas semelhantes de produtos em dados na *Web* [Barbosa et al. 2018]. Aplicações incluem Web Sites de comparação de preços que combinam dados oriundos de mais de 300 lojas, como o Buscapé[1] e o Zoom[2]. Em domínios particulares, a RE é utilizada para encontrar instâncias duplicadas em dados médicos, financeiros, governamentais, entre outros [Konda et al. 2019]. A RE também é imprescindível em áreas de Limpeza e Qualidade de Dados [Christen 2012].

O processo de RE é composto das etapas de *indexação*, *agrupamento*, *comparação*, *classificação* e *avaliação* [Christen 2012]. Um ponto em comum entre algumas etapas é a seleção de atributos, que encontra um subconjunto de atributos relevantes

[1]http://www.buscape.com.br
[2]https://www.zoom.com.br

para uma tarefa. Na RE, os atributos são utilizados para indexar, agrupar e comparar os registros, e os desafios incluem: escolher o melhor atributo para definir as chaves de bloco; selecionar o conjunto de atributos correto para agrupar as instâncias; e definir o grupo de atributos ideal e a importância de um atributo para comparar um par de registros. Alguns trabalhos abordam tais desafios e propõem métodos de seleção de atributos relevantes para etapas específicas da RE [Canalle et al. 2017, Silva et al. 2018]. No entanto, geralmente os atributos são escolhidos manualmente por um usuário especialista no domínio dos dados, o que requer tempo e aumenta o custo total do processo [Christen 2006, Christen 2012, Papadakis et al. 2015].

O problema da seleção de atributos tem recebido muita atenção, sobretudo nas áreas de Mineração de Dados e Aprendizagem de Máquina. Mas, na RE, há uma falta de estudos experimentais que avaliem o impacto da seleção de atributos em diferentes cenários e etapas do processo. Esta avaliação é necessária porque a eficácia da RE pode aumentar ou diminuir dependendo dos atributos adotados em cada uma das etapas. Por exemplo, na indexação, a eficácia do melhor atributo difere da do pior em 92% (resultados da Seção 5.1). Além disso, todos os algoritmos, métodos e funções aplicados no processo de RE dependem previamente da seleção de atributos (e.g., o algoritmo *Standard Blocking* que cria um conjunto de blocos utilizando os atributos disponíveis [Christen 2012]). No mais, os experimentos também proveem direcionamentos relacionados à eficácia da combinação de diferentes estratégias e atributos em vários cenários, bem como à importância da seleção de atributos no processo em comparação com outros fatores.

Sendo assim, o objetivo deste trabalho é analisar de forma experimental o impacto da seleção de atributos considerando o processo completo de RE. Para tal, um conjunto de experimentos são realizados utilizando dados reais e sintéticos de diferentes domínios e tipos de atributos. As avaliações experimentais são baseadas em quatro questões de pesquisa que cobrem todo o processo de RE (Seção 4). Os experimentos mostram que a seleção de atributos influencia a eficácia da RE em todas as etapas. As contribuições deste artigo são: (i) um projeto fatorial que avalia qual fator possui maior efeito no resultado da RE; (ii) uma análise da influência do atributo de indexação nos métodos de indexação *Schema-Agnostic* e *Configurations-Based*; (iii) uma avaliação do impacto da seleção de atributos nos algoritmos de agrupamento *Standard Blocking* e *Sorted Neighborhood*; (iv) uma análise do efeito da combinação de atributos na etapa de comparação; e (v) um conjunto de implementações e *datasets* disponíveis publicamente.[3]

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. O processo de RE é detalhado na Seção 3. A Seção 4 descreve a metodologia experimental. Os resultados e análises experimentais são exibidos na Seção 5 e, finalmente, as conclusões e os trabalhos futuros são apontados na Seção 6.

2. Trabalhos Relacionados

Os estudos de RE são divididos conforme suas etapas. Assim, esta seção detalha trabalhos relevantes de cada uma das etapas e aponta as principais diferenças frente ao nosso estudo.

Indexação. Existem pelo menos duas estratégias de indexação: *Schema-Agnostic* e *Configuration-Based*, as quais são comparadas em Papadakis et al. (2015). Os autores

[3] `http://www.dcc.ufmg.br/~mirella/projs/deduplica`

realizam uma avaliação experimental dessas técnicas combinadas com nove métodos de blocagem do estado da arte. Os autores concluem que a técnica *Schema-Agnostic* oferece maior robustez para definição das chaves porque é não-supervisionada e independente do domínio. Em seguida, Silva et al. (2017) avaliam as funções de indexação *Soundex* e *Suffix* combinadas com algoritmos tradicionais de RE em 11 *datasets* de vários domínios. Eles concluem que as funções *Soundex* e *Suffix* têm resultados semelhantes, sem diferença significativa em termos da *F-Measure*. Silva et al. (2018) propõem um método de seleção de atributos para a indexação que utiliza métricas como densidade, repetição e distintividade para classificar os atributos. Os autores realizam experimentos que demostram a eficácia do método proposto considerando vários domínios de dados e tipos de atributos.

Agrupamento. Depois de indexados, os registros são agrupados utilizando um dos vários métodos existentes. Em Baxter et al. (2003) os métodos *Bigram Indexing* e *Canopy Clustering* são comparados com as abordagens *Standard Blocking* e *Sorted Neighborhood*, sendo *Bigram Indexing* o mais eficiente e de melhor acurácia. Após, Draisbach e Naumann (2009) comparam os métodos de agrupamento com os algoritmos de janela deslizante (e.g, *Sorted Neighborhood*). Os experimentos demostram que os algoritmos de janela são melhores que os métodos de agrupamento no quesito eficiência. Um estudo mais completo sobre agrupamento é realizado por Christen (2012), o qual apresenta um *survey* com avaliação experimental de 12 variações de seis técnicas de agrupamento existentes. No estudo, os métodos analisados são: *Standard Blocking, Sorted Neighborhood, Suffix Array-Based Indexing, Canopy Clustering, Q-gram-Based Indexing* e *String-Map-Based Indexing*. Resultados mostram que a técnica *Q-gram-Based Indexing* é uma das mais lentas e não é adaptável para grandes conjuntos de dados, e as abordagens tradicionais são as mais rápidas (i.e., blocos e vizinhos). Por fim, Caldeira e Ferreira (2018) apresentam um método para blocagem e processamento dos blocos considerando a relevância dos termos (meta-blocagem). O método proposto supera técnicas do estado da arte em termos de eficácia e reduz o tempo de criação dos blocos pela metade.

Comparação e Classificação. No fim do processo de RE, um par de registros é classificado. Assim, as etapas de comparação e classificação são complementares, pois depois que os atributos são comparados, a classificação é baseada em um limiar de similaridade entre os atributos. Várias funções foram propostas na literatura. Cohen et al. (2003), por exemplo, analisam as funções *TFIDF, SoftTFIDF, Levenshtein, Scaled Levenstein, Jaro, Jaro-Winkler, Jaccard* e *NaiveAvgOverlap* em dados de nomes pessoais. Os resultados mostram que o melhor método para comparação de nomes pessoais é uma versão escalada do algoritmo de *Levenshtein*. Em seguida, Christen (2006) compara 20 funções de similaridade incluindo: *Soundex, Phonex, phonix, Jaro, Winkler* e *Edit Distance*. Os experimentos utilizam quatro *datasets* contendo nomes pessoais. Segundo o autor, a melhor função de classificação não é clara. Entretanto, a técnica *Simple Phonex* tem desempenho melhor que as técnicas *Complex Phonix* e *Double-Metaphone*. Além disso, os algoritmos de *Jaro* e *Jaro Winkler* são eficazes em todos os conjuntos de dados utilizados. Considerando a seleção de atributos, Canalle et al. (2017) apresentam uma abordagem que seleciona atributos relevantes para a etapa de comparação, utilizando critérios como densidade, repetição e qualidade da fonte. Experimentos são executados em dados reais e sintéticos com diferentes cenários de dados duplicados. Os resultados demonstram que a estratégia proposta seleciona atributos eficazes para a comparação em todos os cenários.

Tabela 1. Visão geral dos trabalhos relacionados frente ao nosso estudo.

Trabalho	Indexação	Agrupamento	Comparação	Classificação	Avaliação do Impacto do Atributo na Etapa
Silva et al. (2018)	X				Sim
Silva et al. (2017)	X				Não
Papadakis et al. (2015)	X				Não
Caldeira e Ferreira (2018)		X			Não
Christen (2012)		X			Não
Baxter et al. (2003)		X			Não
Canalle et al. (2017)			X		Sim
Christen (2006)			X	X	Não
Cohen et al. (2003)			X	X	Não
Nosso Estudo	X	X	X	X	Sim

Finalmente, a Tabela 1 apresenta um resumo dos trabalhos comparando-os com o nosso estudo. Em sua maioria, os estudos analisam apenas uma etapa da RE isoladamente. Por exemplo, Papadakis et al. (2015) investigam só as funções de indexação, enquanto nosso trabalho analisa em um mesmo ambiente experimental o processo completo da RE. Ademais, considerando a seleção de atributos, a maioria dos trabalhos citados não considera o impacto da seleção automática de atributos nos experimentos, pois os atributos são escolhidos manualmente por um especialista. Apesar de existirem trabalhos recentes, e.g., Silva et al. (2018) e Canalle et al. (2017), esses estudos não consideram avaliações experimentais para medir o impacto do atributo no processo. Nosso estudo difere, pois investigamos o impacto da tarefa de seleção de atributos em todo o processo de RE.

3. Processo de Resolução de Entidades

O processo de RE é dividido em *Indexação*, *Agrupamento*, *Comparação*, *Classificação* e *Avaliação*. Um ponto em comum entre algumas etapas é a tarefa de seleção de atributos. Nesse sentido, esta seção apresenta detalhadamente o processo de RE conforme Figura 1 e discute os problemas relacionados à seleção de atributos no processo.

(1) - Indexação. Inicialmente, todos os registros são indexados por um valor de chave de bloco (do termo em inglês *Block Key Value* - BKV). Para tal, atributos são escolhidos e seus valores são utilizados como chave (*Schema-Agnostic*), ou uma regra de codificação é aplicada no valor dos atributos para gerar a chave (*Configurations-Based*). No fim da etapa, com exceção das instâncias que contêm valores nulos nos atributos, cada registro está associado a um BKV. Existem diversas técnicas para criar um BKV. Entretanto, uma das principais é a *Soundex* que codifica os valores baseado na pronúncia [Christen 2012]. Outra opção é a *Suffix*, que cria sufixos de tamanho K a partir de um valor.

(2) - Agrupamento. Depois da indexação, cada BKV define grupos de registros similares. Sem o agrupamento, os registros são comparados todos com todos (i.e., *Naive Duplicate Detection*). Existem diferentes abordagens para esta etapa, entretanto os principais métodos incluem comparação por blocos e vizinhos mais próximos. O algoritmo ***Standard Blocking* (SB)** cria blocos que agrupa registros semelhantes. Logo, os registros são comparados entre eles apenas dentro dos blocos. Os grupos são definidos de acordo com o BKV dos registros. Diferente deste, o algoritmo ***Sorted Neighborhood* (SN)** combina os registros por meio de uma chave ordenada, que é similar a uma chave de bloco. Porém, antes de executar as comparações, todos os registros são ordenados pelo BKV. Depois, uma janela deslizante de tamanho $w > 1$ percorre todos os registros de D, e o primeiro registro da janela é comparado com todos os outros dentro da mesma janela.

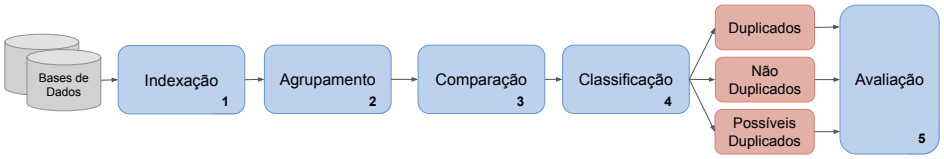


Figura 1. Processo de Resolução de Entidades (adaptado de Christen (2012))

(3) - Comparação. Após o agrupamento, os registros de cada grupo são comparados com todos os outros do mesmo grupo, par a par, utilizando medidas de similaridade. Geralmente, a semelhança entre duas instâncias é calculada comparando-se vários atributos, e quanto maior a similaridade dos atributos, mais provável que sejam instâncias de uma mesma entidade. Ou seja, uma função de similaridade calcula a correspondência entre dois valores e retorna um valor no intervalo $[0, 1]$, no qual 1 representa semelhança máxima. Funções populares são *Jaro*, *Jaro Winkler* e *Levenshtein* [Christen 2006].

(4) - Classificação. A partir da comparação, os pares de registros são classificados como *duplicados*, *não duplicados* e *possíveis duplicados*. A terceira classe necessita de uma avaliação mais detalhada, geralmente feita por um usuário especialista no domínio dos dados. Para classificá-los, existem duas abordagens: não supervisionada e supervisionada. Na primeira, os pares são classificados com base apenas na similaridade entre os atributos. Então, a forma mais simples é aplicar um limiar *(threshold)* sobre este valor. Na segunda, um conjunto de treinamento com pares verdadeiros e não correspondentes é utilizado, formando um classificador supervisionado [Christen 2012].

(5) - Avaliação. Finalmente, a etapa de avaliação analisa a eficácia dos algoritmos por meio de métricas como *Precision*, *Recall*, *F-Measure* [Christen 2012].

Seleção de Atributos na RE. Um ponto em comum entre as etapas indexação, agrupamento e comparação é a seleção de atributos. Um dos problemas é escolher atributos que proporcionem os melhores resultados para o processo em termos de eficácia. Por exemplo, considere o cenário em que as etapas de RE mostradas anteriormente são executadas sobre os dados exibidos na Tabela 2. Estes dados são oriundos de duas fontes identificadas pela coluna *ID* e referem-se a três Filmes (coluna *Filme*). Assim, o objetivo é encontrar os filmes duplicados nestas fontes. Considerando a seleção de atributos na *indexação/agrupamento*, a seguinte situação pode ocorrer:[4] (*i*) indexando por *Fonte* são definidos um bloco para os registros de *IMDb* e outro para *TheMovieDB* – Filmes 1 e 2 não são comparados pois estão em blocos distintos, o que prejudica a eficácia; (*ii*) indexando por *Diretor* ou *Título* terá maior eficácia, pois o processo compara os Filmes 1 e 2, bem como 4 e 5 que agora estão no mesmo bloco. Para os demais atributos, as situações são análogas. De forma semelhante, na *comparação/classificação*, têm-se os problemas: (*i*) comparando por meio dos atributos *Título*, *Duração* e *Gênero*, os Filmes 1 e 2, e 4 e 5 não são considerados cópia, pois não existe similaridade nos atributos *Duração* e *Gênero*; (*ii*) comparando por *Título* e *Diretor*, todas as duplicatas são identificadas corretamente porque os valores dos atributos são similares. Para outras combinações de atributos, situações

[4]Utilizando o próprio valor do atributo como chave de bloco.

Tabela 2. Identificando registros duplicados em duas fontes de dados

Filme	ID	Fonte	Título	Ano	Diretor	Duração	Gênero	Indicação
A	1	IMDb	Robin Hood	2010	Ridley Scott	2h 20m	Aventura	14
	2	TheMovieDB	Robin Hood	2010	Ridley L. Scott	2h 22m	Ação	Null
B	3	IMDb	Avatar	2009	James Cameron	2h 42m	Fantasia	12
C	4	IMDb	The Matrix	1999	Lana Wachouski	2h 20m	Ação	12
	5	TheMovieDB	Matrix	1999	Lana Wachouski	2h 16m	Ficção Científica	Null

semelhantes ocorrem. Por fim, estes problemas destacam a importância de selecionar atributos relevantes para cada uma das etapas do processso de RE.

4. Metodologia Experimental

Para avaliar a seleção de atributos na RE, as seguintes questões são adotadas: **(Q1)** Qual fator possui maior impacto no processo de RE entre a função de indexação, o atributo de indexação e o algoritmo de agrupamento? **(Q2)** O mesmo atributo de indexação é eficaz nos métodos *Schema-Agnostic* e *Configurations-Based*? **(Q3)** Os resultados da RE são eficazes utilizando o mesmo atributo de indexação nos algoritmos *Standard Blocking* e *Sorted Neighborhood*? **(Q4)** O conjunto de atributos utilizados na comparação afeta o resultado da RE? Destas questões são criados os cenários descritos a seguir (Figura 2).

Cenário 1 - Projetos Fatoriais. Este cenário considera projetos fatoriais para analisar o efeito dos fatores sobre o processo de RE. O projeto fatorial avalia o efeito de k fatores sobre uma variável y [Jain 1992]. Desse modo, um projeto fatorial $2^k r$ com $(k = 3)$ e $(r = 10)$ é executado, isto é, cada configuração do projeto é analisada sobre dez *datasets* sintéticos. As replicações são definidas com uma confiança de 95% e um erro máximo de 5%. Os fatores examinados são: o atributo de indexação, a função de indexação e o algoritmo de agrupamento, e a métrica *F-Measure* é a variável resposta Y.

Cenário 2 - Métodos de Indexação. Este cenário analisa a eficácia dos métodos *Agnostic* e *Configurations-Based*. Geralmente, apenas um atributo é adotado para indexar. Assim, o valor do atributo é utilizado como BKV, ou a codificação *Soundex* é aplicada sobre o valor do atributo para gerar os BKVs. Nas outras etapas, os algoritmos *Standard Blocking* ou *Sorted Neighborhood* e *Jaro Winkler* são aplicados. Especificamente, o objetivo é avaliar se o mesmo atributo de indexação tem resultados eficazes nos dois métodos. Para tal, a *F-Measure* é computada para cada configuração de indexação.

Cenário 3 - Algoritmos de Agrupamento. Neste cenário a RE é executada variando tanto o atributo de indexação quanto o algoritmo de agrupamento, ou seja, cada atributo é combinado com cada algoritmo. Na comparação, a função de *Jaro Winkler* é utilizada. Por fim, a *F-Measure* é computada para cada atributo em cada algoritmo.

Cenário 4 - Comparação das Instâncias. Este cenário executa a RE com o algoritmo *Naive Duplicate Detection* variando o grupo de atributos na comparação. A função de *Levenshtein* é utilizada para comparar os atributos. Alguns *datasets* contêm muitos atributos. Assim, combinações de até cinco atributos que melhor descrevem a entidade são consideradas. Por fim, a *F-measure* é calculada para cada combinação.

***Datasets* Experimentais.** Dados reais e sintéticos com vários domínios e tipos de atributos são utilizados para experimentação dos cenários. Os sintéticos são criados com o

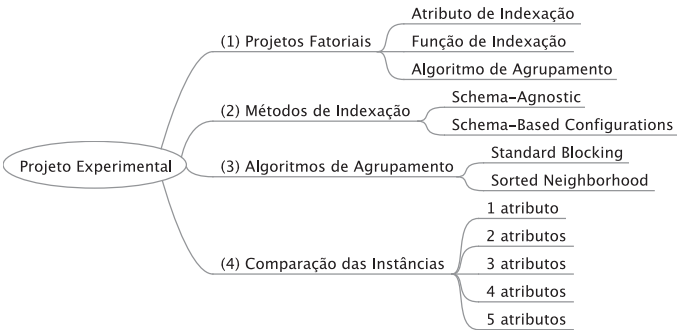


Figura 2. Projeto experimental com as dimensões avaliadas e variáveis em cada dimensão

Data Set Generator Program [Christen 2012], e os reais são extraídos do *DuDe toolkit*[5]: *CORA*, *Restaurant* e *CD Information*. Os *datasets* escolhidos são amplamente utilizados em pesquisas de RE, e.g., Canalle et al. (2017) , Christen (2012b), Papadakis et al. (2015).

5. Análise Experimental

Nesta seção, os resultados experimentais são apresentados e discutidos conforme questões de pesquisas elencadas na seção de metodologia experimental, como segue.

5.1. Avaliação do Impacto dos Fatores no Processo de RE

O resultado da RE depende da escolha de vários fatores incluindo o atributo de indexação, a função de indexação e o algoritmo de agrupamento. Assim, esta seção apresenta o resultado do projeto fatorial $2^3 10$ que avalia quais dos três fatores da RE têm maior efeito. O objetivo é responder a questão **Q1**: Qual fator possui maior impacto no processo de RE entre a função de indexação, o atributo de indexação e o algoritmo de agrupamento?

Para abordar **Q1**, os seguintes fatores e níveis são analisados: (*Fator A*) as funções de indexação *Soundex* e *Suffix*, (*Fator B*) os algoritmos de agrupamento *Standard Blocking* e *Sorted Neighborhood*, e (*Fator C*) o melhor e o pior atributo de indexação conforme a *F-Measure* (*Given Name* e *Address2*). A Tabela 3 exibe os valores da *F-Measure* para cada configuração do projeto fatorial ordenadas pelo fator *C*. O maior valor da *F-Measure* é alcançado no experimento *4*, $0,73 \pm (0,01)$, onde o atributo de indexação *Given Name* é utilizado. Por outro lado, o experimento *6* tem o resultado mais ineficaz para a RE - atributo *Address2* e *F-Measure* de $0,38 \pm (0,03)$. O melhor resultado de *Given Name* e o pior de resultado *Address2* diferem em aproximadamente $0,35$ em termos da *F-Measure* (experimentos *4* e *6* respectivamente), ou seja, quando o atributo *Given Name* é utilizado na indexação, a eficácia do processo aumenta em cerca de 92%.

Comparando os níveis da função de indexação (*Soundex* e *Suffix*) e do algoritmo de agrupamento (blocos e vizinhos), os resultados mostram que a mudança das técnicas não produz uma variação significante da *F-Measure* porque as diferenças são mínimas. Por exemplo, os resultados do processo são eficazes utilizando o algoritmo *Standard Blocking* (experimento *1*) e o algoritmo *Sorted Neighborhood* (experimento *3*), e a diferença é apenas $0,01$ da *F-Measure*. O mesmo acontece com as funções de indexação *Soundex*

[5] `https://hpi.de/naumann/projects/data-quality-and-cleansing/dude-duplicate-detection`

Tabela 3. Configurações do Projeto Fatorial

#	Fator (A) Indexação	Fator (B) Algoritmo	Fator (C) Atributo	*F-Measure*
1	Soundex	SB	Given Name	$0,71 \pm 0,01$
2	Suffix	SB	Given Name	$0,67 \pm 0,01$
3	Soundex	SN	Given Name	$0,72 \pm 0,01$
4	Suffix	SN	Given Name	$0,73 \pm 0,01$
5	Soundex	SB	Address2	$0,39 \pm 0,03$
6	Suffix	SB	Address2	$0,38 \pm 0,03$
7	Soundex	SN	Address2	$0,43 \pm 0,03$
8	Suffix	SN	Address2	$0,40 \pm 0,03$

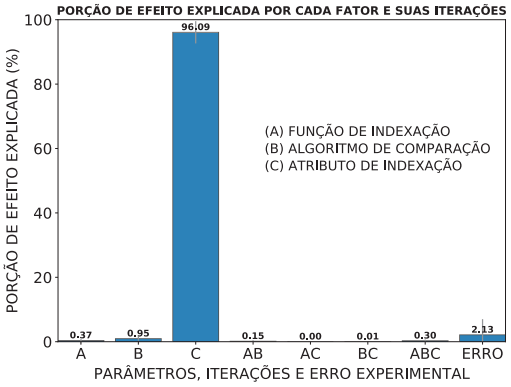


Figura 3. Efeito dos Fatores

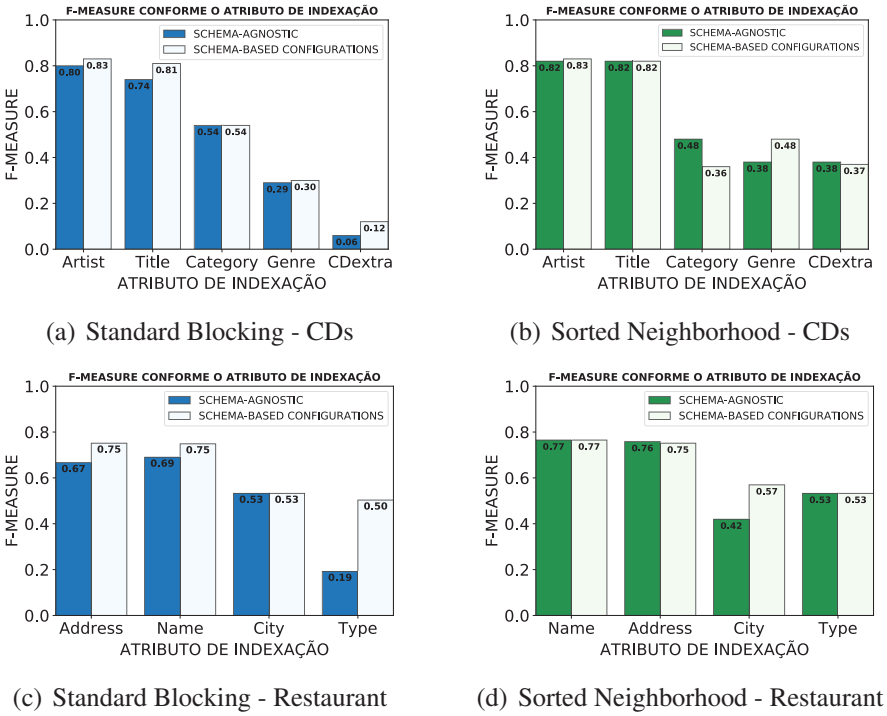
e *Suffix*, porque nos experimentos *1* e *2*, e *3* e *4*, os resultados são eficazes em termos da *F-Measure* quando comparado com os demais resultados. Por outro lado, os piores resultados da RE são alcançados quando o atributo *Given Name* é trocado pelo atributo *Address2* em qualquer combinação (experimentos *5*, *6*, *7* e *8*).

Complementando essas discussões, a Figura 3 exibe os resultados do projeto fatorial que sintetiza os experimentos da Tabela 3. O eixo-X é cada fator analisado, bem como suas iterações, e o eixo-Y a porção de efeito explicada em relação a *F-Measure* por cada fator e suas iterações. Note que o intervalo de confiança e o erro experimental são exibidos. Os resultados mostram que o atributo de indexação sozinho tem maior efeito sobre todo o processo de RE - efeito de $96,09 \pm (3,46)$, ou seja, o atributo usado na indexação explica a maior variação nos resultados em termos da *F-Measure*. Ademais, as iterações entre os fatores são insignificantes, porque o efeito explicado pelas iterações não atinge nem 1% dos resultados, isto é, as combinações de dois ou três fatores como *AB* e *ABC* não explicam os resultados da *F-Measure*, pois o atributo domina de forma isolada.

Finalmente, conclui-se que dependendo do atributo escolhido para indexar, a *F-Measure* pode aumentar ou diminuir prejudicando a eficácia da RE, pois utilizar o melhor ou o pior atributo de indexação altera significativamente a *F-Measure*. Além disso, nota-se que a eficácia da RE está mais relacionada com a escolha do atributo de indexação que é combinado com as outras funções, do que com as funções propriamente ditas, pois os resultados variam significativamente quando o atributo é alterado em ambas as técnicas.

5.2. Análise do Atributo nos métodos *Schema-Agnostic* e *Configurations-Based*

Na indexação, uma chave de bloco é atribuída a cada registro. Uma das formas é utilizar o valor dos atributos (*Schema-Agnostic*), e outra é aplicar uma regra de codificação (*Configurations-Based*). Nesse contexto, esta seção analisa o impacto da escolha do atributo de indexação nos dois métodos. Os experimentos são executados nos algoritmos *Standard Blocking* e *Sorted Neighborhood*. Especificamente, o objetivo é responder a questão **Q2**: O mesmo atributo de indexação é eficaz nos métodos *Schema-Agnostic* e *Configurations-Based*? Para responder **Q2**, experimentos em dados sintéticos e reais variando o atributo e o método de indexação são executados. A Figura 4(a)-(d) apresenta os resultados nos dados reais, onde o eixo-X exibe os atributos de indexação e o eixo-Y o valor da *F-Measure* em cada configuração de indexação para cada atributo.



(a) Standard Blocking - CDs (b) Sorted Neighborhood - CDs

(c) Standard Blocking - Restaurant (d) Sorted Neighborhood - Restaurant

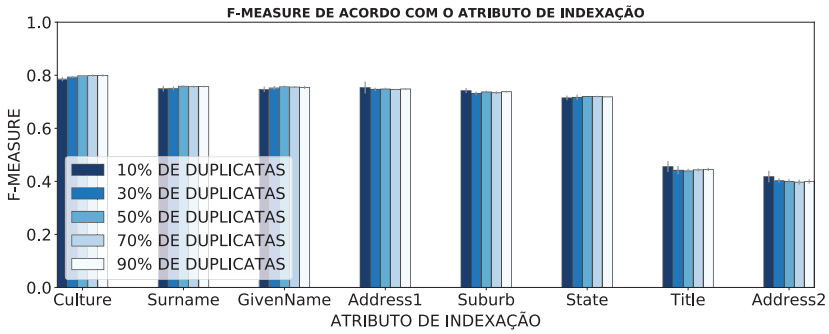
Figura 4. Atributo de indexação nos métodos *Schema-Agnostic* e *Based Configurations*.

Os resultados mostram que os atributos mais eficazes o são em ambas as técnicas de indexação. Analisando a Figura 4(a), por exemplo, o atributo *Artist* é o mais eficaz em ambas as técnicas. Ademais, comparando os algoritmos de agrupamento, os atributos mais eficazes mantêm a eficácia em ambos algoritmos com os dois métodos de indexação. Por exemplo, o atributo *Name* é eficaz em *Standard Blocking* e em *Sorted Neighborhood*, independente da indexação, conforme Figura 4(c)-(d). Assim, a eficácia da RE está mais relacionada ao atributo do que à função de indexação, pois em ambas as funções os resultados são semelhantes. Porém, o valor da *F-Measure* difere quando o atributo é alterado.

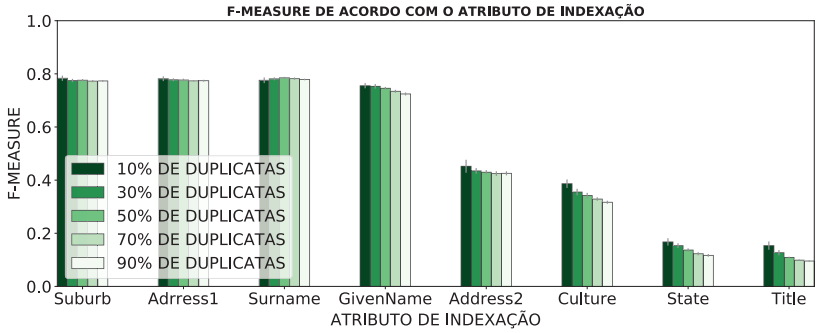
5.3. Avaliação do Atributo nos Algoritmos *Standard Blocking* e *Sorted Neighborhood*

Na RE, os registros podem ser agrupados de acordo com o vizinho mais próximo ou pelo BKV. Nesta seção, a questão **Q3** é abordada: Os resultados da RE são eficazes utilizando o mesmo atributo de indexação nos algoritmos *Standard Blocking* e *Sorted Neighborhood*? Outro objetivo é analisar se os atributos eficazes mantêm a eficácia em conjunto de dados com muitas duplicatas (e.g., 90%). Para tal, dados sintéticos com distintos cenários de duplicidade são utilizados (10% - 90%). A Figura 5(a)-(b) apresenta os resultados da *F-Measure* para cada atributo de indexação nos dois algoritmos, agrupados por atributo (eixo-X). O eixo-Y exibe a *F-Measure*, e as cores das barras representam o conjunto de dados utilizado em cada experimento. Note que o intervalo de confiança é exibido.

Os resultados demostram que: (i) atributos eficazes mantêm a eficácia em ambos algoritmos independente da quantidade de duplicatas, porque os valores da *F-Measure* são similares em todos os *datasets* (e.g., atributo *Culture* em *Standard Blocking* e atributo *Suburb* em *Sorted Neighborhood*); e (ii) atributos ineficazes o são em todos os conjuntos de dados, porque os baixos valores da *F-Measure* são mantidos quando o cenário de dados



(a) F-Measure dos atributos de indexação no algoritmo Standard Blocking



(b) F-Measure dos atributos no algoritmo Sorted Neighborhood

Figura 5. Avaliação dos atributos de indexação nos conjuntos de dados sintéticos.

duplicados muda (e.g., atributo *Address2* em *Standard Blocking* e *Title* em *Sorted Neighborhood*). Além disso, comparando os atributos de indexação entre os algoritmos, os resultados demonstram que alguns atributos de indexação são eficazes em um algoritmo e no outro não. Por exemplo, o atributo *Culture* é eficaz no algoritmo *Standard Blocking* e ineficaz no algoritmo *Sorted Neighborhood*. O atributo *State* é eficaz na abordagem de blocos e ineficaz no método de vizinhança. No entanto, há atributos que possuem resultados significantes em termos da *F-Measure* em ambos algoritmos (e.g., *Given Name*, *Surname* e *Address1*). Por fim, percebe-se que a escolha do atributo depende do método de agrupamento pois um atributo pode ser eficaz em um método e ineficaz em outro.

5.4. Análise da Combinação de Atributos na Etapa de Comparação

Na comparação dos registros, um subconjunto de atributos é selecionado com o intuito de remover aqueles atributos que não contribuem para o processo. Assim, esta seção aborda a questão **Q4**: O conjunto de atributos utilizados na comparação afeta o resultado da RE?

Nesse sentido, a Figura 6 exibe os resultados em termos da *F-meausre* para cada combinação de atributos nos conjuntos de dados *Synthetic*, *Cora* e *Restaurant*. Pode-se pensar que geralmente apenas um atributo não é suficiente para comparar um par de registros. Contudo, os resultados mostram que em alguns domínios de dados um único atributo consegue distinguir as instâncias e classificá-las corretamente. Por exemplo, no conjunto de dados *Cora*, apenas o atributo *Title* tem a maior *F-Measure*. Em *Restaurant*, a segunda maior *F-Measure* é só do *Name*. Mas, existem domínios em que somente um atributo não separa os registros de forma eficaz, como ocorre no conjunto de dados *Synthetic*, onde a melhor *F-Measure* é alcançado com um grupo de quatro atributos.

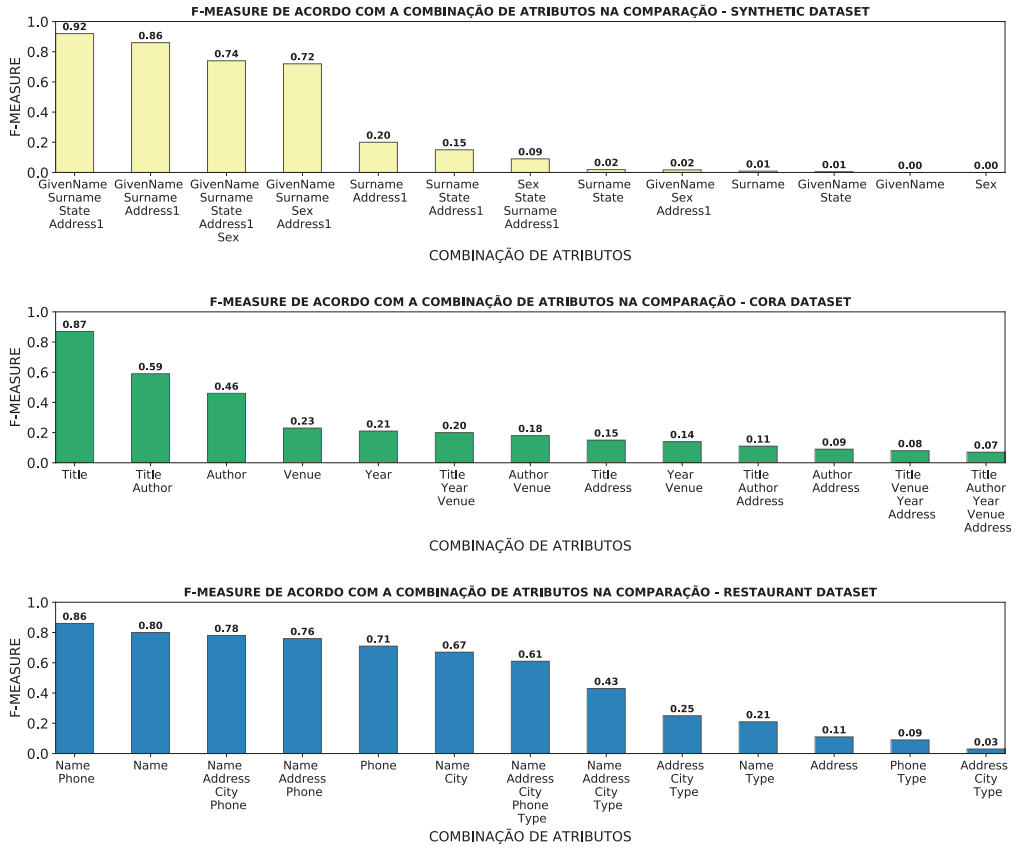


Figura 6. RE considerando diferentes combinações de atributos na comparação.

Outra discussão consiste em pensar que o resultado da RE melhora com uma maior quantidade de atributos na comparação. Porém, geralmente isso não ocorre. Por exemplo, no *Cora*, os atributos *Title*, *Author*, *Year*, *Venue* e *Address* obtêm apenas $0,07$ da *F-Measure*. No *dataset Restaurant*, *Name*, *Address*, *City*, *Phone* alcançam uma *F-Measure* de $0,78$. Quando o atributo *Type* é adicionado ao grupo, o resultado é $0,61$, ou seja, um decréscimo de $0,17$. Aqui, tem-se um exemplo de que a *F-Measure* diminui quando um atributo que não distingue os registros de forma eficaz é utilizado na comparação, pois todas as combinações com o *Type* decresceram. O mesmo ocorre com o *Sex* no *Synthetic*.

No geral, os atributos utilizados na comparação afetam o resultado da RE, porque diferentes combinações produzem diferentes valores da *F-Measure*. Considerando os três *datasets*, a diferença média da *F-Measure* entre o grupo mais eficaz e o grupo menos eficaz é quase $0,85$, ou seja, o impacto da seleção de atributos é significante. Ademais, a quantidade de atributos selecionados está totalmente relacionada com o domínio dos dados, pois o maior valor da *F-Measure* de cada *dataset* é alcançado com grupos de quantidades diferentes. Por exemplo, no *dataset Synthetic*, os atributos *GivenName*, *Surname*, *State* e *Address1* alcançam uma *F-measure* de $0,92$. No *Cora*, apenas *Title* obtém $0,87$.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma análise do impacto da seleção de atributos no processo de RE. Os experimentos avaliaram as etapas da RE e realizaram-se em dados reais e sintéticos. Ao final, o projeto fatorial indica que o atributo utilizado na indexação explica

uma maior variação nos resultados, ou seja, o atributo adotado tem maior influência na RE quando comparado com os métodos de indexação e/ou algoritmos de agrupamento. Na indexação, os métodos *Agnostic* e *Configurations-Based* não possuem diferença significativa, isto é, o atributo tem maior relação com a variação nos resultados do que os métodos de indexação. No agrupamento, alguns atributos são eficazes no algoritmo *Standard Blocking* e ineficazes no *Sorted Neighborhood*. Logo, o critério de seleção de atributos pode mudar conforme algoritmo escolhido. Finalmente, na comparação, o grupo de atributos selecionados afeta a RE, pois diferentes combinações produzem distintos valores da *F-Measure*. Além disso, a quantidade de atributos ideal está relacionada ao domínio dos dados. Em geral, os experimentos destacam a importância do desenvolvimento de novas estratégias que selecionem atributos relevantes para cada uma das etapas da RE. Tais estratégias podem tornar a RE mais eficaz e menos custosa, uma vez que a seleção de atributo é feita automaticamente. No futuro, pretendemos investigar quais métricas estão relacionadas aos melhores atributos do processo de RE em cada uma das etapas.

Referências

Barbosa, L. et al. (2018). Big data integration for product specifications. *Technical Committee on Data Engineering*, 41(2):71–81.

Baxter, R. et al. (2003). A comparison of fast blocking methods for record linkage. In *ACM SIGKDD*, volume 3, pages 25–27, Washington, USA.

Caldeira, L. S. and Ferreira, A. A. (2018). Melhorias no processo de blocagem para resolução de entidades baseadas na relevância dos termos. In *SBBD*, pages 61–72, Rio de Janeiro, Brasil.

Canalle, G. K. et al. (2017). A strategy for selecting relevant attributes for entity resolution in data integration systems. In *ICEIS*, pages 80–88, Porto, Portugal.

Christen, P. (2006). A comparison of personal name matching: Techniques and practical issues. In *ICDM*, pages 290–294, Hong Kong, China.

Christen, P. (2012). A survey of indexing techniques for scalable record linkage and deduplication. *TKDE*, 24(9):1537–1555.

Cohen, W. W. et al. (2003). A comparison of string distance metrics for name-matching tasks. In *WIIW*, pages 73–78, Acapulco, México.

Draisbach, U. and Naumann, F. (2009). A comparison and generalization of blocking and windowing algorithms for duplicate detection. In *QDB*, pages 51–56, Lyon, France.

Jain, R. (1992). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley.

Konda, P. et al. (2019). Executing entity matching end to end: A case study. In *EDBT*, pages 489–500, Lisbon, Portugal.

Papadakis, G. et al. (2015). Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data. *PVLDB*, 9(4):312–323.

Silva, L. S. et al. (2017). Uma avaliação de eficiência e eficácia da combinação de técnicas para deduplicação de dados. In *SBBD*, pages 160–171, Uberlândia, Brasil.

Silva, L. S. et al. (2018). Automatic identification of best attributes for indexing in data deduplication. In *AMW*, Cali, Colombia.

Refinamento Colaborativo de Dados na Web baseado em *Social Coding*

Helton Douglas A. dos Santos[1]*, Marcelo Iury S. Oliveira[2], Bernadette Farias Lóscio[1]

[1]Centro de Informática – Universidade Federal de Pernambuco
Recife, PE – Brasil

[2]Unidade Acadêmica de Serra Talhada – Universidade Federal Rural de Pernambuco
Serra Talhada, PE – Brasil

{hdas,bfl}@cin.ufpe.br, marcelo.iury@ufrpe.br

Abstract. *The Web has emerged as an important platform for sharing information, enabling the publishing and consumption of datasets from different domains. In this contexto, dataset refinement is a primary activity mainly related to data cleansing and enrichment. Usually, refinement is performed by publishers, although consumers often clean and enrich datasets in their consumption activities. However, in general, consumer's effort is lost, since most times the result of the refinement is not shared back with the publisher or other consumers. In this context, this work proposes a refinement strategy based on the principles of social coding to allow the refinement of datasets published on the Web in a collaborative way.*

Resumo. *A Web tem emergido como um importante canal de compartilhamento de informações, habilitando a publicação e o consumo de conjuntos de dados. Neste contexto, o refinamento de conjunto de dados é uma atividade relacionada à limpeza e enriquecimento de dados. Normalmente, o refinamento é realizado pelos publicadores de dados, embora os consumidores também limpem e aprimorem conjuntos de dados em suas atividades de consumo. Porém, o esforço do consumidor é perdido, já que na maioria das vezes o resultado do refinamento não é compartilhado com o publicador ou outros consumidores. Assim, este trabalho propõe uma estratégia baseada nos princípios de social coding para permitir o refinamento de conjuntos de dados publicados na Web de forma colaborativa.*

1. Introdução

Atualmente, o aumento pelo interesse na publicação de dados na Web em diferentes formatos, com licença aberta ou privada, juntamente com o enorme volume de dados gerados pelas redes sociais, tem confirmado o potencial da Web como plataforma de compartilhamento de dados. De maneira geral, as atividades de publicação e consumo de dados na Web são desempenhadas por um conjunto de atores. Um ator pode agir como um publicador ou como um consumidor de dados. O primeiro entrega e produz dados de acordo com condições específicas. O segundo acessa e consome os dados, seja para realizar análises, construir visualizações, ou para gerar novos dados.

[0]Helton Santos e Marcelo Iury agradecem ao CNPq e CAPES pelo apoio financeiro.

Segundo o ciclo de vida dos dados na Web [Lóscio et al. 2015], a limpeza e o enriquecimento dos dados são atividades que fazem parte da fase de refinamento dos dados. Esta fase diz respeito à correção de erros nos conjuntos de dados publicados, como também à atualização e adição de novos dados. Dessa forma, o refinamento de dados na Web pode ser definido como um processo no qual são executados todos os procedimentos relacionados à limpeza e ao enriquecimento de dados.

Geralmente, o refinamento de conjuntos de dados na Web é realizado pelos publicadores de dados antes de efetuarem a publicação dos dados. Porém, consumidores de dados também realizam frequentemente procedimentos de limpeza e de enriquecimento nos conjuntos de dados a fim de melhorar sua qualidade antes da execução de suas atividades de consumo. No entanto, é importante notar que o resultado do refinamento realizado pelos consumidores, na maioria das vezes, não é compartilhado com os publicadores do conjunto de dados original e nem com outros consumidores interessados no mesmo conjunto de dados. Dessa forma, é muito comum que exista retrabalho, tanto por parte dos publicadores como por parte dos consumidores, uma vez que o resultado das atividades de refinamento não são compartilhados.

Em um cenário ideal, provedores e consumidores devem compartilhar seus dados limpos e enriquecidos, de tal forma que ambos poderão dar sua contribuição, sinalizando e corrigindo erros, bem como realizando o enriquecimento dos dados. Nesse contexto, o objetivo deste trabalho é propor uma estratégia baseada nos princípios de *open collaboration* e *social coding* para permitir o refinamento, de forma colaborativa, de conjuntos de dados publicados na Web, contribuindo para reduzir o retrabalho nas atividades de refinamento, bem como para melhorar a qualidade dos conjuntos de dados na Web.

Open collaboration é um sistema de produção que baseia-se em um conjunto distribuído de participantes, fracamente coordenados e orientados a objetivos, que interagem para criar um produto ou serviço [Levine and Prietula 2013]. Por sua vez, *social coding* é uma abordagem fundamentada nos princípios de *open collaboration*, destinada à implementação de projetos de software de maneira compartilhada, por meio de plataformas específicas como o Github[1] [Gousios et al. 2014].

O restante deste artigo está estruturado como se segue. Na Seção 2, são apresentados os trabalhos relacionados. Na Seção 3, é apresentada a solução proposta. Na Seção 4, é discutida a avaliação da solução proposta. Na Seção 5, são apresentadas as conclusões e sugestões para os trabalhos futuros.

2. Trabalhos Relacionados

Neste trabalho, o refinamento de dados pode ser definido como sendo a atividade direcionada a execução de procedimentos de limpeza de dados, como também de enriquecimento, realizando alterações e adições de dados (*e.g.*, dados e metadados), com o objetivo de melhorar a sua qualidade do conjunto de dados como um todo [Lóscio et al. 2015].

Na literatura, é possível encontrar diferentes definições para refinamento, enriquecimento e limpeza de dados. Apesar dessas atividades serem relacionadas, elas não devem ser tratadas da mesma forma. Além disso, entre os trabalhos que apresentam uma definição para esses termos, uma parcela significativa faz uso de definições superficiais,

[1] https://github.com/

não trazendo clareza quanto ao real significado e aplicação de refinamento, limpeza ou enriquecimento.

O trabalho [Rahm and Do 2000] define o conceito de limpeza de dados, como também aborda alguns problemas de qualidade que estão presentes em bases de dados. [Rahm and Do 2000] também descreve as principais fases do processo de limpeza de dados, que vai desde a análise dos dados até a realização da limpeza, fazendo o uso de técnicas e tecnologias usadas em *Data Warehouse*. [Maletic and Marcus 2000] também fornece uma visão geral da literatura sobre limpeza de dados. O principal objetivo desse trabalho é descrever e analisar métodos de detecção automática de erros em conjuntos de dados, como *data mining* e *clustering*. Além disso, o autor apresenta um experimento para investigar diferentes métodos de detecção de erros utilizando um conjunto de dados do mundo real. Por outro lado, [Chapman 2005] escreveu um livro abordando princípios e métodos da limpeza de dados. Ele afirma que limpeza de dados é o processo usado para determinar dados imprecisos ou incompletos, a fim melhorar a qualidade por meio da correção de erros encontrados.

Existem também trabalhos que relacionam o conceito de refinamento de dados com enriquecimento de dados. Contudo, na literatura, a maioria dos trabalhos que traz o enriquecimento de dados diz respeito ao enriquecimento semântico de conjuntos de dados. Por exemplo, [Clarke and Harley 2014] aborda o enriquecimento semântico como um processo direcionado à adição de *tags* semânticas em dados ou metadados a fim de facilitar a compreensão dos dados, como também auxiliar na descoberta. Similarmente, o trabalho proposto por [Fileto et al. 2015] fornece uma visão geral de propostas de enriquecimento semântico para dados em movimento (*i.e.*, dados que descrevem posições espaço temporais de objetos que podem ser capturados por sensores, como GPS), associando aos dados anotações semânticas que descrevem conceitos por meio de ontologias.

Além disso, [dos Santos et al. 2018] realizaram um mapeamento sistemático da literatura na área de publicação e consumo de dados na Web com o intuito de identificar lacunas de pesquisa na nesta área. Como resultado deste mapeamento, foi identificado a ausência de trabalhos que abordem o refinamento de dados na Web. De maneira semelhante, também não foram encontrados estudos relevantes na área de *Social Coding* que tratam ou direcionam aspectos relacionados ao refinamento de dados.

3. Refinamento Colaborativo de Conjuntos de Dados na Web

O refinamento colaborativo tem o objetivo de atribuir também ao consumidor de dados o papel de refinador, podendo solicitar a publicação da nova versão do conjunto de dados após a realização de alguma atividade de refinamento nos dados. O uso de uma estratégia colaborativa pode reduzir também o esforço dos publicadores no refinamento dos conjuntos de dados, visto que esta atividade passa agora a ser realizada também pelos consumidores. Isto também pode levar ao aumento da frequência de atualização dos conjuntos de dados publicados, pois os processos de atualização e correção dos conjuntos de dados, quando realizados por um único publicador, podem tornar-se demorados dependendo do volume de dados.

É importante ressaltar que a construção de uma estratégia para o refinamento colaborativo não é uma atividade trivial. Por exemplo, conflitos podem ocorrer quando dois ou mais consumidores refinam uma mesma versão de um conjunto de dados e tentam fa-

zer sua republicação. Dessa maneira, o versionamento e a resolução de conflitos é uma problemática decorrente do refinamento colaborativo e que serão tratadas neste trabalho.

Nas próximas subseções, detalharemos o funcionamento da estratégia proposta, incluindo todo processo de colaboração envolvendo atores e artefatos. Por fim, apresentaremos a nossa proposta para resolução de conflitos.

3.1. Estratégia Proposta

Este trabalho propõe uma estratégia para o refinamento colaborativo de conjunto de dados na Web, composta por um conjunto de processos e operações que são desempenhadas durante o ciclo de vida de dados da Web [Lóscio et al. 2015, da Silva 2019], envolvendo mais especificamente as etapas de publicação, refinamento, homologação e re-publicação. A estratégia proposta se baseia em princípios básicos de *Open Collaboration* e *Social Coding*.

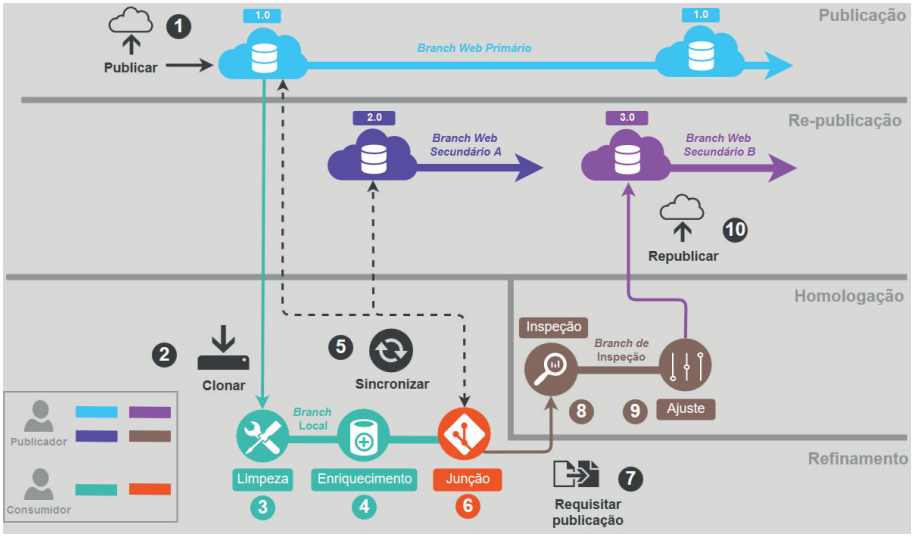


Figura 1. Estratégia para refinamento colaborativo de conjuntos de dados na Web. Fonte: Autores

A Figura 1 apresenta um cenário de utilização da estratégia proposta. Como pode ser visualizado, a estratégia de refinamento colaborativo faz uso de múltiplos *branches* e múltiplas operações.

3.2. Branches

O conceito de *branch* é bastante utilizado em sistemas de controle de versão (*e.g.*, CSV[2] e GitHub[1]). Eles representam cópias lógicas de um ou mais artefatos criadas para que modificações possam ocorrer de forma paralela, *e.g.*, implementar recursos e corrigir *bugs* [Gousios et al. 2014]. Nas plataformas de *Social Coding*, os *branches* são geralmente utilizados para implementar novos componentes de um software, onde instabilidades podem ocorrer no processo de implementação sem afetar os outros usuários.

Na estratégia proposta, a principal finalidade da utilização dos *branches* é limitar e organizar os diferentes estados que um conjunto de dados pode ter durante as etapas de

[2]https://savannah.nongnu.org/projects/cvs/

refinamento, como também gerenciar as diferentes versões geradas, habilitando assim o controle de versão dos conjuntos refinados.

Mais especificamente, são definidos quatro tipos de *branches* diferentes, sendo eles: (i) *Branch Web Primário*, que armazena a primeira versão de um conjunto de dados publicado na Web; (ii) *Branch Local*, que armazena localmente uma cópia de um conjunto de dados publicado na Web (*i.e.*, em computadores pessoais); (iii) o *Branch de Inspeção*, que armazena um conjunto de dados sujeito à inspeção e homologação; e (iv) *Branch Web Secundário*, armazena na Web a nova versão de um conjunto de dados refinado a partir de um *Branch Web Primário*.

Cada *branch* é criado por meio de operações específicas. Por exemplo, o *Branch Web Primário* é criado por meio da operação Publicar. Dessa forma, é possível isolar todas as alterações realizadas sobre um conjunto de dados armazenado no *Branch Local*, não refletindo na versão publicada no *Branch Web Primário*, por exemplo.

3.3. Operações

A estratégia proposta faz uso de um conjunto de atividades e operações que são destinadas a manipulação do conjunto de dados, seguindo um fluxo ordenado como apresentado na Figura 1. Cada atividade e operação possuem papéis e funções específicas que podem ser desempenhados tanto por publicadores como por consumidores.

Operação Publicar: A operação Publicar pode ser vista como uma função *publicar* que, a partir de um conjunto de dados *DW1*, cria e retorna um novo *Branch Web Primário* B_{web} armazenando o conjunto de dados *DW1*:

$$B_{web} = publicar(DW1)$$

Ao realizar a operação Publicar, é criado um novo *Branch Web* que armazena o conjunto de dados em sua versão inicial (versão 1.0), tornando-se disponível na Web para o acesso e o consumo. Para que um conjunto de dados possa passar pelos processos de limpeza e enriquecimento é necessário primeiro a execução da operação Clonar. Esta operação tem o objetivo de gerar uma cópia de um conjunto de dados publicado no *Branch Web* e armazená-lo em um *Branch Local* que é criado pela operação, permitindo assim conservação dos dados originais. A operação Clonar pode ser definida como:

Operação Clonar. A operação Clonar pode ser vista como uma função *clonar* que, a partir de um *Branch Web* B_{web}, cria e retorna um novo *Branch Local* B_{local} armazenando uma cópia *DL1* do conjunto de dados *DW1*:

$$B_{local} = clonar(B_{web})$$

Os processos de limpeza e enriquecimento são compostos por procedimentos que tem o objetivo de corrigir erros dentro do conjunto (*e.g.*, correção de ortografia) ou de enriquecer o conjunto (*e.g.*, adição de dados). As operações e os procedimentos de limpeza e enriquecimento podem ser desempenhados sobre cópias de conjunto de dados criadas e armazenadas no *Branch Local* pela operação Clonar. Cada procedimento executado sobre um conjunto de dados gera uma entrada em um *log* de refinamento de conjunto de dados

Com o uso do *log* de refinamento, é possível localizar uma alteração realizada no conjunto de dados, como também identificar valores inseridos e substituídos. Um *log* de

refinamento de um conjunto de dados *C* pode ser descrito como um conjunto $C.L_{ref} = \{Lproc_1, Lproc_2, ..., Lproc_n\}$, no qual cada $Lproc_i$ é uma entrada no *log* e descreve um procedimento de refinamento que foi executado sobre o conjunto de dados *C*.

De modo geral, um $Lproc_i$ pode ser definido como {*t, id, a, v, va*}. Em particular, *t* representa o *timestamp* da realização do procedimento, *id* representa o identificador do registro, *a* representa o atributo do conjunto de dados modificado, *v* representa o novo valor inserido e *va* representa valor anterior substituído, quando houver. Exceções a esse formato são os procedimentos de enriquecimento resultam em entradas diferentes, que registram informações a respeito a estrutura do conjunto de dados ou de seus metadados. O *log* de refinamento pode ser implementado através de um arquivo tabular ou mesmo um arquivo JSON.

A execução dos procedimentos pode ser auxiliada por ferramentas específicas de refinamento de dados, que por sua vez podem gerar os *logs* de forma automática. O *log* de refinamento é essencial para as operações Sincronizar e para a atividade de Junção.

Operação Sincronizar. A operação Sincronizar pode ser vista como uma função *sincronizar* que, a partir do *log* de refinamento *logDL2* de uma nova versão refinada *DL2* do conjunto de dados *DL1*, armazenado no *Branch Local* B_{local}, verifica se existem novas versões publicadas na Web, analisando os *logs* de refinamento de ambos, *logDL2* e *logDW2*, e retornando um conjunto de procedimentos não conflitantes $P = \{p_1, p_2, ..., p_n\}$ e um conjunto de conflitos $C = \{c_1, c_2, ..., c_n\}$. Dois procedimentos p_i e p_j são considerados conflitantes quando atuam em um mesmo conteúdo do conjunto de dados, por exemplo um mesmo registro e atributo, e em versões diferentes de um mesmo conjunto de dados. Cada par de procedimentos conflitantes $< p_i, p_j >$ corresponde a um conflito *c*:

$$P, C = sincronizar(DL2)$$

A operação Sincronizar recupera o *log* de refinamento da versão mais recente de um conjunto de dados publicado na Web e o compara com o *log* de refinamento do conjunto de dados do *Branch Local*. A sincronização dos *logs* de refinamento é realizada por meio do confronto dos procedimentos realizados no conjunto de dados local com o conjunto publicado na Web verificando a existência de conflitos. Como já mencionado, o *log* de refinamento é capaz de fornecer informações precisas sobre as alterações realizadas no conjunto de dados por meio de procedimentos de limpeza e enriquecimento desempenhados, como também local da alteração e o valor inserido ou removido.

A atividade de junção compreende a resolução dos conflitos resultantes da operação Sincronizar. Como parte da atividade de junção, são executados os procedimentos de refinamento sobre o conjunto de dados local. Após a execução dos procedimentos e realização da atualização, uma nova versão do conjunto de dados local (*DL3*) é gerada juntamente com o novo *log* de refinamento *logDL3*. Para que um conjunto de dados possa ser inspecionado e homologado, é necessário que o consumidor requisite a republicação do conjunto refinado através da operação Requisitar Publicação.

Operação Requisitar Publicação. A operação Requisitar Publicação pode ser vista como uma função *requisitar* que, a partir de um conjunto de dados refinado e atualizado *DL3*, o *log* do refinamento contendo os procedimentos e alterações realizadas sobre

ele *logDL3* e o ator do refinamento $A_{consumidor}$, cria e retorna um *Branch de Inspeção* destinado à inspeção e homologação do conjunto de dados $B_{inspecao}$:

$$B_{inspecao} = requisitar(DL3, logDL3, A_{consumidor})$$

Ao executar esta operação, um novo *Branch de Inspeção* é criado, e nele é armazenado o conjunto de dados refinado e atualizado pela atividade de Junção (*DL3*), juntamente com o seu *log* de refinamento *logDL3*. Por fim, a criação do *Branch de Inspeção* é necessária para que a atividade de inspeção e ajuste seja desempenhado de forma isolada.

O processo de inspeção tem o objetivo de analisar o conjunto de dados e verificar a consistência dos dados e metadados, definindo o destino do conjunto de dados, seja para aceitação ou rejeição. Após a verificação, o conjunto de dados pode passar por alguns ajustes por parte do publicador. Os ajustes podem ser necessários como complemento do refinamento e tratam de mudanças adicionais aplicadas sobre o conjunto de dados inspecionado. Por exemplo, o publicador ao inspecionar o conjunto de dados pode se deparar com algum problema onde ele mesmo pode realizar o ajuste necessário, não necessitando da interferência do consumidor. O ajuste também pode ser motivado por uma comunicação extra entre o consumidor e o publicador buscando um melhor entendimento do refinamento realizado, podendo conduzir a ajustes adicionais. Por fim, após a inspeção e possíveis ajustes, o conjunto de dados é homologado e pronto para a republicação na Web

Operação Operação Re-publicar. A operação Re-publicar pode ser vista como uma função *republicar* que, a partir de um conjunto de dados homologado *DL3*, um *log* de refinamento *logDL3* contendo os procedimentos e alterações realizadas sobre ele, o ator do refinamento $A_{consumidor}$ e o ator da republicação $A_{publicador}$ cria e retorna um novo *Branch Web Secundário* B_{websec} armazenando uma nova versão do conjunto de dados *DW3*:

$$B_{websec} = republicar(DL3, logDL3, A_{consumidor}, A_{publicador})$$

Esta operação cria um novo *Branch Web Secundário* e armazena o conjunto de dados homologado *DL3* juntamente com o seu *log* de refinamento *logDL3*. Nesta operação, deve ser informado o consumidor que realizou o refinamento do conjunto, como também o publicador que está realizando a republicação. Após a republicação, o conjuntos de dados republicado torna-se disponível na Web para o acesso, podendo ele ser acessado por outros consumidores, como também passar por um novo refinamento se necessário. Dessa forma, um novo ciclo de vida pode ser iniciado, como também uma nova interação da estratégia de refinamento, contribuindo assim para o aumento da qualidade dos conjuntos de dados publicados na Web.

3.4. Resolução de Conflitos Refinamento

A estratégia de refinamento colaborativo define que consumidores que modifiquem as cópias privadas dos conjuntos de dados, *i.e.*, cada consumidor modifica apenas seu *Branch Local* de um conjunto de dados. Esse isolamento permite que dois ou mais consumidores trabalhem em paralelo. No entanto, conflitos podem emergir devido ao trabalho concorrente, e se tornam mais complexos à medida que as mudanças crescem sem serem integradas e à medida que novas operações de refinamento são realizadas no conjunto de dados.

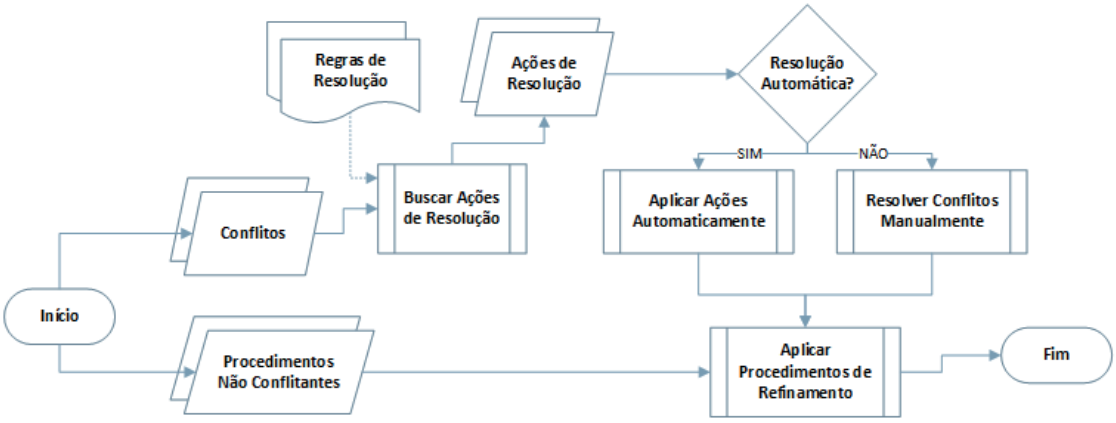


Figura 2. Fluxograma - Junção de conjuntos de dados. Fonte:Autor

A fim de solucionar os conflitos resultantes do refinamento colaborativo de um mesmo conjunto de dados, foi proposta a abordagem para resolução de conflitos apresentada na Figura 2 e descrita a seguir. A solução proposta faz uso de heurísticas para facilitar a resolução de conflitos. Estas regras definem heurísticas para resolução de conflitos mais frequentes. Por exemplo, dado um conflito $c1 =< p_1, p_2 >$, tal que $p1$ é um *procAddCampoVazio* realizado no conjunto de dados local que tem como entrada $timestamp = 1528111295$, $i = 1$, $a =$ "$area$", $v =$ "100" e $p2$ é um *procAddCampoVazio* realizado na versão mais recente do conjunto de dados publicada na Web que tem como entrada $timestamp = 1528810818$, $i = 1$, $a =$ "$area$", $v =$ "200", observa-se que os procedimentos foram executados em um mesmo registro e em um mesmo atributo. Uma regra de resolução de conflito pode analisar os procedimentos da seguinte forma: Se os procedimentos $p1$ e $p2$ são *procAddCampoVazio*, então selecione o procedimento mais recente. Dessa forma, a ação de resolução descarta o procedimento mais antigo, permanecendo o mais recente.

Cada ação de resolução pode resolver um ou mais conflitos de forma automática, sem interferência do usuário. Por outro lado, nem todos os conflitos podem ser resolvidos por meio das regras de resolução. Para os casos restantes, a atuação do consumidor é necessária para resolver o conflito. Alguns dos conflitos são facilmente resolvidos. Esta resolução manual pode inclusive ser assistida por uma ferramenta gráfica que forneça funcionalidades para comparação dos elementos dos elementos conflitantes e permita ao usuário selecionar qual alteração realizada é a correta ou mais adequada. Ocasionalmente, porém, o conflito é tão difícil de resolver que o usuário não pode fazê-lo sozinho de tal maneira que as mudanças feitas em ambas as versões mescladas sejam levadas em consideração.

4. Avaliação

Para a avaliação da estratégia de refinamento proposta neste trabalho, optou-se pela execução de um experimento no qual um conjunto de participantes (estudantes de pós-graduação) realizaram o processo de refinamento em três cenários distintos.

No primeiro cenário, o refinamento foi realizado apenas por um único participante, atuando no papel de publicador, não sendo assim usada a estratégia de refinamento colaborativo. Em especial, o participante com o papel de publicador possui experiência ativa

na publicação de dados na Web, como também o conhecimento de todo o processo convencional de publicação. No segundo cenário, o refinamento foi realizado por dois participantes, um consumidor e um publicador, sendo aplicada a estratégia colaborativa. Após realizar as operações de limpeza e enriquecimento, o consumidor requisitou a publicação do refinamento realizado, anexando o conjunto de dados refinado e o *log* de refinamento. Por sua vez, o publicador verificou que há uma nova requisição de republicação e inspecionou o conjunto de dados com o auxílio do *log* de refinamento anexado. Por fim, o publicador validou o conjunto de dados e realizou a republicação da nova versão. O terceiro cenário se diferencia do segundo no que tange ao refinamento ter sido realizado por dois consumidores e um publicador. Porém, um dos consumidores ao requisitar a publicação do conjunto refinado, realizou a sincronização do *log* de refinamento a fim de atualizar a sua versão do conjunto por meio do processo de Junção.

Para viabilizar a realização do refinamento, foi desenvolvido um conjunto de serviços para Refinamento Colaborativo[3] que possibilitam gerenciar as operações Requisitar Publicação, Sincronizar e Republicar, como também o processo de Inspeção. Esses serviços foram incorporados à solução DWMS proposta por [Oliveira et al. 2018]. O DWMS é uma coleção de serviços, organizados em módulos, que permitem aos usuários compartilhar conjuntos de dados na Web.

Também foi desenvolvido um conjunto de dados com base em dados[4] da Empresa de Manutenção e Limpeza Urbana (Emlurb). Em particular, o conjunto de dados usado na avaliação continha 100 registros. Nos quais, foram introduzidos 20 erros, tais como erros referenciais, valores falsos e valores abreviados.

Para execução do experimento, foram selecionados 4 participantes, sendo que um dos participantes assumiu o perfil de publicador e os outros três de consumidor. Em especial, o participante com o perfil de publicador possui experiência na publicação de dados na Web. Dessa forma, foi possível garantir que de fato o participante com perfil de publicador tem o conhecimento de todo o processo convencional de publicação de dados na Web.

Todo o processo de avaliação foi realizado em uma sala de reunião fechada. Primeiramente, foi exposta a estratégia para o refinamento colaborativo com intuito de guiar os participantes no desempenho de todo o processo. Foram explicados os procedimentos de refinamento que poderiam ser utilizados, tanto os procedimentos de limpeza, quanto os de enriquecimento. Em seguida, foi apresentado o *log* de refinamento que registra cada procedimento de refinamento realizado.

O DWMS foi instanciado e executado em um servidor local e cada participante teve acesso por meio de um navegador Web convencional. Em cada cenário, os participantes tiveram 20 minutos para realizar as atividades. É importante destacar que neste experimento foi utilizado o conjunto de dados apenas no formato CSV.

4.1. Hipóteses e Critérios de Avaliação

A seguir são apresentadas as hipóteses avaliadas pelo experimento.

[3]Detalhes de implementação e arquitetura desse serviço estão disponível em https://blind-review.org/

[4]http://dados.recife.pe.gov.br/dataset/central-de-atendimento-de-servicos-da-emlurb-156/resource/8d4b73c8-d1e1-4efc-97a3-086a682b93b2

- **H1**: A estratégia para o refinamento colaborativo de dados na Web reduz o esforço do publicador no refinamento de conjuntos de dados.
- **H2**: Quando consumidores acessam uma nova versão de um conjunto de dados que foi refinado anteriormente por outro consumidor, há reaproveitamento de trabalho.
- **H3**: Quando um conjunto de dados é refinado pelo consumidor por meio da estratégia colaborativa, há aumento na qualidade do conjunto de dados.

Para avaliar a eficiência da estratégia, foi realizada uma pesquisa qualitativa com cada participante. Ao final de todo o experimento pediu-se que os participantes respondessem um questionário composto de afirmações relacionadas à estratégia proposta. Assim, a partir das notas atribuídas, obteve-se as métricas destinadas à validação das hipóteses H1 e H2. Por meio das métricas, calculamos a média de cada item do questionário de avaliação, i.e., somamos as notas e dividimos pelo número de participantes.

Para avaliar a qualidade do conjunto de dados, utilizou-se os critérios de completude e corretude [Wang and Strong 1996]. Para cada cenário, foi avaliado a qualidade do conjunto de dados da versão inicial e da versão refinada. Assim, ao final de cada cenário, identificou-se quais procedimentos de refinamento foram realizados com base no *log* de refinamento. Verificou-se também a quantidade de procedimentos de limpeza e de enriquecimento que foram realizados e contabilizamos quantos erros foram corrigidos com os procedimentos realizados ao final de cada cenário, assim como o que foi enriquecido. Dessa forma, foi possível mensurar a corretude e completude do conjunto de dados antes e depois de cada cenário e assim direcionar à validação da hipótese H3.

4.2. Resultados

De forma geral, os participantes indicaram resultados positivos em relação à diminuição do esforço relacionado às atividades de refinamento proporcionada pela estratégia. Em particular, o participante que atuou como publicador afirmou que mesmo com as atividades adicionais de inspeção e republicação, os consumidores já haviam corrigido a maior parte dos erros, diminuindo assim seu esforço.

Os participantes apontaram que a estratégia colaborativa também permitiu que o trabalho gasto no refinamento de um conjunto de dados possa ser reaproveitado por meio da republicação deste conjunto. Os mesmos reconheceram o benefícios que essa colaboração pode trazer a novos consumidores, inclusive se sentiram mais motivados a contribuir. Contudo, os participantes também afirmaram que a utilização da estratégia colaborativa não evita necessariamente a realização de novas atividades de refinamento. Ainda existem casos que consumidores podem ou devem incrementar o refinamento já realizado anteriormente, seja com o enriquecimento ou com ajustes de limpeza que julgaram necessários.

Seguindo com os resultados, foram avaliados também a qualidade do conjunto de dados refinado em cada cenário (Figura 3). Por meio do *log* de refinamento gerado por cada participante, foram identificados quais erros contidos na versão inicial do conjunto de dados foram corrigidos a fim de avaliar a corretude do conjunto de dados obtido ao final do refinamento.

Como apresentado na Figura 3, no Cenário 1, o participante que desenvolveu o papel de publicador ao realizar o refinamento do conjunto de dados realizou somente 4

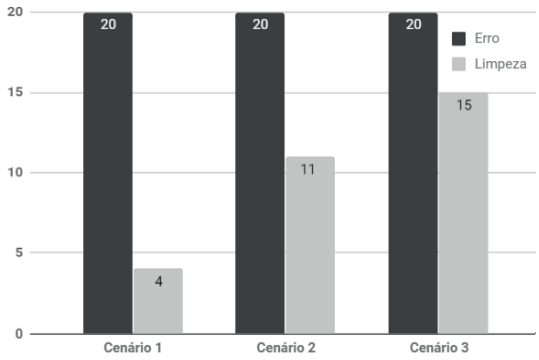


Figura 3. Distribuição de número de erros inseridos e erros corrigidos por cenário. Fonte: Autor

procedimentos de limpeza, corrigindo, assim, apenas 20% dos erros contidos. Por outro lado, no Cenário 2, foram realizados, no total, 11 procedimentos de limpeza que corrigiram 11 dos 20 erros contidos no conjunto de dados, totalizando 55% de erros corrigidos. Comparando os Cenários 1 e 2, pode-se observar que houve um aumento expressivo no número de procedimentos de limpeza realizados.

No Cenário 3, foram realizados 15 procedimentos de limpeza sobre o conjunto de dados, corrigindo 15 dos 20 erros inseridos, *i.e*, 75% dos erros. O elevado indicador de correção se deu pelo fato de 2 consumidores terem realizado colaborativamente o refinamento do conjunto de dados. Ao refinar o conjunto de dados colaborativamente, os participantes somaram esforços. Um dos participantes realizou 8 procedimentos de limpeza, enquanto o outro realizou mais 7. É importante destacar que não houve conflito nos procedimentos realizados, pois foram realizados para a correção de diferentes erros, que, por sua vez, estavam inseridos em diferentes locais no conjunto de dados.

Por outro lado, devido a não realização de procedimentos de enriquecimento sobre o conjunto de dados pelos participantes do experimento, não foi possível avaliar a completude do conjunto de dados refinado ao final de cada cenário. Porém, os indicadores mostram que a utilização da estratégia para o refinamento colaborativo aumentou consideravelmente a qualidade dos conjuntos de dados, como ilustrado nos Cenários 2 e 3.

5. Conclusão e Trabalhos Futuros

Este trabalho propõe uma nova abordagem para a realização de refinamento de dados na Web a partir do trabalho colaborativo de consumidores e publicadores de dados. A estratégia proposta faz uso de um conjunto de operações e *branches* para criação, homologação e manutenção de versões refinadas de conjuntos de dados publicado na Web.

Realizando a análise dos resultados obtidos concluiu-se que a estratégia para o refinamento colaborativo de conjuntos de dados na Web se mostrou eficiente nos cenários em que ela foi aplicada (Cenário 2 e 3), e em todo o experimento realizado as hipóteses elencadas foram validadas. Dessa forma, de acordo com os experimentos realizados, o uso da estratégia de refinamento colaborativa contribui para reduzir o retrabalho, o esforço do publicador, como também favorece o aumento da qualidade dos dados publicados na

Web.

Como trabalhos futuros, destacam-se o problema da resolução de conflitos que ocorre no processo de junção dos conjuntos de dados e a definição de heurísticas de resoluções de conflitos de forma que esta operação seja realizada automaticamente. Além disso, está prevista a realização de novos experimentos com mais participantes, com uma maior variedade de conjuntos de dados a serem refinados, e com diferentes faixas de duração, a fim de comparar o refinamento realizado em ambos. A partir dos novos experimentos, serão coletadas informações úteis para a realização de melhorias na estratégia proposta.

Referências

Chapman, A. D. (2005). *Principles of data quality*. GBIF.

Clarke, M. and Harley, P. (2014). How smart is your content? using semantic enrichment to improve your user experience and your bottom line. *Science*, 37(2):41.

da Silva, K. M. (2019). Um modelo de ciclo de vida dos dados na web. Master's thesis, Universidade Federal de Pernambuco, Centro de Informática, Curso de Pós-Graduação em Ciências da Computação, Recife.

dos Santos, H. D. A., Oliveira, M. I. S., Glória de Fátima, A., da Silva, K. M., Muniz, R. I. V. C. S., and Lóscio, B. F. (2018). Investigations into data published and consumed on the web: a systematic mapping study. *Journal of the Brazilian Computer Society*, 24(1):14.

Fileto, R., Bogorny, V., May, C., and Klein, D. (2015). Semantic enrichment and analysis of movement data: probably it is just starting! *SIGSPATIAL Special*, 7(1):11–18.

Gousios, G., Pinzger, M., and Deursen, A. v. (2014). An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering*, pages 345–355. ACM.

Levine, S. S. and Prietula, M. J. (2013). Open collaboration for innovation: Principles and performance. *Organization Science*, 25(5):1414–1433.

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.

Maletic, J. I. and Marcus, A. (2000). Data cleansing: Beyond integrity analysis. In *Iq*, pages 200–209.

Oliveira, L. E. R., Oliveira, M. I. S., Santos, W. C. d. R., and Lóscio, B. F. (2018). Data on the web management system: a reference model. In *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age*, page 2. ACM.

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

Wang, R. Y. and Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33.

Evolution-based Refinement of Cross-language Ontology Alignments

Juliana Medeiros Destro, Julio Cesar dos Reis, Ricardo da S. Torres[1], Ivan Ricarte[2]

[1]Institute of Computing – University of Campinas (UNICAMP) – Brazil

[2]School of Technology – University of Campinas (UNICAMP) – Brazil

{juliana.destro,jreis,rtorres}@ic.unicamp.br, ricarte@unicamp.br

Abstract. *Ontology alignment plays a key role for information interconnectivity between computational systems relying on ontologies described in different natural languages. Existing approaches for ontology matching usually provide equivalent type of relation in the generated mappings. In this article, we propose a refinement technique to enable the update of the semantic type of the mapping such as "is-a", "part-of", etc. Our approach relies on information from the ontology evolution to apply refinement actions. We formalize the refinement actions and procedures, as well as apply the proposal in application scenarios.*

1. Introduction

Ontologies are usually created by different authors, using different vocabularies and, possibly, in different natural languages. The number of ontologies created in different languages grows as their use increases [Trojahn et al. 2014]. The process of creating correspondences, or mappings between concepts, is called ontology matching. Ontology mappings are crucial for enabling interconnectivity in multiple systems.

Ontology mapping refinement expands the semantic relations identified during the matching process. We differentiate *relation* from *relationship*, where the former represents a mapping, and the latter represents concept connections in an ontology. Refinement can modify or enrich semantic relations. For instance, during the refinement process, an equivalence ($\equiv$) relation (*i.e.*, a relation defining that two interrelated concepts are equivalent) can be modified to an *is-a* ($\sqsubseteq$) (*i.e.*, representing a relation in which one concept is a specialization of the other) [Arnold and Rahm 2014]. Mapping refinement poses a challenge due to the difficulties in establishing semantic relations between concepts, beyond the relatedness identified by the matching procedures.

Ontology mappings with enriched semantic correspondences might boost ontology merging [Raunich and Rahm 2011]. However, most of the matching approaches are approximative and infer mappings based on thresholds of relatedness between concepts. In this sense, finding only exact matches or at most *is-a* relations between pairs of concepts affects not only monolingual ontology matching, but also cross-lingual matching, when the source and target ontologies are labeled in different natural languages.

When a knowledge domain expands, ontologies representing the domain need to be updated to reflect domain changes. Consequently, ontologies are constantly evolving, by adding and removing concepts and relationships over time. These changes indicate how concepts and their relationships with each other evolved. In this investigation, we

argue that ontology changes can be a valuable source of information to enhance the correspondences found between concepts beyond equivalences based on the type of semantic relation.

Our goal is to investigate the use of ontology evolution information, such as change operations (*e.g.*, changing the value of an attribute, and removal of a concept), to help in the process of mapping refinement. We believe that the use of this information might provide an understanding of how the concepts were updated over time to support the decision and application of actions required to modify the type of semantic relation in mappings.

Our conducted investigation assesses change operations and their correlation with semantic relations in mappings. In particular, we analyze how the evolution of concepts impacts on their relationships with neighbor concepts on the same ontology via illustrative cases. In addition, we analyze how that fact can be useful to enrich the semantics of correspondences. Overall, our study reveals a promising approach on the use of ontology evolution changes to enhance semantic relations in mappings.

The remainder of this article is organized as follows: Section 2 presents the foundations and related work. Afterwards, in Section 3 we present a set of formal definitions including the research problem. Section 4 presents our proposal for refinement of cross-lingual mappings. Section 5 introduces an application scenario to exemplify our proposal and discusses lessons learned. Finally, Section 6 wraps up the article and points out future research.

2. Related Work

The effects of ontology evolution in mappings have been already demonstrated in the literature [Dos Reis et al. 2014]. Ontology changes may impact established mappings and, for instance, cause modifications in semantic relations between interrelated concepts. Several examples of ontology mapping changes where presented by [Gross et al. 2012]. A detailed descriptive analysis of the impact of ontology changes on mappings were presented in the work [Dos Reis et al. 2014]. They showed the correlation between ontology changes and mapping evolution. The method proposed by [Dinh et al. 2014] aims at identifing the most relevant concept's attributes for supporting mapping adaptation when ontologies evolve, using differences identified among current and past versions of the ontologies.

Several approaches were developed to tackle the matching problem. For instance, [Trojahn et al. 2014] presented an extensive survey on matching systems and techniques for accomplishing multilingual and cross-lingual ontology matching. Ontology matching techniques have considered the use of similarity methods relying on background knowledge. Similarity measures aim to calculate the degree of relatedness between concepts exploiting different knowledge sources (*e.g.*, ontologies, thesauri, and domain corpora).

Align ontologies by doing matching with external knowledge sources was proposed in the work [Aleksovski et al. 2006]. They explored paths between the anchored matched concepts to find mapping between concepts. Differently, [Sabou et al. 2008] proposed to align ontology concepts by selecting the most appropriate ontology over multiple and heterogeneous knowledge sources. The *TaxoMap* approach has used the *WordNet* lexical database as background knowledge [Hamdi et al. 2010].

Domain-specific resources have only been superficially studied in the biomedical domain for the purpose of ontology alignment. For example, [Zhang and Bodenreider 2007] proposed exploring the *Unified Medical Language System* (UMLS)[1] structure to accomplish matching between anatomical ontologies. Their results indicated that domain knowledge is a key factor in the identification of additional mappings compared with the generic matching approach.

Ontology mapping refinement helps to expand the types of semantic relations identified during the matching process. Some techniques use external resources aiming to improve and increase the number and precision of established mappings. The work of [Arnold and Rahm 2014] defined a mapping refinement technique by using a set of equivalent mappings as input. They explored generic external resources and proposed a two-step enrichment technique to improve existing imprecise mappings. They used linguistic techniques and resources like *WordNet* to refine semantic relations between aligned concepts. Their work aimed to transform equivalence between concepts into an *is-a* or *part-of* relation, which may further reflect the real semantics of mapped concepts. The use of external resources influences the results and needs further research to determine their impact. The work of [Stoutenburg 2008] argues that the use of upper ontologies (an ontology which consists of very general terms that are common across all domains) and linguistic resources might enhance the alignment process.

The *TaxoMap* matching tool [Hamdi et al. 2010] explored pattern-based refinement techniques, applying manually created patterns to other mappings in the same domain. In contrast, [Spiliopoulos et al. 2008] presented the *Classification-Based Learning of Subsumption Relations* method for ontology alignment. This automated method relies on the exploration of patterns that describe the relation between concepts (*e.g.*, siblings at the same hierarchy level or attributes with same content). These patterns are identified by applying a classification task using machine learning methods.

Our proposed approach in this investigation is a novel technique for mapping refinement. We combine the information obtained from ontology change operations with semantic similarity measures based on defined refinement actions applicable to mappings.

3. Definitions and Problem Characterization

3.1. Definitions

We use the following notations and definitions throughout this paper.

Ontology. An ontology O specifies a conceptualization of a domain in terms of concepts, attributes and relationships [Gruber 1995]. Formally, an ontology $O = (C_O, R_O, A_O)$ consists of a set of concepts C_O or $Concepts_O$ interrelated by directed relationships R_O. For each concept $c_k \in C_O$, $L(c_k)$ defines the value of the preferred label for c_k expressing its name denoted by a natural language string. For example, *"cardio vascular diseases"* describes the label of a concept. The labels can be defined by properties in RDF schema like *rdfs:label*, and SKOS (*Simple Knowledge Organization System*) like *skos:prefLabel*. Concept $c_k \in C_0$ is associated with a set of attributes $A_O(c) = \{a_1, a_2, ..., a_p\}$. Each relationship $relation(c_1, c_2) \in R_O$ is typically a triple (c_1, c_2, r), where r is the relationship (*e.g.*, "is_a", "part_of", and "advised_by")

[1] `www.nlm.nih.gov/research/umls` (As of May 2018).

inter-relating c_1 and c_2. Neighborhood of a concept consist of the set of concepts with a relation to c_s, defined as $neighborhood(c_s) = c_s, sup(C_s), sub(C_s)$, where $sup(C_s)$ is the set of super concepts of c_s and $sub(C_s)$ is the set of sub concepts of c_s.

Similarity between concepts. Given two particular concepts c_1 and c_2, the similarity between them can be defined as the maximum similarity between each couple of attributes from c_1 and c_2. Formally:

$$sim(c_1, c_2) = \arg\max\ sim(a_{1x}, a_{2y}) \tag{1}$$

where $sim(a_{1x}, a_{2y})$ is the similarity between two attributes a_{1x} and a_{2y} denoting concepts c_1 and c_2 respectively. We can compute this similarity at different linguistic levels: *character*, *string*, and *semantic* level [Dinh et al. 2014].

Similarity measures. Similarity function used to calculate similarity between concepts. Formally:

$$f(c_1, c_2) = sim(c_1, c_2) \tag{2}$$

where $f(c_1, c_2)$ is the similarity function and $sim(c_1, c_2)$ denotes the calculated similarity between concepts c_1 and c_2. We use similarity measures at *character* and *semantic* linguistic levels.

Mapping. Given two concepts c_1 and c_2 from two different ontologies, a mapping m_{12} can be defined as:

$$m_{12} = (c_1, c_2, semType, conf) \tag{3}$$

where $semType$ is the semantic relation connecting c_1 and c_2.

The following types of semantic relation are considered: *unmappable* [$\perp$], *equivalent* [$\equiv$], *narrow-to-broad* [$\leq$], *broad-to-narrow* [$\geq$] and *overlapped* [$\approx$]. For example, concepts can be equivalent (*e.g.*, "cabeça"$\equiv$"head"), ("cabeça" in Portuguese language) one concept can be less or more general than the other (*e.g.*, "thumb"$\leq$"dedo")("dedo" in Portuguese language) or concepts can be somehow semantically related ($\approx$). The $conf$ is the similarity between c_1 and c_2 indicating the confidence of their relation [Euzenat and Shvaiko 2013]. $\mathcal{L}_{XY} = \{(m_{12})_k | k \in \mathbb{N}\}$ consists of the set of mappings between two ontologies O_X and O_Y as the result of an alignment process. Cross-lingual mapping is established between O_X and O_Y with concepts denoted by different natural languages, where $L(c_1)$ is expressed in language α, and $L(c_2)$ is expressed in language β such that $\alpha \neq \beta$. In a cross-lingual mapping, O_X and O_Y have concepts denoted by different natural language.

Ontology change operations. An ontology change operation (OCO) is defined to represent a change in an attribute, in a set of one or more concepts or in a relationship between concepts. OCO is classified into two main categories: *atomic* and *complex* changes (*cf.* Table 1). Each OCO in the atomic category cannot be split into smaller operations, whereas each one in the complex category is composed of more than one

Table 1. Ontology change operations (OCOs) [Hartung et al. 2013].

Change operation		Description
A	$addC(c)$	Addition of a new concept $c \in O_X^j$
t	$delC(c)$	Deletion of an existing concept $c \in O_X^{j-1}$
o	$addA(a, c)$	Addition of a new attribute a to a concept $c \in O_X^{j-1}$
m	$delA(a, c)$	Deletion of an attribute a from a concept $c \in O_X^{j-1}$
i	$addR(r, c_1, c_2)$	Addition of a new relationship r between two concepts c_1 and c_2 which belongs to O_X^{j-1}
c	$delR(r, c_1, c_2)$	Deletion of an existing relationship r between two concepts c_1 and c_2 which belongs to O_X^{j-1}
	$chgA(c, a, v)$	Change of attribute a in concept c with the new value v
	$moveC(c, p_1, p_2)$	Moving of concept c (and its subtree) from concept p_1 to concept p_2
C	$substitute(c_i, c_j)$	Replacement of concept $c_i \in O_X^{j-1}$ by concept $c_j \in O_X^j$
o	$merge(C_k, c_j)$	Fusion of a set of multiple concepts $C_k \subset O_X^{j-1}$ into concept $c_j \in O_X^j$
m	$split(c_i, C_r)$	Split of concept $c_i \in O_X^{j-1}$ into a set of resulting concepts $C_r \subset O_X^j$
p	$toObsolete(c)$	Sets status of concept c to *obsolete* (c is no longer available)
l	$delInnerC(c_i, p_j)$	Deletion of concept c_i where $p_j \in sup(c_i)$ and $sub(c_i) \neq \emptyset$ from ontology O_X^{j-1}
e	$delLeafC(c_i, p_j)$	Deletion of leaf concept c_i where $p_j \in sup(c_i)$ and $sub(c_i) = \emptyset$ from ontology O_X^{j-1}
x	$addInnerC(c_i, p_j)$	Addition of a sub concept c_i under the concept $p_j \in sup(c_i)$ to the ontology O_X^j
	$addLeafC(c_i, p_j)$	Addition of leaf concept c_i where $p_j \in sup(c_i)$ and $sub(c_i) = \emptyset$ to the ontology O_X^j
	$revokeObsolete(c)$	Revokes obsolete status of concept c (*i.e.*, c becomes active)

atomic operation. For instance, the operation $chgA(c, a, v)$ is composed of two atomic operations, $delA(a, c)$ and $addA(a, v)$. We denote successive ontology versions derived from evolution by O^{j-1} and O^j to identify ontologies created in time $j-1$ and j. Changes may occur from one version to another, and we consider existing tools to automatically detect change operations [Noy et al. 2002].

3.2. Problem Statement

Consider two versions of the same source ontology O_X^{j-1} at time $j-1$ and O_X^j at time j, a target ontology O_Y^j, and a set of mappings $\mathcal{L}_{XY}^j$ between O_X^j, and O_Y^j at time j. Suppose that the frequency of new releases of O_X and O_Y is different and at time j only O_X has evolved. We assume that the evolution is likely to provide useful information for mapping refinement of $\mathcal{L}_{XY}^j$, to enrich semantic relations and obtain the refined mappings $\mathcal{L}'^j_{XY}$. All mappings in $\mathcal{L}_{XY}^j$ have initially the type of semantic relation *equivalent* [$\equiv$] or *overlapped* [$\approx$] and we assume them as a mapping candidate set.

Given a mapping $m_{12} \in \mathcal{L}_{XY}^j$ associated with a concept c_1 affected by changes in the ontology, the challenging issue is to determine an exact and suited action of refinement to apply to m_{12}. To address this challenge, we define and formalize a set of *mapping refinement actions* (*cf.* Section 4.1).

The mapping refinement actions are part of refinement procedures, playing a key role to improve the quality of mappings. The objective is to enrich the mapping set by considering different semantic relations between concepts, for instance, equivalence relations can be refined to *is-a* or *part-of*.

In this investigation, we study how $\mathcal{L}_{XY}^j$ can be refined (*e.g.*, new mapping relations derived) based on ontology changes related to ontology evolution. The refined output consists of the $\mathcal{L}'^j_{XY}$. In particular, we address the following research questions:

- How to exploit ontology change operations for mapping refinement?
- Is it possible to reach mapping refinement without applying a new matching operation in the whole target ontology?

4. Refinement of Ontology Mappings

We propose and formalize a set of refinement actions aiming at refining mapping sets (Subsection 4.1) and how these actions are applicable in a refinement procedure (Subsection 4.2).

4.1. Refinement Actions

We present an approach to refine ontology mappings based on different types of ontology changes (Table 1). The proposal explores OCOs for refining mappings individually. For this purpose, we define actions as pre-defined behaviours of mapping refinement into algorithms designed to enrich ontology mappings according to ontology evolution (*cf.* Section 4.2).

The distinct actions representing different possibilities for refining mappings include: *mapping movement*, *mapping derivation*, *semantic relation modification* and *no action*. In the following, we formally describe each action. To this end, let $m_{12} \in \mathcal{L}^j_{XY}$ be the mapping between two particular concepts $c_1 \in O^j_X$ and $c_2 \in O^j_Y$.

Mapping derivation source. This is an action for which an existing mapping from $\mathcal{L}^j_{XY}$ derives a new mapping with the same target concept and different source concept. This action results in addition of a new mapping m_{k2} to $\mathcal{L}'^j_{XY}$.

$$\begin{aligned} &deriveS(m_{12}, c_k) \longrightarrow m_{12} \in \mathcal{L}^j_{XY} \wedge m_{k2} \notin \mathcal{L}^j_{XY} \wedge \\ &(\exists c_k \in O^j_X, m_{k2} \in \mathcal{L}'^j_{XY} \wedge sim(c_1, c_k) \geq \sigma) \wedge \\ &m_{12} \notin \mathcal{L}'^j_{XY} \end{aligned} \tag{4}$$

where $sim(c_1, c_k)$ denotes the similarity between c_1 and $c_k \in neighborhood(c_1)$, and σ denotes the threshold used to compare the derived mapping.

Mapping derivation target. This is an action for which an existing mapping m_{12} in $\mathcal{L}^j_{XY}$ derives a new mapping with the same source and a different target. This action results in addition of a new mapping m_{1v} to $\mathcal{L}'^j_{XY}$.

$$\begin{aligned} &deriveT(m_{12}, c_v) \longrightarrow m_{12} \in \mathcal{L}^j_{XY} \wedge m_{1v} \notin \mathcal{L}^j_{XY} \wedge \\ &(\exists c_v \in O^j_Y, m_{1v} \in \mathcal{L}'^j_{XY} \wedge sim(c_1, c_v) \geq \sigma) \wedge \\ &m_{12} \in \mathcal{L}'^j_{XY} \end{aligned} \tag{5}$$

Semantic relation modification. This is an action in which the type of the semantic relation of a given mapping is modified. This action is designed for supporting the refinement of mappings with different types of semantic relations rather than only considering the type of equivalence relation ($\equiv$).

$$\begin{aligned} &modSemType(m_{12}, new\_semType_{12}) \longrightarrow m_{12} \in \mathcal{L}'^j_{XY} \wedge \\ &new\_semType_{12} \in \{\perp, \equiv, \leq, \geq, \approx \wedge semType_{12} \neq new\_semType_{12}\} \end{aligned} \tag{6}$$

The action for the modification of semantic relation can be applied in conjunction with the actions of move of mapping and derivation of mapping. That is when moving a mapping, it is also possible to modify the type of the semantic relation of such mapping. The same applies for derivation of mapping.

4.2. Refinement Procedure

The mapping refinement phase takes into account concepts from one version of the source ontology to another (O_X^{j-1} and O_X^j) to refine a candidate mapping set (suggests modifications on the mappings). The necessary instances of OCOs are identified from one ontology version at time $j-1$ to another at time j with a *diff* computation [Hartung et al. 2013]. It generates a *diff*, which is basically a set of changes identified between two versions of the same ontology. This article considers only the changes affecting O_X^j, *i.e.*, *diff*(O_X^{j-1}, O_X^j).

The candidate mapping set $\mathcal{L}_{XY}^j$ undergoes the mapping refinement procedure. We describe the procedure in two phases:

1. The output of executed ontology change detection tools is used to identify mappings with potential of refinement. The identification is based on the type of ontology evolution operations that affected the concepts. For instance, the addition of a concept to an ontology may indicate a specialization of another concept (*e.g.*, in O_X^j, the concept "*Eagle*" was added as child of the concept "*Bird*", being the former a specialization of the latter). Therefore, any candidate mapping involving the concepts "*Eagle*" and "*Bird*" are identified with possibility of refinement.
2. After the selection of mappings for refinement, for each selected mapping from $\mathcal{L}_{XY}^j$, an action is executed based on the type of ontology change. The action may include local rematch between concepts, a direct decision to perform modification in the semantic relation of the candidate mapping (*e.g.*, a $\equiv$ relationship may be replaced with a $\sqsubseteq$), or other appropriate action. The final output refers to the mapping set $\mathcal{L}'^j_{XY}$.

Algorithm 1 presents the main procedure to refine $\mathcal{L}_{XY}^j$. The input is the candidate mappings $\mathcal{L}_{XY}^j$ and the $diff_{(O_X^{j-1}, O_X^j)}$. For each mapping $m_{12} \in \mathcal{L}_{XY}^j$, the algorithm verifies if the concept $c_1 \in O_X^j$ was affected by change operations with the use of the $diff_{(O_X^{j-1}, O_X^j)}$. The algorithm then invokes the appropriate procedure for each case by considering addition change operations and revision change operations. If the concept was not affected by change operations from the $diff_{(O_X^{j-1}, O_X^j)}$, then no action is applied to (m_{12}). The output is the refined mapping $\mathcal{L}'^j_{XY}$.

We grouped the OCOs into two categories: (i) **AdditionOCO** adds concepts or information to concepts into the ontology. It consists of OCOs by including: $addC(c)$, $addInnerC(c_s, p_s)$, $addLeafC(c_s, p_s)$, $revokeObsolete(c)$, $addA(a, c_s)$ and $addR(r, c_{s1}, c_{s2})$; (ii) The **RevisionOCO** group of ontology changes revise existing concepts. It consists of OCOs such as: $merge(C_k, c_s)$ and $split(c_i, C_s)$. In the following, we explain the procedures involved by Algorithm 1.

AdditionProcedure. This procedure is invoked when c_1 was affected by some OCO in the *AdditionOCO* group. Algorithm 2 presents the proposed strategy for refining mappings associated to addition changes. For each mapping m_{12}, the neighborhood of the both c_1 and c_2 is retrieved to perform a local rematch. The *rematch* function receives a set of source concepts C_1 and a set of target concepts C_2 and returns a similarity matrix (*simMatrix*). The objective in applying a local rematch is to compare the similarities between the neighborhood of the source and target concepts. The similarities values found then drive modifications to the semantic relation established in m_{12}.For ex-

Algorithm 1 Mapping refinement procedure.

Require: $\mathcal{L}^{j}_{XY}; diff_{(O^{j-1}_X, O^{j}_X)}$

for all $m_{12} \in \mathcal{L}^{j}_{XY}$ **do**
  for $c_1 \in m_{12}$ **do**
    if $AdditionOCO(c_1) \in diff_{(O^{j-1}_X, O^{j}_X)}$ **then**
      $AdditionProcedure(m_{12})$
    else if $RevisionOCO(c_1) \in diff_{(O^{j-1}_X, O^{j}_X)}$ **then**
      $RevisionProcedure(m_{12}; diff_{(O^{j-1}_X, O^{j}_X)})$
    end if
  end for
end for
return $\mathcal{L}'^{j}_{XY}$

Algorithm 2 Mapping refinement for addition changes.

Require: m_{12}

for $c_1 \in m_{12}$ **do**
  $neighC_1 \leftarrow neighborhood(c_1)$;
  $neighC_2 \leftarrow neighborhood(c_2)$;
  $simMatrix(C_1, C_2) \leftarrow rematch(neighC_1, neighC_2)$;
  for all $(c_{1i}, c_{2i}) \in simMatrix(C_1, C_2)$ **do**
    if $c_{1i} = sup(c_1)$ **and** $(sim(c_{1i}, c_2) > sim(c_1, c_2))$ **then**
      $semType \leftarrow relation(c_{1i}, c_1)$;
      $modSemTypeM(m_{12}, semType)$;
      $deriveS(m_{12}, c_{1i})$;
    end if
    if $(c_{2i} = sup(c_2)$ **or** $c_{2i} = sub(c_2))$ **and**
    $sim(c_1, c_{2i}) => sim(c_1, c_2)$ **then**
      $deriveT(m_{12}, c_{2i})$;
    end if
  end for
end for

ample, if $sim(sup(c_1), c_2) > sim(c_1, c_2)$, the algorithm modifies the semantic relation in m_{12} to the same semantic relation of $sup(c_1)$ and c_1 and add a new mapping between $sup(c_1)$ and c_2. The local rematch also helps establishing a derivation of mapping when the $sim(c_1, sub(c_2)) \geq sim(c_1, c_2)$ or $sim(c_1, sup(c_2)) \geq sim(c_1, c_2)$.

RevisionProcedure. This procedure is used to refine mappings when c_1 was affected by some OCO in the *RevisionOCO* group. Algorithm 3 describes the proposed strategy for the refinement. For each input mapping m_{12}, the algorithm retrieves the concepts from O^{j-1}_X involved in merge or split ontology change operations. In the *merge* operation, an initial set of concepts $C_k \subset O^{j-1}_X$ gives place to a concept $c_1 \in O^{j}_X$. On the other hand, in a *split* operation, an initial concept $c_1 \in O^{j-1}_X$ is split in a set of concepts $C_s \subset O^{j}_X$.

The algorithm extracts the before evolution concepts c_1 (in the split) and the set of

Algorithm 3 Mapping refinement for revision changes.

Require: $m_{12}; diff_{(O_X^{j-1}, O_X^j)}$

for $c_1 \in m_{12}$ **do**
  if $split(c_i, C_s) \in diff_{(O_X^{j-1}, O_X^j)}$ **and** $(c_1 \in C_s)$ **then**
    if $sim(c_i, c_2) > sim(c_1, c_2)$ **and** $semType(m_{12}) = \equiv$ **then**
      $modSemTypeM(m_{12}, \leq)$;
    end if
  end if
  if $merge(C_k, c_1) \in diff_{(O_X^{j-1}, O_X^j)}$ **then**
    $neighC_2 \leftarrow neighborhood(c_2)$;
    $simMatrix(C_k, C_2) \leftarrow rematch(C_k, neighC_2)$;
    for all $(c_{ki}, c_{2i}) \in simMatrix(C_k, C_2)$ **do**
      if $(c_{2i} = sup(c_2)$ **or** $c_{2i} = sub(c_2))$ **and** $sim(c_{ki}, c_{2i}) \geq sim(c_1, c_2)$ **then**
        $deriveT(m_{12}, c_{2i})$;
      end if
    end for
  end if
end for

concepts C_k (in the merge) and compares them with $c_2 \in m_{12}$. Our aim is to explore the similarities between c_2 and these concepts and set of concepts before the revision to O_X^j, to extract information and refine m_{12}.

For example, useful information for refinement would be the similarity of the concept $c_i \in O_X^{j-1}$ involved in the split of $c_1 \in m_{12} \wedge c_1 \in C_s$, and c_2. If $sim(c_i, c_2) > sim(c_1, c_2)$, we can infer that c_1 and c_2 do not hold an $\equiv$ relation and, thus, refine the semantic relation of m_{12}.

5. Application and Discussion

We illustrate our approach via two scenarios where the refinement procedures are applied to a pair of ontologies in the biomedical domain, O_X^j and O_Y^j, at time j. We explore concepts described in different natural languages (English and Portuguese, respectively).

5.1. Scenario 1

In this first scenario, we explore the refinement procedure applied to addition change OCO. Ontology O_X evolved over time, generating two different versions from $j-1$ to j. Concept c_1 "*Angina*" in ontology O_X^j is added as a sub concept of "*Heart*".

A set of mappings $\mathcal{L}_{XY}^j$ between O_X^j and O_Y^j, on time j, is given as input for the refinement procedure. Figure 1 illustrates the mapping $m_12 \in \mathcal{L}_{XY}^j$ between concepts c_1 "*Angina*" and c_2 "*Cardiopatia*". The refinement procedure requires as input the list of change operations (OCOs) detected from one version of the ontology to another.

Our proposal leverages the evolution information to refine the proposed mapping set by computing the similarity values between concept c_1 "*Angina*" and the concepts

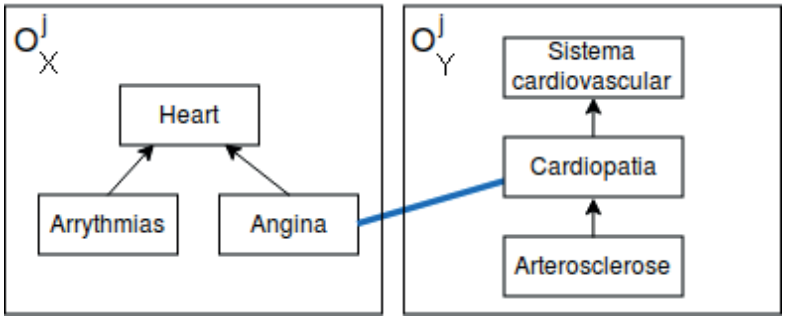


Figure 1. Illustration of the mapping $m_s t \in \mathcal{L}^j_{XY}$ candidate for refinement.

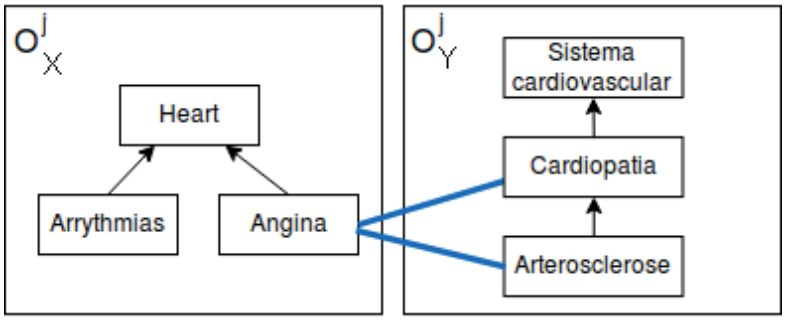


Figure 2. Resulting $\mathcal{L}'^j_{XY}$ after our refinement procedure (application of the derivation action).

of the neighborhood of target concept in j "*Cardiopatia*". To this end, we perform a cross-lingual local rematch. This local rematch is represented in step 2 of Algorithm 2.

If the similarity value between the concepts c_1 "*Angina*" and some neighbor c_{2i} of c_2 is higher than the original similarity value given by $sim(c_1, c_2)$, *i.e.* $sim(c_1, c_{2i}) \geq sim(c_1, c_2)$, the algorithm derives a mapping between c_1 and c_{2i} to reflect this finding, as illustrated in Figure 2. The input for the refinement procedure is the mapping sets and the list of change operations (OCO) affecting the source ontology.

5.2. Scenario 2

The second scenario, concept "*Cardiopathy*" in ontology O_X is split into a set of concepts $C_s \subset O^j_X$: "*Arrhythmias*" and "*Angina*".

Split operations $split(c_i, C_s)$ are used in ontology engineering to represent specializations of a given concept [Noy et al. 2002], creating a [$\leq$] relation between the original concept c_i and the concepts in the $C_s \subset O^j_X$. In order to leverage this information to modify the semantic relation of the candidate mapping between concepts "*Angina*" in O^j_X and "*Cardiopatia*" in O^j_Y, our proposed algorithm calculates similarity values between the original concept in concept from the ontology at time $j-1$ ("*Cardiopathy*") and the target concept in j "*Cardiopatia*". To this end, we perform a cross-lingual local rematch, represented in step 9 of Algorithm 3.

If the similarity $sim(c_i, c_2)$ between concepts c_i "*Cardiopathy*" and c_2 "*Cardiopatia*" is higher than the the original similarity value given by $sim(c_1, c_2)$ involving the concepts c_1 "*Angina*" and the concept c_2 , *i.e.* $sim(c_i, c_2) \geq sim(c_1, c_2)$, the algorithm modifies the semantic relation between c_1 and c_2 to reflect this finding (*cf.* Figure 3).

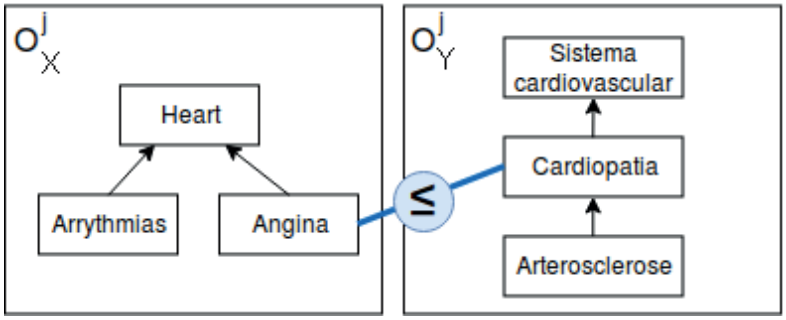


Figure 3. Cross-lingual mapping $\mathcal{L}'^{ij}_{XY}$ after refinement procedure.

6. Discussion and Conclusion

Ontology mapping refinement remains an open research problem. The result of mapping refinement increases the usefulness of mapping sets, benefiting the semantic data integration of systems. This article proposed an original approach with the use of ontology change operations detected during ontology evolution to leverage mapping refinement.

We assumed that ontology evolution information is useful to decide the more adequate approach for the refinement and improve the mapping quality outcome. To the best of our knowledge, the use of ontology change operations for mapping refinement has never been proposed in literature. This aspects refers to the key originality of this paper.

Answering our research questions (i) *how to exploit ontology change operations for mapping refinement* and (ii) *if it is possible to reach mapping refinement without applying a new matching operation in the whole target ontology*: the actions defined based on the change operations and performed during the refinement procedure enrich the proposed mapping set with semantic context, which is beneficial for ontology merging and system integration. Our proposal defined algorithms that reach mapping refinement without applying a new matching operation with the whole target ontology.

Future work involves the investigation and systematic experimentation of our approach using real-world ontologies.

Acknowledgments

We thank CAPES (grant #88881.145912/2017-01), CNPq (grant #307560/2016-3), the São Paulo Research Foundation (FAPESP) (Grants #2017/02325-5 and #2013/08293-7)[2].

References

Aleksovski, Z., Klein, M., ten Kate, W., and van Harmelen, F. (2006). Matching unstructured vocabularies using a background ontology. In *Proceedings of the 15th International Conference on Managing Knowledge in a World of Networks*, pages 182–197.

Arnold, P. and Rahm, E. (2014). Enriching ontology mappings with semantic relations. *Data & Knowledge Engineering*, 93:1–18.

Dinh, D., Dos Reis, J. C., Pruski, C., Da Silveira, M., and Reynaud-Delaître, C. (2014). Identifying relevant concept attributes to support mapping maintenance under ontology evolution. *Journal of Web semantics*, 29:53–66.

[2]The opinions expressed in this work do not necessarily reflect those of the funding agencies.

Dos Reis, J. C., Pruski, C., Da Silveira, M., and Reynaud-Delaître, C. (2014). Understanding semantic mapping evolution by observing changes in biomedical ontologies. *Journal of biomedical informatics*, 47:71–82.

Euzenat, J. and Shvaiko, P. (2013). *Ontology matching*, pages 42,71–84,306. Springer, Heidelberg, 2nd edition.

Gross, A., Hartung, M., Thor, A., and Rahm, E. (2012). How do computed ontology mappings evolve?-a case study for life science ontologies. In *Proceedings of the 2nd Joint Workshop on Knowledge Evolution and Ontology Dynamics*, volume 890.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43:907 – 928.

Hamdi, F., Safar, B., Niraula, N. B., and Reynaud, C. (2010). Taxomap alignment and refinement modules: Results for oaei 2010. In *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010)*, volume 689, pages 212–219.

Hartung, M., Gross, A., and Rahm, E. (2013). COnto-Diff: Generation of Complex Evolution Mappings for Life Science Ontologies. *Journal of Biomedical Informatics*, 46:15–32.

Noy, N. F., Musen, M. A., et al. (2002). Promptdiff: A fixed-point algorithm for comparing ontology versions. *AAAI/IAAI*, 2002:744–750.

Raunich, S. and Rahm, E. (2011). Atom: Automatic target-driven ontology merging. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 1276–1279. IEEE.

Sabou, M., d'Aquin, M., and Motta, E. (2008). Exploring the semantic web as background knowledge for ontology matching. *Journal on data semantics XI*, pages 156–190.

Spiliopoulos, V., Valarakos, A., and Vouros, G. (2008). Csr: discovering subsumption relations for the alignment of ontologies. *The Semantic Web: Research and Applications*, pages 418–431.

Stoutenburg, S. K. (2008). Acquiring advanced properties in ontology mapping. In *Proceedings of the 2nd PhD Workshop on Information and Knowledge Management (PIKM 2008)*, pages 9–16. ACM.

Trojahn, C., Fu, B., Zamazal, O., and Ritze, D. (2014). State-of-the-art in multilingual and cross-lingual ontology matching. In *Towards the Multilingual Semantic Web*, pages 119–135. Springer.

Zhang, S. and Bodenreider, O. (2007). Experience in aligning anatomical ontologies. *International journal on Semantic Web and information systems*, 3(2):1.

Análise Integrada de Grafos de Proveniência Heterogêneos por meio de uma Abordagem *PolyStore*

Yan Mendes[1], **Victor Ströele**[1], **Daniel de Oliveira**[2], **Kary Ocaña**[3]

[1]Universidade Federal de Juiz de Fora (UFJF) - Juiz de Fora - MG – Brasil

{yanmendes,victor.stroele}@ice.ufjf

[2]Universidade Federal Fluminense (UFF) - Niterói - RJ – Brasil

danielcmo@ic.uff.br

[3]Laboratório Nacional de Computação Científica (LNCC) - Petrópolis - RJ – Brasil

karyann@lncc.br

Os autores gostariam de agradecer a FAPEMIG, FAPERJ, CAPES, CNPq e PTI-Lasse por financiarem parcialmente a pesquisa apresentada nesse artigo.

Abstract. *Workflows' provenance data are captured by several existing Wofkflow Management Systems (WfMSs). Distinct WfMSs use different storing formats to represent data and, usually, captures and store data in different granularities using a graph-like shape. This allows researchers to analyze and validate their workflows' results. Yet, in more complex scenarios where scientists need to compare provenance data originated from different WfMSs and workflows, a challenge emerges. To solve this problem, we propose an approach named* `PolyFlow`*, based on Polystore systems, being able to integrate multiple heterogeneous provenance databases adopting an on-demand global schema (ProvONE), i.e., it transforms the data in execution time, allowing researchers to query multiple provenance graphs via , exploring and linking provenance of different workflows. To assess* `PolyFlow`*'s viability, we developed conceptual to two WfMSs (Swift/T and Kepler) using a real experiment to analyze phylogenetic data.*

Resumo. *Dados de proveniência de um workflow são capturados por quase todos os Sistemas de Gerência de Workflows (SGWfs) existentes. Cada SGWf utiliza um formato próprio para representar tais dados, e, comumente, captura e armazena os dados em diferentes granularidades na forma de um grafo. Isso permite cientistas analisarem e validarem resultados de um workflow específico. Entretanto, em cenários mais complexos em que o cientista necessita analisar grafos de proveniência oriundos de múltiplos SGWfs e workflows, um desafio surge. Para resolver esse problema, propomos uma abordagem chamada* `PolyFlow`*, que se baseia no conceito de Sistemas PolyStore, sendo capaz de integrar diversos bancos de dados de proveniência heterogêneos adotando um esquema ProvONE global sob demanda, i.e., sem necessidade de conversão prévia dos dados (que pode ser bastante custosa). Os cientistas podem, então, consultar múltiplos grafos de proveniência nesse banco de dados integrado via* `PolyFlow`*, explorando e vinculando a proveniência de workflows diferentes. De forma a analisar a viabilidade da abordagem* `PolyFlow`*, desenvolvemos mapeamentos para dois SGWf (Swift/T e Kepler) utilizando um experimento real de análise de dados filogenéticos.*

1. Introdução

Nos últimos anos, os *Workflows* Científicos (Wfs) têm se tornado um padrão *de fato* para representar experimentos científicos baseados em simulações computacionais

[de Oliveira et al. 2019]. Um Wf é uma abstração capaz de representar uma sequência lógica de invocações de programas e/ou serviços (*i.e.,* atividades) e suas dependências de dados [Mattoso et al. 2010]. Os Wfs podem ser implementados de diversas formas (*e.g.,* manualmente, via *scripts*), mas são comumente executados por meio de Sistemas de Gerência de *Workflows* (SGWfs).

Os SGWfs são mecanismos complexos desenvolvidos para facilitar a modelagem e a execução dos Wfs. Existem diversos SGWfs como Kepler [Altintas et al. 2004] e o Swift/T [Wozniak et al. 2013], cada um com nichos específicos (*e.g.,* bioinformática, botânica, *etc.*). Além da execução dos Wfs, os SGWfs são responsáveis por oferecerem outras capacidades fundamentais, *e.g.,* a captura de dados de proveniência [Freire et al. 2008], que descrevem todos os artefatos de dados que uma execução de um Wf usou e produziu, bem como as transformações que os dados sofreram. Dados de proveniência em Wfs são baseados em objetos (dados e programas) e seus relacionamentos (dependências) [Moreau et al. 2008], sendo tipicamente representados na forma de um grafo acíclico dirigido (DAG).

Independentemente da estrutura do Wf, o grafo de proveniência resultante é um DAG. A presença de estruturas de repetição, controle de fluxo e paralelismo, por exemplo, afeta somente a complexidade do Wf, mas não a complexidade do grafo de proveniência. Um banco de dados de proveniência é, portanto, composto de vários DAGs cujos nós possuem estruturas variáveis (*i.e.,* os nós podem desempenhar diferentes papéis). Cada nó do grafo pode possuir muitas dependências. A capacidade de análise e interoperabilidade de tais grafos tem sido o objeto de várias iniciativas da comunidade, resultando na recomendação PROV do W3C [Moreau and Groth 2013]. Além disso, extensões do PROV específicas para dados de proveniência de Wfs como o ProvONE [Prabhune et al. 2018] também foram propostas. A vantagem do ProvONE é que o mesmo é capaz de representar a proveniência prospectiva do Wf (*p-prov* - sua definição) e não somente a retrospectiva (*r-prov* - histórico de execução).

Entretanto, apesar dos SGWfs representarem um avanço, os mesmos ainda carecem de mecanismos que auxiliem na análise dos dados de proveniência se considerarmos o experimento científico como um todo, e não somente Wfs de forma isolada. No cenário atual de ciência colaborativa, um Wf pode ser considerado apenas parte de um experimento científico complexo. Tomemos como exemplo a Rede Avançada em Biologia Computacional (Rabicó)[1], que tem como objetivo desenvolver um aparato computacional para apoiar a análise dos dados aplicados a modelos biológicos. A rede possui membros de diversos institutos e universidades como o LNCC e a COPPE/UFRJ. Nesse projeto, duas ou mais equipes geograficamente distribuídas comumente trabalham independentemente com temas em comum, como por exemplo análise filogenética [Ocaña et al. 2011]. Cada equipe eventualmente adota abordagens ligeiramente diferentes, modelando Wfs diferentes em termos de representação (já que cada SGWf possui seu formato próprio) e na escolha dos programas, mas semelhantes em seus objetivos, gerando resultados passíveis de comparação.

Assumamos que dois grupos independentes na mesma rede de pesquisa projetam e implementam dois Wfs, o SciPhy [Ocaña et al. 2011] e o SwiftPhylo [Mondelli et al. 2018], usando diferentes SGWfs, o Kepler e o Swift/T, respectivamente. Cada um dos SGWfs tem suas especificidades, mas ambos são capazes de coletar *r-prov* e *p-prov*. Uma vez que ambos Wfs tem o mesmo objetivo, consomem dados de entrada iguais e geram resultados equivalentes, parece natural tentar usar os grafos de proveniência de suas execuções para comparar e discutir os resultados. No entanto, o Kepler e o Swift/T possuem modelos diferentes de seus bancos de dados de proveniência. Enquanto que o Kepler armazena os dados de proveniência em um Sistema de

[1] `https://www.labinfo.lncc.br/rabico/`

Gerência de Banco de Dados (SGBD) relacional (HSQL), o Swift/T utiliza *logs* que podem ser exportados para um banco de dados MySQL. Além da diferença na representação, cada SGWf grava os dados de proveniência em diferentes granularidades. Assim, embora seja possível que os pesquisadores consultem os dois grafos de proveniência de maneira isolada, a heterogeneidade na representação e a diferença na granularidade dos dados podem dificultar a análise conjunta dos mesmos.

Neste artigo, nos baseamos nesses grafos de proveniência heterogêneos e no ProvONE, como um modelo conceitual canônico, para mostrar como a interoperabilidade de dados de proveniência pode ser alcançada em uma abordagem prática onde podemos assumir um grau de semelhança entre os Wfs e seus grafos de proveniência. Para resolver esta falta de interoperabilidade propomos uma abordagem chamada `PolyFlow` baseada no conceito de Sistemas *PolyStore* [Dziedzic et al. 2016]. Sistemas *PolyStore* são aqueles construídos sobre múltiplos SGBDs heterogêneos e integrados. Além disso, um sistema *PolyStore* se distingue dos SGBDs federados tradicionais uma vez que necessita apenas de um mapeamento das diversas *ilhas* (*i.e.,* formatos dos dados), que são acessados em tempo de execução. Avaliamos o `PolyFlow` com *traces* de Wfs reais (SciPhy e SwiftPhylo) e os resultados apresentaram um *overhead* aceitável, além de reforçar a importância de uma análise integrada dos grafos de proveniência.

Esse artigo se encontra organizado em 4 seções além desta introdução. Na Seção 2 é apresentado o referencial teórico, bem como os trabalhos relacionados. A Seção 3 apresenta o `PolyFlow` e na Seção 4 é conduzida sua avaliação. Por fim, a Seção 5 traz as considerações finais e trabalhos futuros.

2. Referencial Teórico e Trabalhos Relacionados

Nesta seção são apresentados os conceitos necessários para o entendimento deste artigo, como o formalismo de Wfs, proveniência e sistemas *PolyStore*. Além disso, são discutidos os principais trabalhos relacionados.

2.1. *Workflows* e Proveniência

Um Wf pode ser modelado como um grafo $W(A, \phi)$, onde A é o conjunto das atividades de W e ϕ o conjunto das dependências de dados. Dessa forma, temos que $A = \{a_1,a_2, \ldots, a_n\}$ e cada atividade a_i pode ser representada como $a_i(I, P),\ a_i : \{I, P\} \rightarrow O$, onde I é o conjunto de dados de entrada, P os parâmetros e O os dados de saída da atividade a_i. Dessa forma, temos que $I = \{i_1, i_2, \ldots, i_d\}$, onde cada i_d é um arquivo de entrada de a_i, $O = \{o_1, o_2, \ldots, o_k\}$, onde cada o_k é um arquivo de saída de a_i e $P = \{p_1, p_2, \ldots, p_m\}$, onde cada p_m é um parâmetro de a_i.

Cada execução de a_i está associada a uma tupla de m parâmetros $< p_1, p_2, \ldots, p_m >$, onde o valor v_m de cada parâmetro p_m é definido por uma função $\zeta_m(p_m) = v_m$. Consequentemente, temos que o conjunto de dependências de dados $\phi = \{\varphi_{1,2}, \ldots, \varphi_{i,j}\}$, onde cada $\varphi_{i,j} = \langle i_d, a_i, a_j \rangle$, $input(a_i) \in I$, $i_d \neq \varnothing$ e $output(a_i) \in O$. Logo, $\varphi_{i,j} \leftrightarrow \exists o_k \in input(a_j) | O_k \in output(a_i)$.

Um grafo de proveniência G_p gerado a partir da execução de W é definido como $G_p = (V_p, E_p, A_p, T_p)$, onde os nós de V_p representam programas ou artefatos de dados e as arestas de E_p representam a linhagem. Um atributo *type* $\in A_p$ deve existir para todos os nós e arestas. Se um nó v_{pi} representa um programa associado a uma atividade a_i, $Value(v_{pi}, type) = program$. Se esse nó representar um artefato de dados, $Value(v_{pi}, type) = data$. O conjunto A_p também contém outros atributos encontrados nos grafos de proveniência, *e.g.,* nomes de parâmetros de entrada de programas. Os nós que representam os programas também possuem um atributo *name* $\in A_p$. O conjunto T_p representa tipos de arestas que podem ser dos tipos *WasGenera-*

tedBy, Used,WasInformedBy, WasDerivedFrom, WasAttributedTo, WasAssociatedWith ou *ActedOnBehalfOf*, de acordo com a recomendação PROV e o modelo ProvOne.

Especificamente neste artigo, usamos o ProvONE como um modelo canônico capaz de integrar grafos de proveniência (*traces*) de múltiplos Wfs e produzidos pelos diferentes SGWfs. O ProvONE estende a recomendação PROV do W3C com uma representação explícita de *p-prov*, capturando assim as informações mais relevantes sobre atividades do Wf. O ProvONE é composto por diferentes classes e os relacionamentos entre as mesmas. Por questões de restrição de espaço, somente as classes principais são detalhadas nesta seção. Mais informações podem ser obtidas em [Prabhune et al. 2016].

A classe *Program* representa uma tarefa computacional (atividade) que consome e produz dados por meio de suas portas. As instâncias da classe *Program* podem ser atômicas ou compostas. Uma *Port* habilita um *Program* a enviar ou receber dados e/ou parâmetros. Dependências de dados são explicitadas por meio da classe *Channels* que conecta dois ou mais *Programs* por meio de suas *Ports*. A classe *Workflow* representa um tipo especial de *Program*, *i.e.,* uma composição recursiva de programas. A classe *Execution* representa a execução de um *Program*. A classe *User* representa o usuário responsável pela execução. E, finalmente, a classe *Entity* representa as unidades básicas de informação consumidas ou produzidas por um *Program*.

2.2. Sistemas *PolyStore*

PolyStore é um conceito emergente que pode ser visto como um novo tipo de federação de dados, *i.e.,* um sistema de gerência de meta-banco de dados que fornece uma interface unificada e transparente para várias soluções de armazenamento autônomo [Gadepally et al. 2016]. *PolyStores* se diferenciam de sistemas federados tradicionais (que apoiam apenas um modelo de dados), pois é capaz de oferecer suporte para múltiplos modelos de dados. O uso do conceito *PolyStore* visa mitigar os problemas de usabilidade, concedendo aos usuários uma ampla variedade de soluções de armazenamento e linguagens de consulta. Este novo paradigma visa fornecer uma alternativa ao paradigma arquitetural *'one size fits all'*, armazenando e processando diferentes fragmentos de um conjunto de dados geral nos mecanismos que melhor ofereçam apoio a ingestão, consulta e análise de alto desempenho [Gadepally et al. 2016].

Em abordagens *PolyStore*, uma arquitetura conceitual necessita de um *middleware*, que é responsável por manter e orquestrar a submissão e resposta a múltiplas consultas, conhecendo os SGBDs que estão sendo usados para de fato armazenar os dados, direcionando as consultas para as *ilhas* apropriadas. Uma *ilha* é a definição de um modelo de dados e uma linguagem de consulta que representa um tipo de dados. No entanto, os próprios mecanismos de armazenamento podem não oferecer apoio aos formatos de dados e às linguagens de consulta de escolha. É definido também o operador *shim*, responsável por traduzir o modelo de dados e construtos de consulta definidas por uma *ilha* para o modelo e os construtos apoiados pela solução de armazenamento. Na solução proposta por [Gadepally et al. 2016] é possível navegar em diferentes *ilhas*, *i.e.,* podem ser utilizadas linguagens de consulta de diferentes *ilhas* para recuperar dados armazenados em um banco de dados que não pertence à mesma *ilha* que a consulta emitida. Por fim, o operador *cast* é definido para tratar a migração de dados entre soluções de armazenamento diferentes.

2.3. Trabalhos Relacionados

Essa seção apresenta trabalhos encontrados na literatura que abordam o problema de interoperabilidade em grafos de proveniência heterogêneos. Esses trabalhos foram selecionados a partir de um Mapeamento Sistemático da Literatura (disponível em [2] e analisados sob três perspecti-

[2] `https://goo.gl/2qZxUJ`

vas: (i) completude, (ii) usabilidade e (iii) extensibilidade. Sob a perspectiva da completude, os trabalhos foram avaliados considerando a capacidade de capturar *p-prov*, *r-prov* e proveniência evolutiva. Embora proveniência evolutiva não seja definida formalmente na literatura, ela está presente em diversas abordagens [Prabhune et al. 2018]. Quanto à usabilidade, foi avaliada a interface de consulta da solução, mais especificamente, se ela apoia as linguagens de consulta SQL, SPARQL, Cypher e QLP. Essas linguagens foram destacadas, pois, segundo os resultados do mapeamento, são as que os cientistas estão mais familiarizados. Finalmente, para avaliar a extensibilidade, foi considerado o modelo utilizado e o esforço necessário para apoiar os dados descritos por um novo formato para a solução.

[Ellqvist et al. 2009] propõem uma arquitetura baseada em mediadores capaz de interoperar dados de proveniência derivados de diferentes fontes de dados. A arquitetura tem um esquema global que é capaz de representar informações de proveniência expressas por outros modelos, e uma API de consulta capaz de recuperar essas informações de fontes de dados distintas. Os autores implementaram um novo modelo, o *Scientific Workflow Provenance Data Model* - SWPDM), que captura *p-prov* e *r-prov*. [Missier et al. 2010], juntamente com a integração de dados de proveniência heterogêneos, também propõem uma solução para manter o *trace* de proveniência entre as execuções dos Wfs. Como solução de integração, os autores propõem uma extensão do OPM [Moreau et al. 2008]. [Oliveira et al. 2016] propõem uma arquitetura de integração que possui uma camada responsável por transformar os *traces* dos Wfs em fatos Prolog descritos pelo modelo ProvONE, podendo capturar *p-prov* e *r-prov* e evolutiva. Em uma segunda camada é feito o compartilhamento do conhecimento, onde todas as saídas geradas pela primeira camada são armazenadas e podem ser acessadas através de consultas Prolog. [Prabhune et al. 2018] propõem uma estrutura que auxilia os pesquisadores na análise de dados heterogêneos de proveniência. Para isso, eles usam uma solução de armazenamento RDF (Apache Jena TDB), na qual os dados são representados pelo modelo ProvONE. Os usuários podem consultar os dados usando construções SPARQL. Eles implementam três algoritmos de mapeamento que transformam os dados descritos por outros formatos em dados compatíveis com ProvONE.

Em síntese, [Ellqvist et al. 2009] propõem um modelo de dados de proveniência que não captura a proveniência evolutiva. Além disso, toda consulta é feita por meio de uma API, sendo necessário implementar novos *endpoints* para novas consultas. O modelo proposto por [Missier et al. 2010] captura *p-prov* e *r-prov*, mas falta apoio à proveniência evolutiva. É adotado um modelo relacional para armazenar os dados, possibilitando consultas através de construções SQL. [Oliveira et al. 2016] utilizam o modelo ProvONE sendo um indicativo da extensibilidade da solução. Por outro lado, para acessar a base de conhecimento, os usuários devem escrever consultas Prolog o que prejudica a usabilidade da solução. [Prabhune et al. 2018] adotam SPARQL para consultas e, no que diz respeito à extensibilidade, para suportar dados descritos por outros formatos, novos algoritmos de mapeamento devem ser desenvolvidos. Além disso, todas as soluções demandam que uma conversão de todos os dados para o modelo ProvONE seja realizada, o que pode ser bastante custoso.

Considerando o exposto anteriormente, a abordagem proposta neste trabalho visa solucionar as lacunas identificadas. Nesse sentido, as principais contribuições do `PolyFlow` são: (i) ser uma solução extensível, pois utiliza um modelo de dados já definido na literatura e suporta múltiplas soluções de armazenamento, reforçando, consequentemente, seu (ii) apelo de usabilidade, fornecendo suporte a qualquer solução de armazenamento e linguagem de consulta que os usuários em potencial desejarem. Além disso, o `PolyFlow` possui um catálogo interno de mediadores, facilitando a incorporação de novos modelos e eximindo a necessidade da construção de algoritmos de mapeamento. Além disso, por não necessitar de uma conversão completa

para um modelo de dados canônico, o PolyFlow permite que as consultas sejam realizadas sob demanda nos bancos de dados originais.

3. PolyFlow: Integrando Grafos de Proveniência Heterogêneos

A abordagem PolyFlow foi desenvolvida para integrar bancos de dados heterogêneos que representam grafos de proveniência de Wfs. Nesta seção, apresentamos uma extensão do formalismo inicialmente apresentado na Seção 2, o mapeamento de dados necessário e a arquitetura do PolyFlow.

3.1. Formalismo

O PolyFlow se baseia na consulta integrada de múltiplos *traces* de Wfs representados por meio de grafos de proveniência. Dessa forma, devemos definir o conceito de banco de dados de proveniência. Neste artigo, um banco de dados de proveniência $\Im$ pode ser definido como um conjunto de θ grafos G_p, sendo $\Im = \{G_{p1}, G_{p2}, \ldots, G_{p\theta}\}$. Para cada $G_{p\theta} \in \Im$, devemos ser capazes de realizar consultas sobre esses grafos. Assim, a *consulta por valor de parâmetros de traces diferentes* pode ser definida por $Q_M(S, \Im) \Leftrightarrow \{G_{p\theta} \in \Im | \exists p_m \in G_{p\theta} \wedge \lambda(\star, \zeta_m(G_{p\theta}.p_m), v_m) \wedge (p_m, v_m) \in S\}$, onde: (i) Q_M é representado por um conjunto de pares $S = \{(p_1, v_1), (p_2, v_2), \ldots, (p_m, v_m)\}$, onde p_m é o parâmetro a ser consultado e v_m é o valor de referência; e (ii) λ é uma função que compara o valor v_m de um parâmetro p_m com o valor do mesmo parâmetro de um grafo $G_{p\theta}$ por meio $\zeta_m(G_{p\theta.p_m})$ utilizando um operador denominado $\star$. É importante ressaltar que o operador $\star$ pode ser qualquer operador utilizado na álgebra relacional.

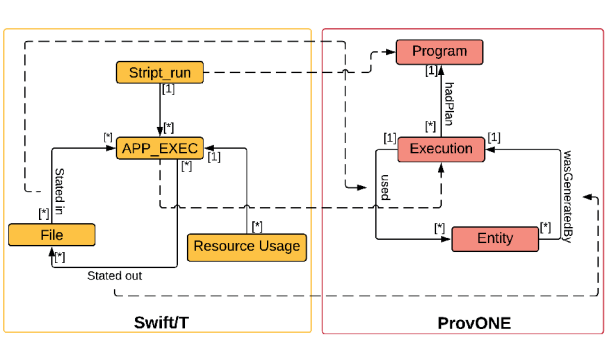
Assumindo que dois grafos $G_{p1}, G_{p2} \in \Im$ possuam parâmetros p_m e p'_m equivalentes ($p_m \equiv p'_m$), mas com denominações diferentes. Logo, dado um conjunto $P_\alpha = \{p_1, p_2, \ldots, p_\mu\} \in G_{p1}$ e $P_\beta = \{p_1, p_2, \ldots, p_\nu\} \in G_{p2}$ e uma linguagem de transformação Υ devemos ser capazes de achar um mapeamento $\upsilon \in \Upsilon$ de forma que cada $p_\mu \in P_\alpha$ e $p_\nu \in P_\beta$, $\tau(p_\mu) = p_\nu$. Logo, em bases de proveniência heterogêneas, a complexidade está em implementar consultas por valores de parâmetros $Q_M(\Gamma, \Im)$ (onde $\Gamma = S_\alpha \cup S_\beta$, $S_\alpha = \{(p_1, v_1), (p_2, v_2), \ldots, (p_\mu, v_\mu)\}$ e $S_\beta = \{(p_1, v_1), (p_2, v_2), \ldots, (p_\nu, v_\nu)\}$), considerando o mapeamento τ entre os parâmetros de *traces* representados em grafos diferentes.

3.2. Mapeamento para o Modelo ProvOne

Conforme mencionado na Seção 3.1, um desafio em uma abordagem *PolyStore* para análise de proveniência é definir o mapeamento τ entre os parâmetros de *traces* representados em grafos heterogêneos. Neste artigo focamos em prover uma solução para análise integrada (com a semântica do domínio de proveniência) a partir de bancos de dados heterogêneos. A estratégia adotada é baseada em mediação [Özsu and Valduriez 2011], que é a conciliação feita para possibilitar a tradução dos dados descritos por esquemas locais a partir de um esquema global. O PolyFlow adota o modelo ProvONE como modelo global (canônico). Assim, a estratégia adotada neste trabalho é mapear modelos de bancos de dados de proveniência diferentes para o modelo ProvONE.

Tais mapeamentos são utilizados pelos componentes do PolyFlow (*Query Resolvers* e *Entity Mappers, explicados na Seção 3.3*) para executar as consultas, sejam elas em um único ou múltiplos grafos de proveniência. No PolyFlow existem dois tipos de mapeamentos possíveis: (i) 1 - 1: onde duas entidades são equivalentes entre os *schemas*; (ii) 1 - N: quando a entidade do *Schema* Global se encontra associada a composição de várias entidades do *Schema* Local, ou seja, é representada de maneira mais granular no *Schema* Local. Note que mapeamentos N - 1 e N - N podem ser expressos por N mapeamentos 1 - 1 e 1 - N, respectivamente. A Figura 1(a) apresenta o *Schema* Local do banco de dados de proveniência do Swift/T e seu mapeamento com

um fragmento do ProvONE. As setas tracejadas indicam os mapeamentos entre os dois *schemas*. Uma vez que a granularidade entre os modelos é diferente, apenas as entidades do ProvONE envolvidas no mapeamento foram representadas. Esse mapeamento é representado por meio de objetos JSON (*i.e., Entity Mappers*, explicado a seguir). A Figura 1(b) apresenta o *Mapper* da entidade APP_EXEC (Swift/T) para a *Execution* do ProvONE.




```
{
  "name": "app_exec",
  "columns": [
    {
      "alias": "execution_id",
      "projection": "app_exec_id"
    },
    {
      "alias": "prov_startedAtTime",
      "projection": "start_time"
    },
    {
      "alias": "provone_hadPlan",
      "projection": "script_run_id"
    },
    {
      "projection": "duration"
    }
  ]
}
```

Figura 1. (a) Mapeamento do Swift/T para o ProvONE; (b) *Mapper* da entidade *APP_EXEC* para *Execution*.

3.3. Arquitetura do `PolyFlow`

A abordagem `PolyFlow` foi projetada para apoiar a interoperabilidade semântica de dados, se baseando no conceito de sistemas *PolyStore* que proveem a interoperabilidade lógica dos dados. Dessa forma, a arquitetura do `PolyFlow` segue os padrões de arquiteturas de sistemas *PolyStore*. A chave para uma abordagem *PolyStore* é que os vários mecanismos de armazenamento são distintos e acessados de forma isolada por meio de seus próprios mecanismos de consulta e de uma interface comum. Uma visão geral da arquitetura é apresentada na Figura 2. A arquitetura do `PolyFlow` é composta de três camadas: (i) Fonte de Dados, (ii) Camada de Mediação e (iii) Camada de Consulta. O código-fonte do `PolyFlow` e todos os mapeamentos realizados nos experimentos apresentados nesse artigo estão disponíveis em `https://github.com/yanmendes/polyflow.api`.

A camada que representa as fontes de dados contém os múltiplos bancos de dados de proveniência $\Im_1, \Im_2, \ldots, \Im_f$, cada uma delas representada em diferentes formatos. Cada banco de dados está relacionado com uma *ilha* do modelo *PolyStore* (*e.g.,* relacional, JSON, XML). No exemplo da Figura 2, podemos visualizar 3 bancos de dados referentes a 3 SGWfs distintos. Na etapa atual da implementação, a arquitetura não conta com implementações do operador *cast* discutido na Seção 2.2.

A camada de mediação é o núcleo do `PolyFlow`. Nessa camada é realizado todo o mapeamento entre as múltiplas ilhas que compõem a abordagem. A camada de Mediação do `PolyFlow` possui três componentes: (i) *Schema* Global, (ii) *Schema Local* e (iii) *Entity Mappers*. Os *Schemas* Locais são os modelos de dados associados a cada banco de dados de proveniência da camada de fonte de dados. O *Schema* Global é o modelo canônico a ser adotado para realizar as consultas no `PolyFlow`. Apesar da arquitetura possibilitar o uso de múltiplos *Schemas* Globais, no momento consideramos apenas o ProvONE. Finalmente, os *Entity Mappers*, confeccionados por usuários que compreendem os modelos de dados dos dois esquemas, realizam mapeamentos de equivalência entre os *Schemas* Locais e o *Schema* Global. Eles são consumidos pelos *Query Resolvers* no momento do processamento das consultas. É importante ressaltar que é nessa camada onde os usuários tendem a dispensar os maiores esforços, pois o mapeamento pode ser uma tarefa árdua. Somente os *Entity Mappers* são persistidos em formato

JSON no catálogo interno do PolyFlow. O formato JSON foi escolhido por sua popularidade e simplicidade, mitigando a curva de aprendizado que outros formatos mais semânticos (e.g. RDF) impõe por sua complexidade.

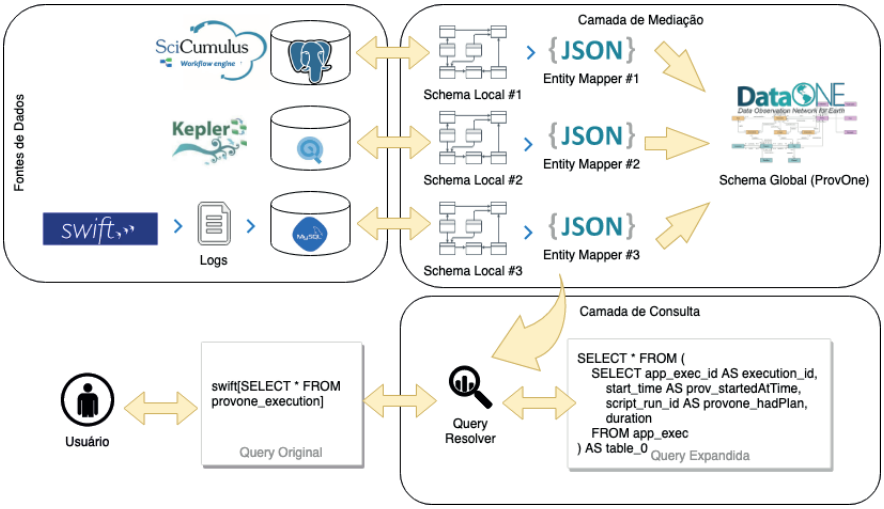


Figura 2. Arquitetura do PolyFlow

Finalmente, a camada de consulta disponibiliza para o usuário o *Resolver* que é o componente responsável por converter consultas baseadas em um *Schema* Global (ProvONE) em consultas válidas nos *Schemas* Locais. Estes funcionam de maneira similar ao operador *shim* da arquitetura *PolyStore*, resolvendo as consultas submetidas às ilhas. Dessa forma, pode submetê-las aos mecanismos de consulta da camada de fonte de dados, recuperando os dados de proveniência e representando-os utilizando o modelo ProvONE. O PolyFlow também é capaz de resolver agregações entre pares de entidades do *Schema* Local recursivamente, i.e. um *Entity Mapper* pode ser composto por pares de *Entity Mappers*. Devido à restrições de espaço, impossibilita-se o destrinchamento mais detalhado de componentes da arquitetura e exemplos. Para um mergulho mais aprofundado, veja a pasta *examples* do repositório do PolyFlow que ilustram diferentes casos de uso.

As consultas submetidas ao *Query Resolver* são SQL-*like* e devem seguir a sintaxe *nome-do-mediador[entidade-a-ser-mediada]*. Todas construções SQL válidas são suportadas por PolyFlow. As consultas são expandidas em tempo de execução por meio da substituição dos mediadores encontrados na consulta por uma *subquery* utilizando os elementos presentes no *Entity Mapper* associado. Por exemplo, assumamos que um determinado usuário necessita consultar todas as execuções de Wfs que foram realizadas no SGWf Swift/T. Ao buscar por todas as *Executions* (*provone_execution*), o PolyFlow realiza a expansão de consulta:

```
SELECT * FROM swift[provone_execution];
<=>
SELECT * FROM (
    SELECT app_exec_id AS execution_id,
    start_time AS prov_startedAtTime,
    script_run_id AS provone_hadPlan, duration
    FROM app_exec
) AS table_0;
```

4. Avaliação Experimental

Nesta seção avaliamos a abordagem PolyFlow sob duas perspectivas: **(i)** o desempenho de con-

sultas com a solução proposta em um banco de dados específico (Kepler e Swift/T consultados de forma isolada) e **(ii)** o desempenho do `PolyFlow` em consultas integradas.

4.1. Estudo de Caso

Conforme mencionado na Seção 1, foi utilizado como estudo de caso o experimento de análise filogenética no contexto do projeto Rabicó. Conceitualmente, esse experimento recebe como entrada um *dataset* de sequências de DNA, RNA e proteínas de forma a gerar uma árvore filogenética que indique a relação evolutiva entre os organismos de entrada. O experimento envolve 7 etapas bem definidas, conforme apresentado na Figura 3: (i) *Importação dos Dados*: os dados são coletados de diversos repositórios biológicos, como o RefSeq; (ii) *Numeração das Sequências*: as sequências obtidas são identificadas e numeradas (etapa opcional); (iii) *Alinhamento Múltiplo de Sequências*: as sequências são alinhadas (por programas como MAFFT, Kalign, ClustalW, Muscle, ou ProbCons), *i.e.,* são identificadas regiões similares que possam ser consequência de relações funcionais, estruturais ou evolutivas; (iv) *Conversão de Sequências*: as sequências alinhadas são convertidas para o formato PHYLIP; (v) *Escolha do Modelo Evolutivo*: o melhor modelo evolutivo para as sequências alinhadas é selecionado (etapa mais custosa do experimento); (vi) *Filtragem de Sequência*: as sequências podem ser filtradas ou trimadas para diminuir a complexidade da geração da árvore filogenética (atividade opcional); e (vii) *Geração da Árvore Filogenética*: a árvore filogenética é finalmente gerada, apontando as relações evolutivas entre os organismos.

Nesse artigo consideramos os Wfs SciPhy e SwiftPhylo como variações do mesmo experimento e cujo resultado deve ser analisado de forma integrada. O SwiftPhylo implementa todas as atividades com a exceção da importação dos dados, pois assume que os dados já se encontram disponíveis para processamento. Por outro lado, o SciPhy não implementa as atividades opcionais do experimento, ou seja, é composto por 5 atividades. Além disso, o SwiftPhylo utiliza o MAFFT como programa de alinhamento, enquanto que o SciPhy executa todos os programas de alinhamento e escolhe o alinhamento com melhor qualidade.

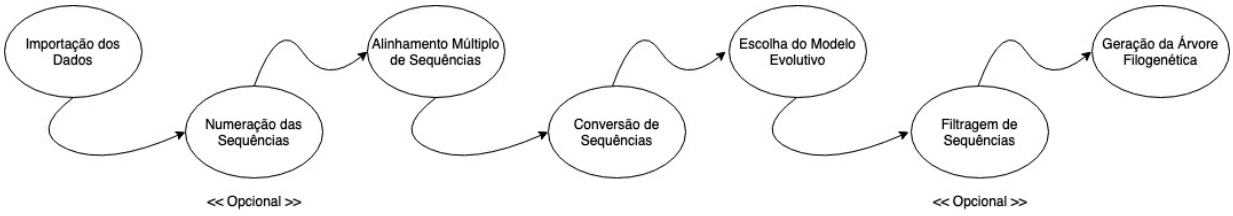


Figura 3. Definição do Experimento de Análise Filogenética

4.2. Resultados

Nesta subseção avaliamos o desempenho do `PolyFlow` em dois experimentos: (i) consultas aos bancos de dados de proveniência de forma isolada e (ii) consulta integrada a dois bancos de dados de proveniência. Para o primeiro experimento consultamos de forma isolada os bancos de dados de proveniência dos SGWf Kepler e Swift/T.

Conforme mencionado na Subseção 3.3, ao submeter uma consulta encapsulada por um mediador, o `PolyFlow` recupera os *Entity Mappers* e expande a consulta utilizando relações do *Schema* Local. Estas operações (*e.g.,* busca dos *Entity Mappers*, expansão da consulta, etc.) introduzem um *overhead* no processamento. Além disso, o tempo de execução de cada consulta no SGBD associado pode aumentar, uma vez que a consulta construída pelo `PolyFlow` não necessariamente é a mais otimizada. Como estratégia de avaliação do *overhead* de definição e expansão das consultas, calculamos o tempo total decorrido desde a requisição ao *endpoint*

do PolyFlow até o recebimento da resposta da consulta e o tempo de execução da consulta propriamente dita no SGBD (hospedado na nuvem *Digital Ocean* no caso do Kepler e localmente no caso do Swift/T).

Utilizamos a entidade *Execution* do modelo ProvONE, a entidade *Entity* e o relacionamento *Used* do PROV (mas que também existem no ProvOne) para avaliar a eficiência das transformações das consultas. Neste contexto, *Used* possui um mapeamento simples (1-1), *Execution* um mapeamento composto (1-2) e *Entity* um mapeamento composto recursivo (1-4). Ao utilizar estas três entidades, conseguimos cobrir os casos de mapeamento descritos na Subseção 3.2. Todos esses *Entity Mappers* estão disponíveis no repositório do Github mencionado anteriormente.

Para essa avaliação inicial, utilizamos consultas simples, projetando todos os atributos dessas entidades e sem considerar agregações. Cada consulta foi executada 10 vezes e os resultados estão dispostos na Tabela 1 (Linhas Kepler e Swift/T). O valor máximo corresponde à primeira requisição, sendo amortizado nas requisições seguintes por estratégias de *caching* utilizadas pelos SGBDS. As requisições foram feitas sequencialmente. O PolyFlow e o SGBD mySQL foram implantados em uma máquina com processador Intel Core i5 @ 2.3GHz, 16GB RAM e SO macOs Mojave 10.14.4. Conforme podemos perceber, uma parcela considerável do tempo total de atendimento da requisição é consumido no processamento da consulta no SGBD associado, *i.e., ilha*. Assim, podemos perceber que o *overhead* adicionado pelo PolyFlow é negligenciável no caso do Kepler (1,6% no pior caso) se comparado ao tempo necessário para processamento da consulta. No caso do Swift/T, como o SGBD estava executando localmente, as consultas foram executadas em um tempo curto se comparado ao tempo que o PolyFlow necessita para expandir a consulta (que é um tempo fixo). Logo, o *overhead* introduzido não é negligenciável nesse caso (90.4% no pior caso). Porém, se consultas mais complexas ou que envolvam a manipulação de um maior volume de dados forem submetidas, a tendência é que o *overhead* reduza em valores percentuais no caso do Swift/T.

O segundo experimento executado avalia consultas integradas aos bancos de dados de proveniência dos SGWfs Kepler e Swift/T. O Wf SciPhy foi modelado no Kepler e o Wf SwiftPhylo no Swift/T. Para essa avaliação integrada comparamos o desempenho do PolyFlow para as consultas executadas de forma integrada, *i.e.,* considerando ambos bancos de dados de proveniência. Assim como no primeiro experimento, utilizamos a entidade *Execution* do modelo ProvONE, a entidade *Entity* e o relacionamento *Used* do PROV. Nesse cenário da avaliação integrada, ambas consultas são submetidas em conjunto ao PolyFlow que se encarrega de distribuir as mesmas para o SGBD associado (*i.e., ilha*) na camada de fonte de dados, conforme exemplificado a seguir:

```
{
    query(query: "SELECT␣*␣FROM␣swift[provone_execution];"),
    query(query: "SELECT␣*␣FROM␣kepler[provone_execution];")
}
```

Assim como no experimento anterior utilizamos consultas simples sem junções ou agregações, projetando todos os atributos dessas entidades. Cada consulta foi executada 10 vezes e os resultados estão dispostos na Tabela 1 (Linha "Integrado"). O valor máximo corresponde à primeira requisição, sendo amortizado nas requisições seguintes por estratégias de *caching* utilizadas pelos SGBDS. As requisições foram feitas sequencialmente.

Como podemos perceber, similarmente ao primeiro experimento, uma parcela considerável do tempo total de atendimento da requisição é consumido no processamento da consulta no SGBD associado (*i.e., ilha*). Porém, é importante ressaltar que no caso da consulta integrada, as requisições aos SGBDs são realizadas em paralelo pelo PolyFlow, *i.e.,* o tempo de consulta

Tabela 1. *Overhead* para diferentes entidades do ProvONE com os dados do Kepler, Swift/T e de forma Integrada.

		Requisição (ms)				Execução no SGBD (ms)				Overhead (ms)				
		min	max	avg	stdev	min	max	avg	stdev	min	max	avg	stdev	avg%
Kepler	Used	842,0	1479,0	1035,3	692,0	834,0	1442,0	1054,5	676,9	6,0	37,0	10,7	29.0	1.6
	Execution	891,0	1132,0	997,1	283,5	885,0	1124,0	985,9	276,0	5,0	49,0	11,2	40,1	1.5
	Entity	903,0	1353,0	1043,8	421,8	895,0	1303,0	1032,7	392,7	5,0	50,0	11,1	41,1	1.5
Swift/T	Used	17.0	78.0	29.9	53.9	9.0	21.0	15.7	15.0	58.0	8.0	14.2	46.7	90.4
	Execution	15.0	42.0	24.6	26.6	9.0	29.0	14.8	17.3	28.0	5.0	9.8	19.8	82.1
	Entity	17.0	108.0	32.6	81.8	11.0	36.0	17.9	24.9	72.0	6.0	14.7	60.6	66.2
Integrado	Used	895.0	1317.0	1076.3	138.4	883.0	1290.0	1056.1	136.7	73.0	7.0	14.6	8.3	1.9
	Execution	894.0	1601.0	1099.8	205.7	887.0	1590.0	1081.5	203.2	72.0	7.0	9.8	1.8	1.7
	Entity	920.0	1785.0	1124.4	258.5	912.0	1775.0	1110.3	258.0	60.0	6.0	8.9	1.8	1.3

computado na Tabela 1 é o máximo entre os tempos de consulta de cada SGBD. Neste caso, sempre as consultas ao banco de dados do Kepler irão dominar o tempo total de execução da consulta, pois eles são consideravelmente superiores aos tempos de execução das consultas no banco de dados do Swift/T. Dessa forma, percebe-se que o *overhead* adicionado pelo `PolyFlow` é negligenciável (1,9% no pior caso) se comparada ao tempo necessário para processamento da consulta. É importante ressaltar que esses resultados foram obtidos a partir de consultas tradicionais da área de proveniência de dados, conforme definido por [de Oliveira et al. 2016]. Entretanto, consultas mais complexas que envolvam dados específicos do domínio, e não somente dados de proveniência, podem ser consideradas, o que aumentaria a complexidade do `PolyFlow`.

5. Considerações Finais e Trabalhos Futuros

Este artigo propõe o `PolyFlow` que possibilita a análise integrada de grafos de proveniência de múltiplos SGWfs, utilizando uma abordagem *PolyStore*. Para isso, foi utilizado o modelo ProvONE como um modelo global para o qual os diferentes bancos de dados de proveniência são mapeados, sob demanda. Bancos de dados de proveniência de experimentos reais da bioinformática executados com os SGWfs Swift/T e Kepler foram integrados utilizando `PolyFlow` para avaliação da abordagem proposta. O `PolyFlow` integrou esses bancos de dados de proveniência heterogêneos de maneira não intrusiva e permitiu ao cientista ter acesso aos dados de proveniência submetendo uma única consulta, permitindo assim uma análise abrangente em um sistema unificado. Resultados demonstram que o *overhead* do `PolyFlow` está relacionado à interação das requisições com o banco de dados, e que é negligenciável na maioria dos casos se comparado ao ganho na obtenção das informações científicas. Até o presente momento o `PolyFlow` não permite que sejam aplicados filtros ou projeções específicas aos resultados integrados de diferentes mediadores. Neste sentido, como trabalho futuro, planejamos implementar uma ilha *PolyStore* e acoplar SGBDs *PolyStore* como o BigDAWG [Gadepally et al. 2016] ao `PolyFlow`, incorporando também novos operadores *shims* e *casts* à arquitetura.

Referências

Altintas, I., Berkley, C., Jaeger, E., Jones, M. B., Ludäscher, B., and Mock, S. (2004). Kepler: An extensible system for design and execution of scientific workflows. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004), 21-23 June 2004, Santorini Island, Greece*, pages 423–424.

de Oliveira, D., Liu, J., and Pacitti, E. (2019). *Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

de Oliveira, W. M., Missier, P., Ocaña, K. A. C. S., de Oliveira, D., and Braganholo, V. (2016). Analyzing provenance across heterogeneous provenance graphs. In *Provenance and Annota-*

tion of Data and Processes - 6th International Provenance and Annotation Workshop, IPAW 2016, McLean, VA, USA, June 7-8, 2016, Proceedings, pages 57–70.

Dziedzic, A., Elmore, A. J., and Stonebraker, M. (2016). Data transformation and migration in polystores. In *2016 IEEE High Performance Extreme Computing Conference, HPEC 2016, Waltham, MA, USA, September 13-15, 2016*, pages 1–6.

Ellqvist, T., Koop, D., Freire, J., Silva, C., and Strömbäck, L. (2009). Using mediation to achieve provenance interoperability. In *Services-I, 2009 World Conference on*, pages 291–298. IEEE.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for Computational Tasks: A Survey. *Computing in Science & Engineering*, pages 20–30.

Gadepally, V., Chen, P., Duggan, J., Elmore, A., Haynes, B., Kepner, J., Madden, S., Mattson, T., and Stonebraker, M. (2016). The bigdawg polystore system and architecture. In *High Performance Extreme Computing Conference (HPEC), 2016 IEEE*, pages 1–6. IEEE.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Ogasawara, E. S., de Oliveira, D., da Cruz, S. M. S., Martinho, W., and Murta, L. (2010). Towards supporting the life cycle of large scale scientific experiments. *IJBPIM*, 5(1):79–92.

Missier, P., Ludäscher, B., Bowers, S., Dey, S., Sarkar, A., Shrestha, B., Altintas, I., Anand, M. K., and Goble, C. (2010). Linking multiple workflow provenance traces for interoperable collaborative science. In *WORKS 2010*, pages 1–8. IEEE.

Mondelli, M. L., Magalhães, T., Loss, G., Wilde, M., Foster, I. T., Mattoso, M., Katz, D. S., Barbosa, H. J. C., de Vasconcelos, A. T. R., Ocaña, K. A. C. S., and Jr., L. M. R. G. (2018). Bioworkbench: A high-performance framework for managing and analyzing bioinformatics experiments. *CoRR*, abs/1801.03915.

Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., and Paulson, P. (2008). The open provenance model: An overview. In *International Provenance and Annotation Workshop*, pages 323–326. Springer.

Moreau, L. and Groth, P. T. (2013). *Provenance: An Introduction to PROV*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers.

Ocaña, K. A., de Oliveira, D., Ogasawara, E., Dávila, A. M., Lima, A. A., and Mattoso, M. (2011). Sciphy: a cloud-based workflow for phylogenetic analysis of drug targets in protozoan genomes. In *BSB11*, pages 66–70. Springer.

Oliveira, W., Missier, P., Ocaña, K., de Oliveira, D., and Braganholo, V. (2016). Analyzing provenance across heterogeneous provenance graphs. In *IPAW*, pages 57–70. Springer.

Özsu, M. T. and Valduriez, P. (2011). *Principles of distributed database systems*. Springer Science & Business Media.

Prabhune, A., Zweig, A., Stotzka, R., Gertz, M., and Hesser, J. (2016). Prov2one: an algorithm for automatically constructing provone provenance graphs. In *IPAW*, pages 204–208. Springer.

Prabhune, A., Zweig, A., Stotzka, R., Hesser, J., and Gertz, M. (2018). P-PIF: a provone provenance interoperability framework for analyzing heterogeneous workflow specifications and provenance traces. *Distributed and Parallel Databases*, 36(1):219–264.

Wozniak, J. M., Armstrong, T. G., Wilde, M., Katz, D. S., Lusk, E. L., and Foster, I. T. (2013). Swift/t: Large-scale application composition via distributed-memory dataflow processing. In *13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013, Delft, Netherlands, May 13-16, 2013*, pages 95–102.

SAVIME: A Database Management System for Simulation Data Analysis and Visualization

Hermano Lustosa[1], **Fabio Porto**[1], **Patrick Valduriez**[2]

[1]National Laboratory for Scientific Computing (LNCC)
Petrópolis – RJ – Brazil

[2]Inria and LIRMM
Montpellier – France

{hermano,fporto}@lncc.br, patrick.valduriez@inria.fr

Abstract. *Limitations in current DBMSs prevent their wide adoption in scientific applications. In order to make scientific applications benefit from DBMS support, enabling declarative data analysis and visualization over scientific data, we present an in-memory array DBMS system called SAVIME. In this work we describe the system SAVIME, along with its data model. Our preliminary evaluation show how SAVIME, by using a simple storage definition language (SDL) can outperform the state-of-the-art array database system, SciDB, during the process of data ingestion. We also show that is possible to use SAVIME as a storage alternative for a numerical solver without affecting its scalability.*

1. Introduction

The increasing computational power of HPC machines allows for performing complex numerical simulations. These simulations produce huge datasets, which are analyzed and visualized to enable researchers to gain insights about the phenomena being studied. Traditionally, simulation code stores its raw data in the file system, and another application reads it from disk, performs analysis and creates the visualization files. However, due to the I/O gap in HPC environments, doing so can be very inefficient for large scale simulations [Ahrens 2015]. Two popular approaches, in-situ and in-transit analysis [Oldfield et al. 2014], have been proposed to address this problem by favoring intense memory usage instead of relying on disk storage.

DBMSs are not commonly adopted in any of these approaches. In-situ and in-transist analysis relies on libraries, and the post-processing approach consists of storing data in scientific data formats such as HDF[Group 2017] and NetCDF[Unidata 2017]. DBMSs are considered inadequate for scientific data management due to many factors. The first one is the impedance mismatch problem [Blanas et al. 2014, Gosink et al. 2006], i.e., the incompatibilities between the representation formats of the source data and the DBMS. This impedance mismatch yields costly conversions between formats, which adds prohibitive overhead during data ingestion.

Also, data usage patterns for scientific applications are very different from those common in commercial applications. The workload is mainly analytical and not necessarily all data needs to be persisted. It is common that data is analyzed and summarized, having its volume being drastically reduced by either storing only the summarized version or by simply discarding parts which are not interesting. Some other key points also do not favor

DBMS usage. Data is heterogeneous, comprising different formats and sources, making it hard to use a single database solution for all data. The analysis is also complex and, in many cases, cannot be easily done with a declarative language like SQL.

However, for the vast variety of queries that can be expressed in a declarative language, a DBMS solution could offer a convenient and efficient way to perform analysis, given that the underlying data model is flexible enough to accomodate such data, and that the process of data ingestion is seamless, not incuring in costly data conversions.

Therefore, given the current lack of a database solution that could facilitate the simulation data analysis and visualization, we propose an array based data model [Lustosa et al. 2017] named TARS, to cope with simulation data and to allow a more efficient representation, along with a prototype system that implements this model, currently named SAVIME. SAVIME supports a DDL and a SDL that enable fast data ingestion, and a DML that allows for declarative analysis and visualization of simulation data.

In this paper, we present SAVIME and the TARS data model, along with an evaluation in which we compare SAVIME with SciDB, the state-of-the-art array DBMS. This document is organized as follows. In Section 2 we discuss the TARS data model implemented in SAVIME. In Section 3 we present SAVIME, its execution model and its DDL, SDL and DML. In Section 4 we show the results of our evaluation comparing SAVIME and SciDB and embedding SAVIME with a real life application. In section 5, we discuss the related work and finally in Section 6 we conclude.

2. Typed Array Data Model

Scientific data is usually represented as multidimensional arrays, which are common as the result of scientific experiments, measurements and simulations. In short, an array is a regular structure formed by a set of dimensions. A set of indexes, one per dimension, identifies a cell that contains values for array attributes.

If carefully designed, arrays offer advantages when compared to tables. Cells in an array are ordered, unlike tuples in a relation. Thus, an array DBMS can quickly lookup cells by taking advantage of this ordering. Using arrays instead of tables can also save storage space, since array indexes do not need to be stored and can be inferred by the position of the cell. Furthermore, arrays can be split into subarrays, called tiles or chunks. These subarrays are used as processing and storage units, and help answering queries efficiently. Thus, if your data is array-ike, using an array database is the way to go.

However, current array data model implementations, e.g., SciDB and RasDaMan, have limitations, preventing a more wide acceptance in scientific applications. In SciDB [Paradigm4 2017] for instance, due to some representation constraints, it might be necessary to preload multidimensional data into an unidimensional array and then rearrange it during data loading. RasDaMan requires either the creation of a script or the generation of compatible file formats for data ingestion. The result is an inefficient loading process in both cases.

Scientists do not necessarily want to persist all data for a long period of time, instead they want to analyze huge dataset as switfly as possible. Therefore, adopting a DBMS which imposes high overhead for data ingestion does not make sense, even an array DBMS that offers a data representation adequate for scientific data, because the possible

convenience of using a declarative query language does not compensate the trouble to ingest data into the system. In this section we present a briefly overview of the TARS model as presented in [Lustosa et al. 2017].

2.1. Model Overview

Given the lack of flexibility in current array DBMSs, we propose the TARS data model to cope with the aforementioned issues. A TAR Schema (TARS) contains a set of Typed ARrays (TARs). A TAR has a set of dimensions and attributes. A TAR cell is a tuple of attributes accessed via a set of indexes. These indexes define the cell location within the TAR. A TAR has a type, formed by a set of roles. A role in a type defines a special purpose data element with specific semantics.

In TARS (Figure 1), we define mapping functions as a way to provide support for sparse arrays, non-integer dimensions and heterogeneous memory layouts. With TARS, it is possible to combine array data from different sources, different storage layouts, and even with different deegres of sparsity by associating different mapping functions to different subarrays, or as we call them, subTARs.

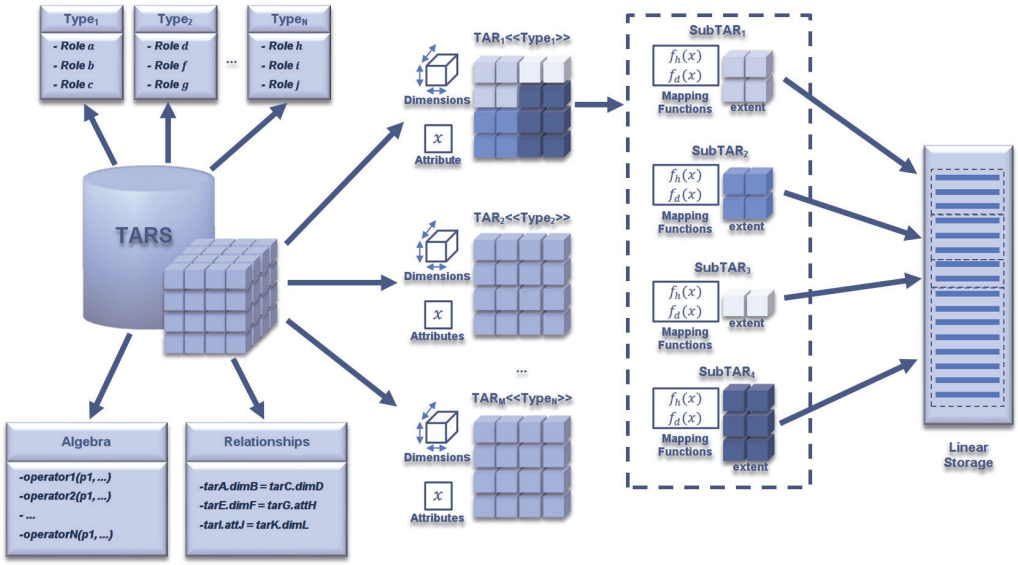


Figure 1. Typed Array Schema Elements

A subTAR covers a n-dimensional slice of a TAR. Every subTAR is defined by the TAR region it represents and two mapping functions: position mapping function and data mapping function. The position mapping function reflects the actual data layout, since it defines where every TAR cell within a given subTAR ended in linear storage. Therefore, the position mapping function should implement the multidimensional linearization technique used for the data. The data mapping functions translate a linear address into data values. In a simple scenario, this function does basically a lookup into a linear array that stores the data. In a more complex scenario, it could compute a derived value from the actual subTAR data.

2.2. Physical Specification

In this section, we describe how the TARS data model is implemented in SAVIME. TARS structures are created in SAVIME with the use of the supported SDL and DDL. Users can

define TARs, datasets, and types. Once a TAR is defined and a series of datasets are loaded into the system, it is possible to specify a subTAR by attaching datasets to it. A dataset is a collection of data values of the same type, like a column in a column-store DBMS (SAVIME uses vertical partitioning). A dataset can contain data for a TAR attribute within a TAR region specified by a subTAR.

TAR dimensions indexes form a domain of values that are represented in SAVIME in two main forms. It can be an implicitly defined range of equally spaced values, in which case, all the user must specify is the lower and upper bounds, and the spacing between two adjacent values. It is called implicit because these indexes do not need to be explicitly stored. For instance, the domain $D_i = (0.0, , 2.0 , 4.0 , 6.0 , 8.0 , 10.0)$ is defined by the lower bound 0.0, the upper bound 10.0 and all values are equally spaced in 2.0 units.

Dimensions whose indexes do not conform with these constraints have an explicit definition. In this case, the user provides a dataset specifying the dimension indexes. For instance, consider the following domain $D_e = (1.2, , 2.3 , 4.7 , 7.9 , 13.2)$. It has a series of values that are not well-behaved and equally spaced, and thus, can not be represented implicitly.

The data representation within the subTAR requires the combination between the dimension domain and the dimension specifications. All subTARs in a TAR have a list of dimension specifications, one for each dimension in the TAR. These dimension specifications define the TAR region the subTAR encompasses, but they also are a fundamental part in the implementation of the mapping functions. These functions are defined conceptually in the model, but are implemented considering six possible configurations between dimension specifications (ORDERED, PARTIAL and TOTAL) types and dimension types (IMPLICIT and EXPLICIT).

An ORDERED dimension specification indicates that the indexes for the cells in that dimension are dense and sorted in some fashion. A PARTIAL dimension implementation, indicates that there are some holes in the datasets, meaning that some cells at given indexes are not present. Finally the TOTAL representation indicates that data is fully sparse and that all indexes must be given for every cell, in other words, it means that we have a degenerated array that is basically tabular data.

3. The system SAVIME

SAVIME has a component-based architecture common to other DBMSs, containing modules such as an optimizer, a parser and an query processing engine, along with auxiliary modules to manage connections, metadata and storage. A SAVIME client communicates with the SAVIME server by using a simple protocol that allows both ends to exchange messages, queries and datasets. All modules are currently implemented as a series of C++ classes, each one of them with an abstract class interface and an underlying concrete implementation.

3.1. Languages DDL, SDL and DML

SAVIME's DDL supports operators to define TARS and Datasets, for instance, the commands:

```
CREATE_TAR("FooTAR", "*", "Implicit, I, long, 1, 1000, 1
                         | Implicit, J, long, 1, 1000 , 1",
```

```
                              "attrib, double");
CREATE_DATASET("FooBarDS1:double", "ds1_data_source");
CREATE_DATASET("FooBarDS2:double", "ds2_data_source");
LOAD_SUBTAR("FooTAR", "Ordered, I, 1, 100 | Ordered, J, 1, 100",
            "attrib, FooBarDS1");
LOAD_SUBTAR("FooTAR", "Ordered, J, 101, 200 | Ordered, I, 1, 100",
            "attrib, FooBarDS2");
```

Initially we can issue a CREATE_TAR command to create a TAR name FooTAR. It has 2 dimension (I and J) whose indexes are long integers. These are implicit dimensions, whose domains are integers equally spaced by 1 unit from 1 to 1000. This TAR also has a single attribute named attrib whose type is a double precision real number.

After that we create 2 datasets named FooBarDS1 and FooBarDS2, they are double typed collections of values in a data source (usually a file or memory based file). Finally we issue 2 LOAD_SUBTAR commands to create 2 new subtars for the TAR FooTAR, the first one ecompasses the region that contains the cell s whose indexes are in $[1, 100] \times [1, 100]$ for dimension I and J respectively, in both case we have an ordered representation indicating that data is dense and ordered first by the I index and second by J index. The second subtar, however, encompasses the cells whose indexes are in $[1, 100] \times [101, 200]$ but instead ordered first by J index and second by the I index. It is an example of how the SDL works, since users can express and consolidate data sources with different ordering layouts into a single TAR. The final part of the command indicates that datasets FooBarDS1 and FooBarDS2 are attached to the ”attrib” attribute, meaning that the data for ”attrib” in the cells whithin each subTAR region can be found in these datasets.

SAVIME also supports a functional DML with operators similar to the ones implemented in SciDB, for operations such as filtering data based on predicates, calculating derived values, joins and aggregations. Here is an example of a query in SAVIME.

```
AGGREGATE(
          WHERE(
                 DERIVE(FooTAR, attrib2, attrib*attrib),
                 attrib2 >= 2.0 and attrib2 <= 10.0
                ),
           sum, attrib2, sum_attrib2, I
);
```

This DML query consists of three nested operators. Initially, a new attribute called attrib2 is created by the operator DERIVE and its value is defined as the square of the attribute attrib. After that, the WHERE operator is called, it filters data according to the predicate, in this case, it returns a TAR whose cells have the value for attrib2 set between 2 and 10. Finally, we use the AGGREGATE operator to group data by dimension I indexes and sum the value for attrib2 creating the sum_attrib2, the resulting TAR will present only one dimension (I) and a single attribute sum_attrib2 whose values are the result of the sum of the attrib2 accross dimension J.

3.2. Query Processing

As presented in the previous section, a SAVIME query contains a series of nested operators. Most of them expect one or more input TARs, and originate a newly created output TAR.

Unless a special operator is called to materialize the query resulting TAR, it is generated as a stream of subTARs, sent to the client and then discarded. SAVIME operates TARs as a subTARs stream pipelined across operators. SubTARs are processed serially or in parallel with OpenMP constructs.

During query processing, when a subTAR for a TAR holding intermediated results is generated and passed on to the next operator, it is maintained in a temporary subTARs cache. These subTARs contain their own group of datasets that could require a lot of storage space or memory. Therefore, once a subTAR is no longer required by any operator, it must be removed from memory. An operator implementation is agnostic regarding its previous and posterior operations in the pipeline, and does not know when to free or not a subTAR. All the operators implementation needs to establish is when it will not require a given subTAR any longer. When this happens, the operator notifies the execution engine that it is done with a given subTAR and it is then discarded.

However, since the same subTAR can potentially be input into more than one operator during a query, freeing it upfront is not a good idea, because it might be required again. In this case, SAVIME would have to recreate it. To solve this problem, every subTAR has an associated counter initially set to the number of operators that have its TAR as their input. When an operator notifies the engine that it no longer needs that specific subTAR, the respective counter is decreased. Once the counter reaches zero, all operators possibly interested in the subTAR are done, and now it is safe to free the subTAR. This approach always frees the used memory as soon as possible and never requires a subTAR to be created twice. However, some operators might require many subTARs to be kept in memory before freeing them. In an environment with limited memory, it would not be feasible to cope with very large TARs in this case. A solution then, would be the adoption of a more economical approach, trading off space with time by freeing and regenerating the subTARs whenever memory is running low.

4. Experimental Evaluation

We ran a series of experiments in order to validate SAVIME as a feasible alternative to simulation data management. We compare SAVIME with SciDB and evaluate how SAVIME affects the performance of actual simulation code.

4.1. SAVIME vs. SciDB

In this section, we compare SAVIME, SciDB (version 16.9) and a third approach based on the usage of NetCDF files (version 4.0), used as a baseline. All scripts, applications and queries used in our evaluation are available at `github.com/hllustosa/savime-testing`.

We use two datasets, a dense and a sparse one, based on data from the HPC4e BSC seismic benchmark [Center 2016] in our experiments. The dense dataset contains 500 trials (one dataset for each) for a 3D regular mesh with dimensions 201x501x501 containing a velocity field. In total, we have over 30 billion array cells and more than a 120 GB of data. All data is held in a single 4D structure (TAR in SAVIME, array in SciDB and in a single NetCDF file) containing the X, Y, and Z dimensions, and an extra trial dimension to represent all 500 trials. Both structures have the same tiling configuration, i.e., the same number of chunks/subTARs (500 of them, one for each trial) with the same extents.

The sparse dataset is also a 4D structure with 500 simulation trials, but only a subset of the cells are present (around 24% of the dense dataset). It comprises almost 8 billion array cells and over 30 GB of data. We used a sparse 4D array/TAR in SciDB and SAVIME, and a 2D dense array in NetCDF. NetCDF lacks the ability to natively represent sparse data, thus we indexed the x, y and z values and represented them as a single dimension and stored coordinate values as variables.

The computational resource used is the fatnode from the cluster Petrus at DEXLab. This fatnode has 6 Intel(R) Xeon(R) CPU E5-2690 processors amounting to 48 cores and over 700 GB of RAM. Data is kept in a shared-memory file system to simulate an in-transit data analysis, in which data is not kept on disk (for both SAVIME and SciDB). Initially, we evaluate the loading time of 500 tiles/chunks in all three approaches, considering that data is being transferred and continually appended to a single array/TAR/file as it is being generated by a solver.

As we can see in Figure 2 on the left graph, the ingestion time taken by SciDB is almost 20 times longer than the time taken by SAVIME, due to costly rearrangements needed on data to make it conform with the underlying storage configuration. Besides, there is an extra overhead during the lightweight data compression done by SciDB, which makes the dataset roughly 50% smaller when stored but increased loading time prohibitively. In contrast, SAVIME does not alter or index the data during the process of data ingestion, therefore the loading process is computationally much cheaper.

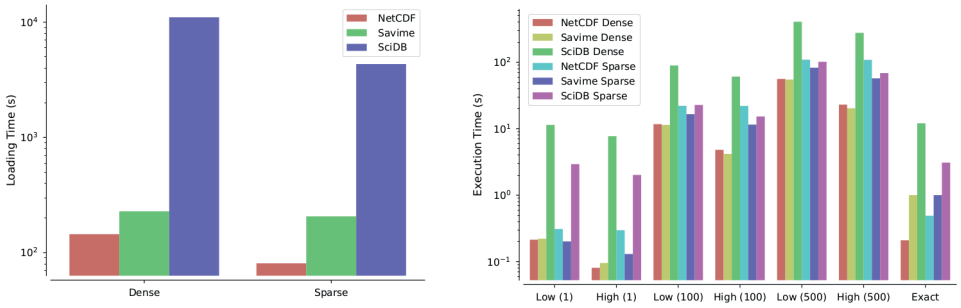


Figure 2. Ingestion and query execution time

We evaluate the performance considering ordinary and exact window queries. Ordinary Window queries consist of retrieving a subset of the array defined by a range in all its dimensions. The performance for this type of queries depends on how data is chunked and laid out. High selectivity queries, which need to retrieve only a very small portion of the data tends to be faster than full scans. Therefore, we compared low and high selectivity queries, filtering from a single to all 500 tiles. We also considered the performance of exact window queries, which is the easiest type of Window Query. They consist of retrieving data for a single tile or chunk, meaning the system has close to zero work filtering out the result.

We implement these queries as specific operators in SAVIME and SciDB, and with the help with a custom OpenMP application using the NetCDF library. The experimental results are shown in Figure 2 on the right graph. The average time of 30 runs are presented.

We considered window queries with low selectivity (over 70 % of all cells in a tile) and high selectivity (around 20 % of all cells in a tile), and intersecting with only 1, 100 or even the total 500 tiles.

It is noticeable that SAVIME either outperforms SciDB or is as efficient as it in all scenarios. The most important observation is that, even without any previous data preprocessing, SAVIME is able to simply take advantage of the existing data structure to answer the queries efficiently, which validates the model as a feasible alternative to existing implementations. The results show that, for any storage alternative in both dense and sparse formats, the subsetting of a single tile is very efficient. The differences shown for the exact window query and for low and high selectivity window queries that touch a single tile are very small. SciDB takes a few seconds in most cases, while SAVIME takes in average 1 second. NetCDF is the most efficient in this scenario, retrieving desired data in less than a second.

However, for queries touching 100 or 500 chunks, we can see the differences between querying dense and the sparse arrays. The dense dataset is queried more efficiently, since it is possible to determine the exact position of every cell and read only the data of interest. It is not possible for sparse data, since one is not able to infer cell positions within the tiles. In this case, every single cell within all tiles that intersect with the range query must be checked.

In dense arrays, we can observe a reduced time for retrieving data in high selectivity queries in comparison with low selectivity queries. The execution time of window queries should depend only on the amount of data of interest, since cells can be accessed directly and thus, no extra cells need to be checked. The execution times considering 100 or 500 tiles in SAVIME and NetCDF are in accordance with this premise. However, SciDB shows poorer performance, being up to 8 times slower. It is very likely that SciDB needs to process cells outside of the window of interest depending on the compression technique and the storage layout adopted. SciDB seems to be more sensible to tiling granularity, requiring fine-grained tiles that match the window query to have a performance similar to the NetCDF approach.

There is not much to be done for querying sparse arrays except for going through every cell in the tiles intersecting the window specified by the query. The query time for sparse data in all alternatives show very similar performance. The main difference is that for achieving this result with NetCDF, an OpenMP application needed to be written, while the same result could be obtained with a one-line query in SAVIME and SciDB.

Our conclusion is that the regular chunking scheme imposed by SciDB not only slows down the ingestion process significanly as it has not real impact in improving performance for simples operations, since SAVIME using a more flexible data model can solve similtar queries presenting a compatible performance.

4.2. Integration with Numerical Solver

In this section, we evaluate the amount of overhead imposed to the simulation code when integrating with SAVIME. We use the simulation tools based on the MHM numerical method [Gomes et al. 2017] as a representative numerical simulation application. We compare three approaches. In the first approach, SAVIME is used IN-TRANSIT, in a single node (fatnode) while the simulation code runs in a different set of nodes, and thus

data needs to be transferred. In the second approach, SAVIME is used IN-SITU, with individual SAVIME instances running on each node, the same used by the simulation code. In this scenario, the data does not need to be transferred, since it is maintained in a local SAVIME instance that shares the same computational resources used by the simulation code. In the third approach SAVIME is not used, but instead, the data is stored in ENSIGHT files (the standard file format used by MHM), and analysis are performed by an ad-hoc Message Passing Interface application in Python. This last scenario serves as a baseline implementation, thus we call it the baseline approach. The computational resource used is the Petrus cluster at DEXLab, with 8 nodes, each with 96 GB of RAM and 2 Intel(R) Xeon(R) CPU E5-2690 processors.

Preliminarily, we start the evaluation by measuring the overhead of loading data in a remote node running SAVIME. In Figure 3 we can see the time of running the simulation and discarding the data, which is the time to solely carry out the computations without any I/O whatsoever and the time to transfer and load data into SAVIME. We vary the size of the MHM meshes, which impact on the level of detail of the simulation. The larger the mesh size is, the more realistic and complex the simulation is, and also, more resources (time and memory) are consumed.

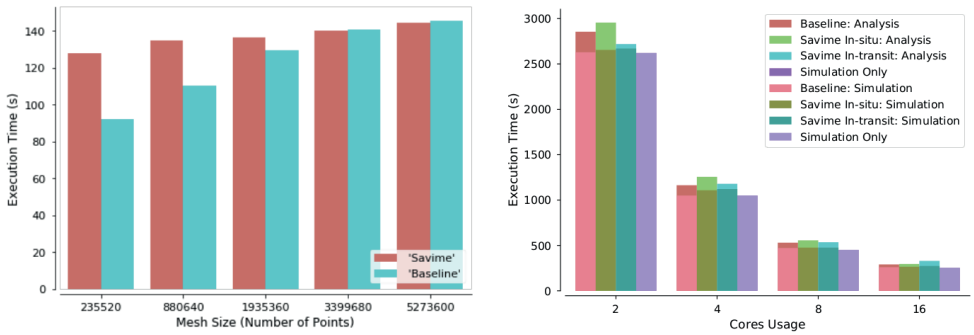


Figure 3. Mesh size x Simulation Execution Time (left) and Simulation and Analysis Scalability (right)

Results show that for very small meshes, which are computationally cheap, the time to load data is more significant, and thus there is some overhead (around 40%) in storing data in SAVIME. However, as meshes get larger, the time taken to compute them increases in a manner that the transfer and loading time is negligible in comparison to the time taken to compute the solution. The transfer and loading time is masked by the dominant process of solving the systems of equation. Therefore, this shows that, for large enough problems, it is possible to load data into SAVIME without compromising the simulation code performance. However, under an impedance mismatch scenario, as observed with SciDB, the loading time would be significant, impacting on the simulation cost.

To evaluate the integration between SAVIME and a simulation tool based on the MHM solver, we use a 2D transport problem over a mesh with 1.9 million points. We run the simulation up to the 100th time step, and store either in SAVIME (approaches 1 and 2) or in an ENSIGHT file (approach 3), data from 50% of all the computed time steps. In both cases, data is always kept in a memory based file system, and never stored on disk.

Once data has been generated, it is analyzed with a PARAVIEW pipeline that carries out the computation of the gradient of the displacement field of the solution. This part is either done by a special operator in SAVIME, or by an AD-HOC MPI Python application using the Catalyst library (baseline), depending on the approach being run. Additionally, we measure the cost of running the simulation without any further analysis, to highlight the *simulation only* cost.

Figure 3 shows the results when running the three approaches, varying the amount of MPI processes spawned or the number of cores used by the MHM simulation code. In this experiment, the simulation code runs and produces its results and then, the simulation output data is read and processed in the analysis step. The plot shows, for each evaluated number of MPI processes, the simulation time and the analysis time as stacked bars. The graph shows that the cost of the analysis process is significantly smaller than the cost for computing the simulation. Moreover, as the *Simulation Only* run shows, the overhead introduced by storing the data in SAVIME or as an ENSIGHT file is negligible, which confirms the claim that SAVIME can be introduced into the simulation process without incurring in extra overhead. From the point of view of the effect of SAVIME on simulation scalability, the storage of data in SAVIME does not impair the capability of the simulation code to scale up to 16 cores.

The IN-TRANSIT approach differs from the other two approaches since it uses a separate computational resource to execute the analysis step. As we see in Figure 3, even when we increase the number of cores the simulation code uses, the analysis time does not change, because the analysis step is done in the fatnode, and always uses the same number of cores (16) independently from the actual number of cores used by the simulation code. The IN-TRANSIT approach illustrates a scenario in which all data is sent to a single computational node and kept in a single SAVIME instance. This approach offers some extra overhead and contention, since all data is sent to a single SAVIME instance, but this enables posterior analysis that transverse the entire dataset without requiring further data transfers.

The SAVIME IN-SITU approach uses the same computational resources used by the simulation code. When we increase the number of cores used by the simulation code, we also increase the numbers of cores used by SAVIME for analysis. The same is true for the baseline approach, meaning that the AD-HOC application also uses the same number of cores the simulation code uses. Even though the SAVIME IN-SITU approach is slightly slower than the baseline approach, we see that both are able to scale similarly. The difference observed in performance between using SAVIME and coding a specialized application becomes less significant as we increase the number of cores being used during the analysis phase. Nevertheless, the small performance loss in this case might be justified by the convenience of using a query language to express analysis instead of the extensive and error prone process of coding other applications to execute analysis.

5. Related Work

The definition of the first array data models and query languages dates back to the works of Baumann [Baumann 1994] and Marathe [Marathe and Salem 1997] [Marathe and Salem 1999]. Since that time, a myriad of systems emerged in order to allow for the storage and analysis of data over multidimensional arrays. Many array DBMSs

have been proposed, the most prominent ones are RasDaMan [Baumann et al. 1997] and, more rencently, SciDB [Cudre-Mauroux et al. 2009].

Due to the fact that ingesting data into these systems is not an easy task, other arrays systems, such as ArrayBridge [Xing et al. 2017] and ChronoDB [Zalipynis 2018] have been developed. ArrayBridge works over SciDB, and gives it the ability to work directly with HDF5 files. ChronoDB works over many file formats in the context of raster geospatial datasets. SAVIME has the similar goal to ease data ingestion, however it does so by enabling seamless data ingestion considering many different array data source by supporting a SDL and a flexible data model, which makes it different from ArrayBridge. SAVIME also offers its own DML, while ChronoDB makes use of existing applications to process data. SAVIME is also different from TileDB [Papadopoulos et al. 2016], which is not exactly a DBMS with a declarative query language, but a library that deals with array data. In addition, SAVIME is a specialized in-memory solution, while the rest of these systems are usually more disk oriented solutions.

6. Conclusion

The adoption of scientific file formats and I/O libraries rather than DBMSs for scientific data analysis is due to a series of problem concerning data representation and data ingestion in current solutions. To mitigate these problems, and to also offer the benefits of declarative array processing in memory, we propose a system called SAVIME. We showed how SAVIME, by implementing the TARS data model, does not impose the huge overhead present in current database solutions for data ingestion, while also being able to take advantage of preexisting data layouts to answer queries efficiently.

We compared SAVIME with SciDB and a baseline approach using the NetCDF platform. The experimental results show that SciDB suffers from the aforementioned problems, not being an ideal alternative and that SAVIME enables faster data ingestion, while maintaining similar performance during window queries execution. We showed that SAVIME can also match the performance of NetCDF for loading and querying dense arrays while providing the benefits of a query language processing layer.

We also assess SAVIME's performance when integrating with simulation code. In this evaluation, we showed that storing data in SAVIME does not impair the scalability of the solver. In addition, results also show that it is possible to retrieve SAVIME data and generate viz files efficiently by using the special purpose visualization operator.

SAVIME is available at github.com/hllustosa/Savime. Future work might focus on the improvement and optimization of current operators, the development of new special purpose TAR operators.

References

Ahrens, J. (2015). Increasing scientific data insights about exascale class simulations under power and storage constraints. *IEEE Computer Graphics and Applications*, 35(2):8–11.

Baumann, P. (1994). Management of multidimensional discrete data. *The VLDB Journal*, 3(4):401–444.

Baumann, P., Furtado, P., Ritsch, R., and Widmann, N. (1997). The rasdaman approach to multidimensional database management. In *Proceedings of the 1997 ACM Symposium on Applied Computing*, SAC '97, pages 166–173, 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.

Center, B. S. (2016). New hpc4e seismic test suite to increase the pace of development of new modelling and imaging technologies. [Online; accessed 01-feb-2018].

Cudre-Mauroux, P., Kimura, H., Lim, K.-T., Rogers, J., Simakov, R., Soroush, E., Velikhov, P., Wang, D. L., Balazinska, M., Becla, J., DeWitt, D., Heath, B., Maier, D., Madden, S., Patel, J., Stonebraker, M., and Zdonik, S. (2009). A demonstration of scidb: A science-oriented dbms. *Proc. VLDB Endow.*, 2(2):1534–1537.

Gomes, A. T. A., Pereira, W. S., Valentin, F., and Paredes, D. (2017). On the implementation of a scalable simulator for multiscale hybrid-mixed methods. *CoRR*, abs/1703.10435.

Gosink, L., Shalf, J., Stockinger, K., Wu, K., and Bethel, W. (2006). Hdf5-fastquery: Accelerating complex queries on hdf datasets using fast bitmap indices. SSDBM '06, pages 149–158, Washington, DC, USA. IEEE Computer Society.

Group, T. H. (2017). Hdf5 - the hdf group. [Online; accessed 01-feb-2018].

Lustosa, H., Lemus, N., Porto, F., and Valduriez, P. (2017). TARS: An Array Model with Rich Semantics for Multidimensional Data. In *ER FORUM 2017: Conceptual Modeling : Research In Progress*, Valencia, Spain.

Marathe, A. P. and Salem, K. (1997). A language for manipulating arrays. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 46–55, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Marathe, A. P. and Salem, K. (1999). Query processing techniques for arrays. In *ACM SIGMOD Record*, volume 28, pages 323–334. ACM.

Oldfield, R. A., Moreland, K., Fabian, N., and Rogers, D. (2014). Evaluation of methods to integrate analysis into a large-scale shock shock physics code. In *Proceedings of the 28th ACM International Conference on Supercomputing*, ICS '14, pages 83–92, New York, NY, USA. ACM.

Papadopoulos, S., Datta, K., Madden, S., and Mattson, T. (2016). The tiledb array data storage manager. *Proc. VLDB Endow.*, 10(4):349–360.

Paradigm4 (2017). Scidb. [Online; accessed 01-feb-2018].

Unidata (2017). netcdf. [Online; accessed 01-feb-2018].

Xing, H., Floratos, S., Blanas, S., Byna, S., Prabhat, Wu, K., and Brown, P. (2017). ArrayBridge: Interweaving declarative array processing with high-performance computing. *arXiv e-prints*, page arXiv:1702.08327.

Zalipynis, R. A. R. (2018). Chronosdb: Distributed, file based, geospatial array dbms. *Proc. VLDB Endow.*, 11(10):1247–1261.

ETERNAL: Uma estratégia eficiente de tolerância a falhas utilizando memória não volátil

Davi B. Gomes[1], **Angelo Brayner**[1], **Javam C. Machado**[1]

[1]Universidade Federal do Ceará
CEP 60440-900 – Fortaleza – CE – Brazil

davi.braga@lsbd.ufc.br, brayner@dc.ufc.br, javam.machado@lsbd.ufc.br

Abstract. *In-memory DBMSs have proven to be an efficient alternative for managing large volumes of data. They're characterized by using RAM as their primary storage medium. However, such systems need nonvolatile storage to ensure the durability of its transactions. Byte addressable non-volatile memories (NVRAM) are ideal candidates for securing such a property because they have access times close to traditional RAM memory but still guarantee data persistence. In order to address the problem of transaction durability in main memory DBMSs, this paper proposes ETERNAL, a durability architecture that makes efficient use of nonvolatile memory, improving the process performance of persistence in such DBMSs. Experiments reveal that ETERNAL provides a higher throughput than the write-ahead logging (WAL) approach. It is also noteworthy that, even using nonvolatile memory as a storage medium, the WAL approach does not have a mechanism to deal with the scenario in which nonvolatile memory runs out.*

Resumo. *SGBDs em memória principal têm se mostrado como alternativa eficiente para o gerenciamento de grandes volumes de dados. Eles caracterizam-se por utilizar memória RAM como seu meio de armazenamento primário. No entanto, tais sistemas precisam de armazenamento não-volátil para garantir a durabilidade de suas transações. Memórias não-voláteis endereçáveis por byte (NVRAM) são candidatas ideais para assegurar tal propriedade, pois possuem tempo de acesso próximo ao de memória RAM tradicional, mas ainda assim garantem a persistência dos dados. A fim de atacar o problema da durabilidade de transações em SGBDs em memória, este trabalho propõe ETERNAL, uma arquitetura de durabilidade que faz o uso eficiente de memória não-volátil, melhorando o desempenho do processo de persistência em tais SGBDs. Os experimentos revelam que ETERNAL provê um throughput superior a abordagem de escrita antecipada em log (WAL). Destaca-se ainda o fato, que, mesmo utilizando memória não-volátil como meio de armazenamento, a abordagem WAL não possui um mecanismo para lidar com o cenário em que a memória não volátil se esgota.*

1. Introdução

O uso de sistemas de bancos de dados em memória principal (SBDMP) tem crescido significativamente. Esse fato ocorre principalmente em cenários cujas aplicações requerem alto desempenho, no que concerne a altas taxas de *throughput* e a baixo tempo de resposta

de consultas. SBDMPs caracterizam-se pelo fato que seus dados tornam-se persistentes, quando o gerenciador de recuperação descarrega para memória não-volátil uma imagem do banco de dados em memória ou registros de log. Assim, ocorrendo uma queda do sistema, todos os dados, que estavam em memória volátil, podem ser recuperados.

Para garantir a durabilidade de transações confirmadas, a técnica mais utilizada em sistemas de banco de dados convencionais é a gravação antecipada de log (*write-ahead logging* – WAL) na qual qualquer transação só poderá ser confirmada após todas as suas alterações terem sido gravadas antecipadamente em log. Tal técnica, no entanto, pode introduzir uma significante sobrecarga no desempenho de sistemas de banco de dados em memória [Faerber et al. 2017], pois, apesar das transações serem executadas rapidamente em memória principal, precisam necessariamente passar pela etapa de confirmação e escrita de registros no arquivo de log residente em memória não-volátil. Como se sabe, os dispositivos atuais de memória não-volátil, como discos rígidos (HDs), apresentam altos tempos de latência de acesso. Desta forma, para minimizar custo dessa etapa, um possível candidato para o armazenamento persistente do log é a nova classe de memórias não-voláteis endereçáveis por byte.

As memórias não-voláteis endereçáveis por byte (NVRAM) representam uma alternativa para incrementar significativamente o desempenho de sistemas que demandam persistência. Dentre as suas principais características, destacam-se a baixa latência de acesso, bem próxima de uma memória RAM convencional, e o baixo consumo energético, não sendo necessária uma corrente elétrica permanente para garantir a persistência dos dados [Arulraj and Pavlo 2017]. Um dos principais impedimentos ao seu uso generalizado, no entanto, é seu maior custo quando comparado aos discos tradicionais, o que exige um uso eficiente.

Cargas de trabalho OLTP possuem um padrão de acesso *skewed* com relação à idade e ao uso do dado. Tipicamente, apenas alguns registros "quentes"são escritos mais frequentemente, porém, com o passar do tempo, ocorre diminuição da frequência de escrita e tais registros tornam-se "frios", passando a ter menor probabilidade de serem escritos no futuro [Eldawy et al. 2014], eventualmente recebendo alguma leitura. Em termos de desempenho, é interessante que tais registros quentes permaneçam em meios de armazenamento de acesso mais rápidos. Algumas abordagens que procuram diminuir o custo do meio armazenamento levando em conta o *skew* de cargas OLTP em SGBDs de memória principal já foram propostas anteriormente [DeBrabant et al. 2013] [Amora et al. 2018]. Nenhuma, no entanto, tentou aproveitar essa característica no mecanismo de recuperação do sistema de banco de dados, o componente onde encontra-se o gargalo relacionado ao custo adicional da técnica WAL, para armazenamento de registros de log.

Este trabalho propõe ETERNAL, um mecanismo de recuperação que explora as propriedades de armazenamento híbrido para garantir durabilidade de transações confirmadas. Para tanto, o ETERNAL garante que as transações são primeiramente confirmadas e persistidas em um armazenamento intermediário persistente de NVRAM. ETERNAL usa a baixa latência e endereçabilidade por byte da NVRAM para atualizar individualmente a última imagem escrita de um dado. Assim, há uma redução no número de registros de log para várias atualizações sobre um mesmo dado. Adicionalmente, ETERNAL permite o despejo assíncrono do conteúdo da NVRAM para HDD, fazendo com que

a etapa de confirmação das transações não fique diretamente dependente da latência de acesso ao disco.

Este artigo está assim estruturado. Inicialmente, os trabalhos relacionados são apresentados e discutidos na sessão 2. Na sessão 3 são descritos a arquitetura de ETERNAL e os protocolos de confirmação, de despejo em memória não-volátil e de recuperação. Na sessão 4 são discutidos os experimentos e na sessão 5 tem-se a conclusão e propostas de trabalhos futuros.

2. Trabalhos Relacionados

A abordagem dominante de recuperação de falhas utilizada por sistemas baseados em disco é o protocolo ARIES [Mohan et al. 1992]. Em geral, a propriedade de durabilidade e a recuperação em sistemas de memória principal são baseadas em WAL. No entanto, ao examinar os detalhes, a diferença entre os sistemas baseados em disco e os em memória principal é grande. Muitos sistemas em memória principal evitam usar protocolos no estilo ARIES por razões de desempenho [Faerber et al. 2017].

2.1. Recuperação em Banco de Dados em Memória Principal

O subsistema de *logging* de SGBDs em memória principal é otimizado para alto *throughput* e para baixa latência. Como o *I/O* do log é o maior gargalo, esses sistemas procuram reduzir o volume de logs o máximo possível, principalmente em sistemas baseados em disco. Hekaton, por exemplo, realiza o log do tipo *redo only* gravando a versão mais recente do registro apenas para transações garantidas [Diaconu et al. 2013]. Essa abordagem evita escrever informações de *undo* completamente. H-Store/VoltDB usam uma variante enxuta de log lógico chamada *command logging* [Malviya et al. 2014] que registra somente a solicitação de invocação de transação. O registro de log desse esquema consiste unicamente no nome do procedimento armazenado, parâmetros de entrada e o ID da transação. Para minimizar ainda mais a escrita de dados em logs, vários sistemas também evitam os metadados de índices [Faerber et al. 2017]. Apenas as atualizações para o banco de dados são registradas.

2.2. Memória Não Volátil e Recuperação

[Kimura 2015] e [Arulraj et al. 2016] realizaram trabalhos que fizeram uso das capacidades únicas da memória não volátil para recuperação de falhas, mas assumem que a maior parte do conteúdo persistente do banco caberá em NVRAM, não lidando com cenários de armazenamento híbrido. [Huang et al. 2014] propôs NV-Logging, uma estratégia de log em memória não volátil para SGBDs tradicionais em disco, porém, a liberação do armazenamento NVRAM depente de *checkpoints*. *Checkpoints* em SGBDs tradicionais apenas precisam persistir o estado corrente do *buffer* em memória, pois é assumido que grande parte do banco já se encontra em disco. Já em banco de dados em memória principal, todo o banco está em memória, tornando o processo inviável nesse cenário.

3. ETERNAL

A arquitetura do ETERNAL pode ser vista na Figura 1. O banco de dados se localiza em espaço de armazenamento volátil, materializado pela memória principal, enquanto que o espaço de persistência é composto por NVRAM e HDD. ETERNAL divide a NVRAM

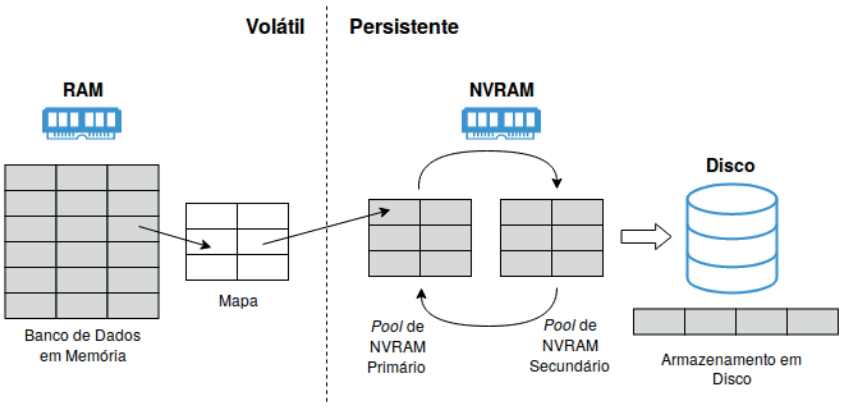


Figura 1. Arquitetura ETERNAL

em *pool* primário e secundário. Há também um mapa ligando registros em memória volátil a endereços de memória não volátil. O mapa indexa as imagens presentes no pool, evitando a necessidade de percorrer a lista encadeada presente em NVRAM no momento da confirmação. Além disso, a latência de acesso da RAM é menor que a da NVRAM, logo, indexar as imagens utilizando um mapa em RAM é mais eficiente. Transações podem realizar dois tipos de operações: leituras e escritas. Inserções, atualizações e deleções são operações de escrita. Para garantir a durabilidade de uma transação, é necessário que as escritas realizadas por ela estejam persistidas em armazenamento durável antes da sinalização de sua confirmação. Em ETERNAL, essa garantia é dada por meio de um mecanismo de lista de imagens.

3.1. Protocolo de Confirmação

Para cada transação em execução, o SGBD manterá uma lista contendo a última imagem de cada registro por ela escrito. Se um registro é escrito mais de uma vez por uma mesma transação, então apenas a sua última imagem desse registro será mantida na lista de imagens. Todo o mecanismo de execução de transações e de preenchimento da lista de imagens é realizado apenas em memória principal. Somente quando uma transação atingir a sua etapa de confirmação, o SGBD garantirá a persistência de sua lista de imagens de maneira atômica. Essa lista será inicialmente persistida no *pool* de NVRAM primário, representado na Figura 1. A transação só poderá ser confirmada após a persistência de sua lista de imagens.

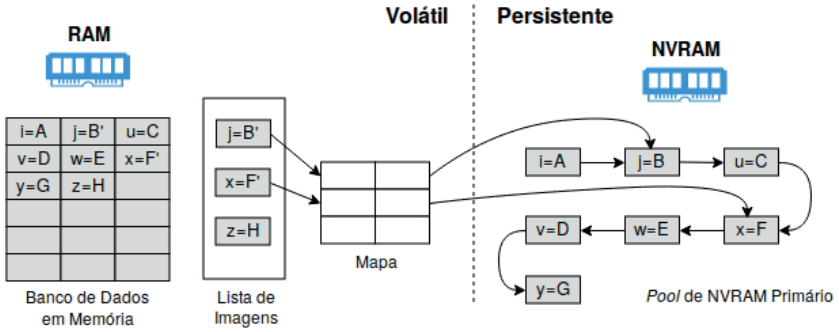


Figura 2. Etapa de confirmação

O processo detalhado de confirmação é ilustrado na Figura 2. Ao entrar na etapa de confirmação de uma transação, o SGBD verificará, para cada registro com imagem presente na lista de imagens, se há ou não uma imagem referente a esse registro presente no *pool* primário de NVRAM. Essa verificação é feita pelo mapa, uma estrutura de dados em memória volátil que mapeia o identificador único do registro à localização de sua imagem no *pool* primário. Caso não seja encontrada uma entrada de imagem do registro, essa imagem será adicionada ao *pool* primário de NVRAM e o mapa em memória volátil será atualizado com essa nova entrada. Caso seja encontrada uma entrada, então a localização da imagem no *pool* de NVRAM é recuperada por meio do mapa e o seu valor será atualizado no local. A etapa de confirmação ocorre de maneira serial, ou seja, apenas uma transação poderá realizá-la por vez.

No exemplo da Figura 2, temos uma transação que atualizou o valor do registro j de B para B', o valor do registro x de F para F' e inseriu uma novo registro z com valor H. A lista de imagens dessa transação, portanto, possui uma imagem para cada registro escrito. Os registros j e x possuem entradas no mapa, indicando que há imagens desses registros no *pool*, já z não possui entrada, indicando que será necessário adicionar uma nova imagem. Observa-se que o *pool* de NVRAM é organizado no formato de uma lista encadeada. A única situação na qual ela precisará ser percorrida é no momento da recuperação após falha. Essa estrutura de dados foi escolhida para que a inserção de novas entradas seja realizada com o menor custo possível, sendo o mapa em memória volátil utilizado para indexar a localização das imagens no *pool*. É possível perceber que o banco de dados em memória principal já possui os dados atualizados com as escritas da transação, mas o *pool* de NVRAM primário ainda possui a imagem dos valores antigos de B e F, além de não possuir uma imagem de H. A etapa de confirmação do exemplo representada na Figura 2, portanto, consiste em atualizar, de forma atômica, as imagens dos registros j e x presentes no *pool* de NVRAM primário, bem como adicionar a imagem de H. O estado do *pool* de NVRAM primário após o término da etapa de confirmação é ilustrado pela Figura 3.

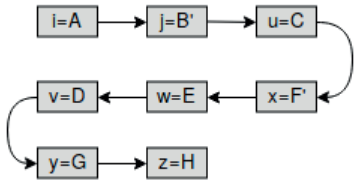


Figura 3. *Pool* de NVRAM após etapa de confirmação

É importante notar que ETERNAL usa NVRAM como meio de persistência dos dados necessários à etapa de confirmação. Isso é uma vantagem, pois significa que o tempo de resposta das transações estará sujeito à latência de acesso da NVRAM, que é próxima à latência de acesso da memória RAM tradicional.

3.2. Uso Eficiente de NVRAM

Um aspecto em que ETERNAL é significativamente melhor que as abordagens baseadas em WAL é com relação ao uso eficiente das memórias do tipo NVRAM, pois nossa abordagem tira proveito da capacidade de endereçamento por byte desse tipo de dispositivo. Em uma típica abordagem baseada em log, novas entradas são concatenadas a calda do

log conforme registros vão sendo escritos por transações. A Figura 4 mostra a evolução de um log no caso em que um registro é inserido inicialmente com valor A, em seguida tem o seu valor atualizado para B e finalmente é atualizado para C.

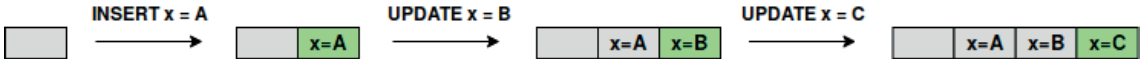


Figura 4. Abordagem baseada em WAL

No exemplo da Figura 4, um registro passa por três operações de escrita, gerando três entradas em log. O caso onde um mesmo registro é escrito repetidas vezes por diversas transações não é incomum, de fato esta é uma característica típica de cargas de trabalho OLTP. Nesse tipo de carga, registros acessados mais recentemente têm maiores chances de serem acessados em um futuro próximo, tais registros também são chamados de "quentes". Eventualmente, com o tempo, a taxa de acesso desses registros cai e eles passam a ser chamados de "frios". [Eldawy et al. 2014]

Já no exemplo da Figura 5, é demonstrado como se dá o uso do armazenamento na abordagem ETERNAL. Como as imagens de registros escritos são armazenadas em NVRAM, um meio endereçável por byte, então é possível indexá-las a nível de registro. Quando um mesmo registro é escrito mais de uma vez, basta atualizar a sua última imagem presente no *pool* primário de NVRAM, pois apenas a última imagem é necessária ao protocolo de recuperação.

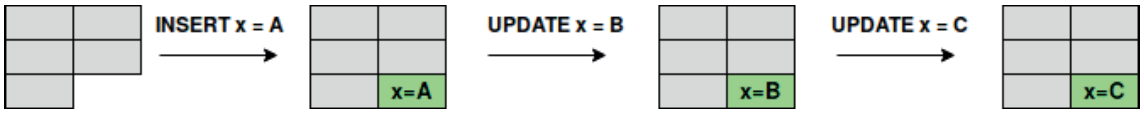


Figura 5. Abordagem ETERNAL

O principal aspecto no qual ETERNAL supera as abordagens baseadas em WAL está relacionado ao comportamento do SGBD no cenário em que o armazenamento NVRAM se esgota. Como demonstrado acima, ETERNAL não precisa manter várias imagens de um mesmo registro que é repetidamente escrito. Essa característica garante que, para uma mesma carga de trabalho OLTP, a situação de esgotamento de NVRAM é menos frequente em ETERNAL que nas abordagens baseadas em WAL, diminuindo drasticamente a necessidade de recorrer a armazenamentos mais lentos. Além disso, ETERNAL implementa um mecanismo assíncrono de despejo de dados de NVRAM para disco.

O uso de NVRAM é imprescindível em ETERNAL, pois, diferente da NVRAM, HDDs e SSDs são endereçáveis por bloco, portanto, os dados devem ser agrupados em páginas, ocasionando overhead adicional em memória principal. Atualizações in place em dispositivos de bloco podem facilmente alcançar diversas páginas, incorrendo no custo do *flush* de várias páginas em um dispositivo que já possui alta latência. Pior ainda, o padrão de acesso a essas páginas não é necessariamente sequencial, penalizando ainda mais o tempo de acesso em tais dispositivos.

3.3. Despejo em Disco

Conforme ilustrado na Figura 1, a arquitetura ETERNAL é composta por dois *pools* de NVRAM. Como já discutido anteriormente, o *pool* de NVRAM primário é utilizado na

persistência da lista de imagens durante a etapa de confirmação. O pool secundário, no entanto, é utilizado como um armazenamento temporário para o despejo assíncrono de dados de imagem para disco. Quando o armazenamento primário é completamente preenchido por imagens de registros, o SGBD realiza uma rotação de *pools*. O *pool* secundário assume o papel de *pool* primário, recebendo as novas cópias de imagens. Já o *pool* primário, que estava completamente preenchido com imagens, assume o lugar do secundário, sendo varrido por uma *thread* auxiliar que despeja dados de imagens em disco. Caso o *pool* primário se esgote enquanto o *pool* secundário ainda estiver realizando um despejo, então as transações serão bloqueadas até o início da próxima rotação.

A *thread* de despejo varre a lista encadeada de imagens presente no *pool* de NVRAM. Cada imagem de registro é copiada para um arquivo sequencial em disco. Antes de iniciar a cópia das imagens, um marcador é inserido no arquivo para indicar o início de um novo despejo, outro marcador será inserido ao término do despejo. O SGDB garante que o *pool* secundário será limpo para uma nova rotação apenas após o término da cópia de todas as imagens de registro para disco, assegurando que todas as imagens presentes no *pool* estarão persistidas em disco antes da limpeza. Na Figura 6, temos uma ilustração de um despejo no qual as imagens dos registros u, v, x e y foram copiadas para disco. É possível ver também os marcadores de início do despejo (ID) e de fim do despejo (FD).

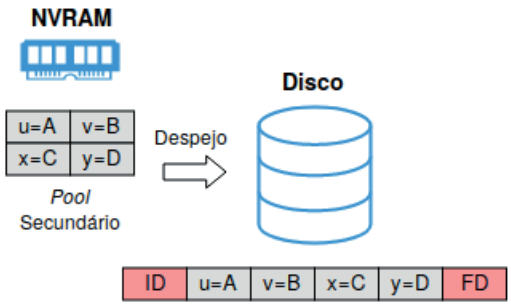


Figura 6. Despejo em disco

A grande vantagem do mecanismo de *pools* rotativos é que a etapa de confirmação das transações não fica diretamente dependente da latência de acesso ao disco, pois a etapa de confirmação é realizada diretamente em NVRAM, tirando proveito da sua menor latência e diminuindo o tempo de resposta das transações.

3.4. Protocolo de Recuperação

Quando ocorre uma queda do SGBD, todo o conteúdo presente em memória não volátil é perdido. No caso específico de SGBDs em memória principal, isso significa que todo o banco deverá ser carregado em memória novamente. Em ETERNAL, o protocolo de recuperação funciona carregando a última imagem transacionalmente consistente de cada registro do banco, não sendo necessário um histórico de atualizações. Como o protocolo de confirmação garante que toda transação deve persistir as imagens de todos registros por ela escritos em armazenamento durável, então existem três localizações possíveis para as imagens: o *pool* de NVRAM primário, o *pool* de NVRAM secundário e o arquivo sequencial em disco. O processo de recuperação consiste em percorrer todos essas localizações, identificar a imagem mais atual de cada registro e carregá-la no banco.

O protocolo de recuperação deve levar em conta que é possível existir mais de uma versão de imagem de um mesmo registro nos diferentes meios de armazenamento. O mapa utilizado no protocolo de confirmação assegura que existe apenas uma imagem de um mesmo registro no *pool* de NVRAM primário. O mesmo vale para o *pool* de NVRAM secundário, mas não para o arquivo sequencial em disco. Mais de uma imagem de um mesmo registro em disco pode ocorrer quando um registro cuja imagem já havia sido despejada é escrito novamente, gerando uma nova imagem no *pool* de NVRAM primário que eventualmente será despejada novamente para disco.

A versão mais nova da imagem de um registro estará sempre no *pool* de NVRAM primário, pois é nele onde as imagens são inicialmente persistidas durante a etapa de confirmação. Em seguida, temos o *pool* secundário, com versões das imagens presentes no momento do despejo. Por último, temos as imagens presentes no arquivo sequencial, já despejadas em disco, com imagens mais antigas no início do arquivo e mais novas no fim. Para garantir que a versão final dos registros após a recuperação será a mais nova, o algoritmo de recuperação inicia o carregamento do banco a partir das imagens mais antigas, as sobrescrevendo sempre que encontrar uma imagem mais nova de um registro já carregado.

Os marcadores de início e de fim de despejo são utilizados para garantir a atomicidade de toda a operação quando uma falha ocorre em meio a um despejo. Se, no momento da recuperação, for detectado um despejo com marcador de início, mas sem um marcador de fim, então significa que esse despejo não foi concluído completamente, possivelmente devido a uma falha. Nesse caso, todas as imagens relativas a esse despejo serão ignoradas. Como o pool secundário só é limpo após a inserção do marcador de de fim de despejo em disco, então temos a garantia de que essas imagens ainda estarão persistidas no pool secundário, assegurando a correta recuperação.

4. Experimentação

Apresenta-se agora a análise experimental comparando o desempenho de ETERNAL com a abordagem estado da arte baseada em WAL e utilizada em [Malviya et al. 2014] para banco de dados em memória principal. Para tornar a comparação mais justa, a abordagem concorrente será adaptada para utilizar NVRAM como meio de armazenamento para seu log. Tal como ocorre em ETERNAL, o log da abordagem WAL será despejado para disco caso a NVRAM seja completamente preenchida. Chamaremos essa abordagem de NVWAL. Os experimentos foram realizados em uma máquina Intel Core i7-7800X com 6 núcleos e 12 threads rodando a uma frequência base de 3.5GHz e 128GB de memória SDRAM DDR4, com sistema operacional Ubuntu 18.04. A NVRAM utilizada no experimento é emulada [Maciejewski 2017].

Assim como acontece em trabalhos que se diferenciam fundamentalmente da maneira como SGBDs tradicionalmente funcionam, ETERNAL e NVWAL foram implementados como *engines standalone* em C++ [Kimura 2015]. Para fins de avaliação, ambos os *engines* de recuperação foram conectados a um banco de dados em memória simples baseado na coleção std::unordered_map da biblioteca padrão de C++. Tanto o banco quanto as *engines* foram compilados em um mesmo binário junto com uma implementação do *benchmark* YSCB em C++. O código fonte está disponível em [Gomes 2019].

4.1. Carga de Trabalho

O *benchmark* usado no experimento é o YCSB [Cooper et al. 2010], utilizado para modelar aplicações OLTP. Como o objetivo do experimento é avaliar o desempenho do sistema de banco em cenários de escrita, não são utilizadas operações de leitura, além disso, cargas de leitura não afetam o subsistema de recuperação. Dois tipos de cargas foram submetidas: carga de atualização, com 100% das operações sendo de atualizações e carga de inserção, com 50% das operações sendo de inserções e 50% sendo de atualizações. O YCSB também permite selecionar a distribuição estatística que modela a frequência de acesso dos registros do banco. As distribuições mais utilizadas para modelar cenários do mundo real são a zipfian e a latest [Cooper et al. 2010]. Em todos os experimentos o banco é inicializado com 12GB de registros. A quantidade de NVRAM emulada é de 2GB, equivalente a aproximadamente 17% do tamanho inicial do banco. Em ETERNAL, cada *pool* de NVRAM tem 1GB. É importante mencionar que a atomicidade das operações realizadas em NVRAM são garantidas pela biblioteca libpmemobj por meio de logs de *undo* e que seu código fonte é aberto [Balcer 2017].

4.2. Análise dos Resultados Experimentais

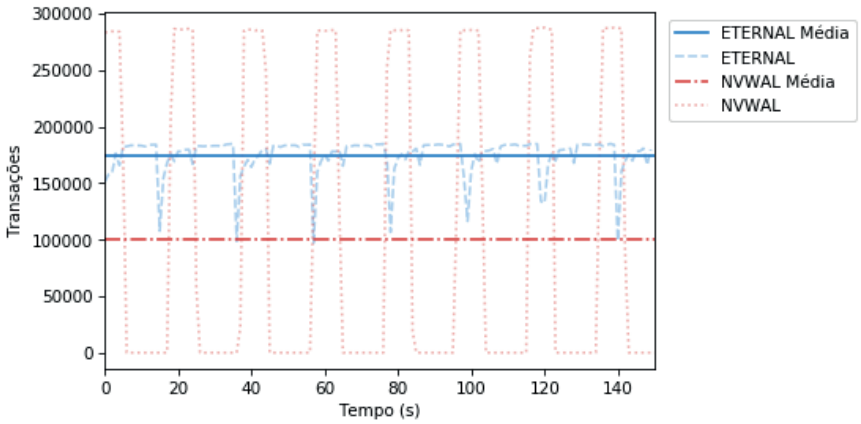


Figura 7. *Throughput*: atualização com distribuição zipfian

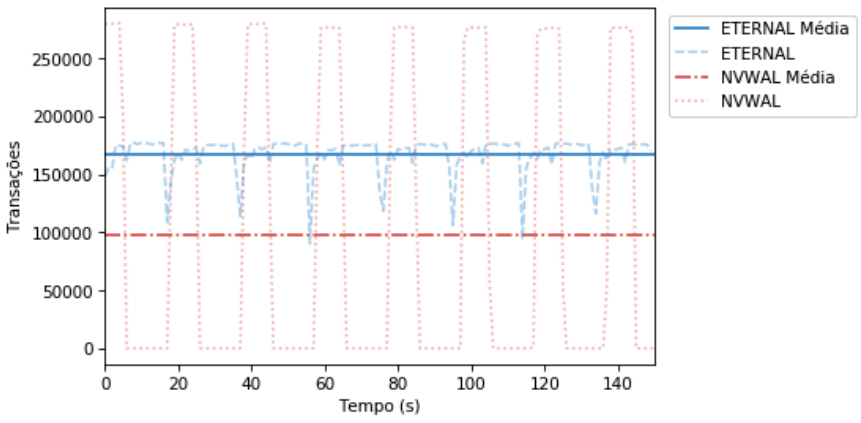


Figura 8. *Throughput*: atualização com distribuição latest

Nas Figuras 7 e 8 são apresentados os gráficos de *throughput* que mostram o número de transações realizadas a cada segundo na carga de atualização utilizando as distribuições zipfian e latest respectivamente. Temos também duas linhas representando o *throughput* médio total do experimento para as duas abordagens. ETERNAL apresentou um *throughput* médio de 174327 TXs/s contra 99792 TXs/s de NVWAL com a distribuição zipfian. Com a distribuição lastest, ETERNAL apresentou um *throughput* médio de 167650 TXs/s contra 97939 TXs/s de NVWAL . O *throughput* total dos experimentos de ETERNAL foi 73% maior quando comparada ao *throughput* de NVWAL.

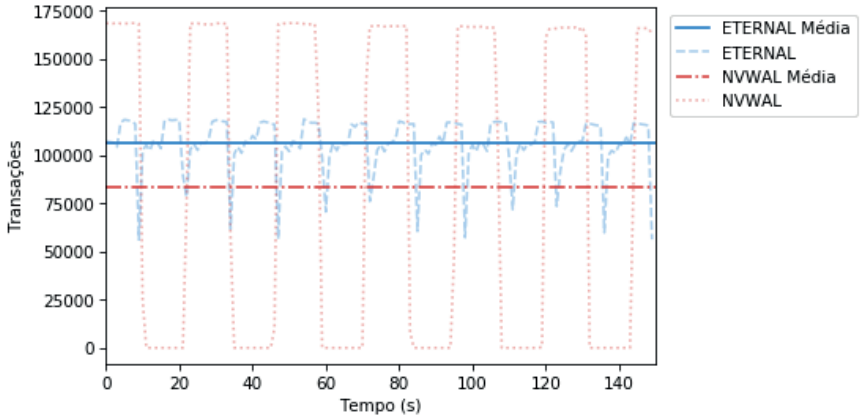


Figura 9. *Throughput*: inserção com distribuição zipfian

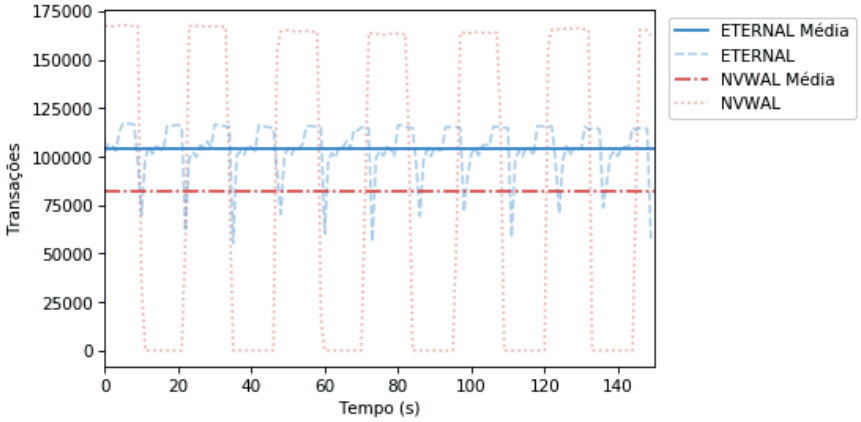


Figura 10. *Throughput*: inserção com distribuição latest

Nas Figuras 9 e 10 são apresentados os gráficos de *throughput* que mostram o número de transações realizadas a cada segundo na carga de inserção utilizando as distribuições zipfian e latest respectivamente. ETERNAL apresentou um *throughput* médio de 106000 TXs/s contra 83579 TXs/s de NVWAL com a distribuição zipfian. Com a distribuição lastest, ETERNAL apresentou um *throughput* médio de 104495 TXs/s contra 82438 TXs/s de NVWAL . A *throughput* total dos experimentos de ETERNAL foi 26% maior quando comparada ao *throughput* de NVWAL.

Analisando os gráficos, é possível perceber que, na média, ETERNAL atinge *throughputs* significativamente maiores que NVWAL. É possível perceber também que

NVWAL apresenta, em alguns momentos, valores de *throughput* maiores que os de ETERNAL. Essa perda momentânea de desempenho pode ser explicada em parte pela sobrecarga adicional causada pela uso do mapa para acessar as imagens no *pool* primário de NVRAM. Outro fator a ser considerado é a sobrecarga gerada pelos logs de *undo* da biblioteca libpmemobj, necessários para garantir a atomicidade das operações realizadas em NVRAM. Ambas distribuições apresentaram resultados semelhantes porque ambas possuem *skew*. O comportamento seria diferente em uma distribuição uniforme. Uma carga uniforme, no entanto, não modela bem padrões de acesso OLTP. O desempenho em cargas com inserções é menor, pois cada inserção representa necessariamente um dado novo e portanto não repetido, aumentando a quantidade de despejos.

A vantagem de ETERNAL se dá no momento em que a NVRAM se esgota, pois, ao fazer o uso eficiente da tecnologia, a abordagem consegue diminuir a quantidade de dados despejados para disco. Por causa desse mecanismo, o *pool* se esgota com menor frequência, permitindo um despejo sem contenção realizado por uma *thread* de fundo. NVWAL despeja, de maneira síncrona, uma quantidade muito maior de dados, o que explica as quedas de *throughput* apresentadas nos gráficos da abordagem. Caso NVWAL utilizasse *pools* de NVRAM, de nada adiantaria, pois ambos os pools seriam preenchidos rapidamente, ocasionando bloqueios de espera pelo despejo. Uma forma de demonstrar a quantidade de acesso a disco evitada por ETERNAL é comparar o tamanho do arquivo sequencial despejado em disco com o tamanho do log gerado por NVWAL. Ao final do experimento, com a carga de atualização, o arquivo sequencial de ETERNAL possuía um tamanho de 6GB, já o log gerado por NVWAL possuía 18GB. Portanto, NVWAL realizou 3 vezes mais acessos a disco que ETERNAL, o que também explica o seu desempenho geral inferior.

5. Conclusão e Trabalhos Futuros

Este trabalho apresentou ETERNAL, uma arquitetura de recuperação que faz o uso eficiente das memórias não voláteis endereçáveis por byte. ETERNAL projeta e implementa um protocolo de confirmação, um protocolo de recuperação e um mecanismo assíncrono de despejo de dados de NVRAM para disco. Quando comparado a outras abordagens baseadas em WAL, ETERNAL demonstra ganhos significativos de *throughput* de até 73% em cargas de trabalho OLTP, como pode ser constatado nos resultados experimentais.

A partir de ETERNAL, muitos trabalhos futuros podem ser realizados. O *pool* secundário é esvaziado durante a realização de um despejo. É possível melhorar esse mecanismo permitindo que imagens de registros mais acessados permaneçam no *pool*, evitando o custo de adicionar novas imagens do mesmo registro nas próximas rotações. Novos experimentos podem ainda ser realizados em outros *benchmarks* como o TPC-C.

Agradecimentos

Esta pesquisa foi parcialmente financiada pela Lenovo, como parte do seu investimento em pesquisa e desenvolvimento de acordo com a Lei de Informática, pela CAPES (Número do Processo: 1697962) e pelo LSBD/UFC.

Referências

Amora, P. R. P., Teixeira, E. M., Praciano, F. D. B. S., and Machado, J. C. (2018). Smartltm: Smart larger-than-memory storage for hybrid database systems. In *XXXIII Sim-*

pósio Brasileiro de Banco de Dados, SBBD 2018, Rio de Janeiro, RJ, Brazil, August 25-26, 2018., pages 13–24.

Arulraj, J. and Pavlo, A. (2017). How to build a non-volatile memory database management system. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1753–1758. ACM.

Arulraj, J., Perron, M., and Pavlo, A. (2016). Write-behind logging. *Proceedings of the VLDB Endowment*, 10(4):337–348.

Balcer, P. (2017). http://pmem.io/2015/06/15/transactions.html. Acessado em: 27/05/2019.

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.

DeBrabant, J., Pavlo, A., Tu, S., Stonebraker, M., and Zdonik, S. (2013). Anti-caching: A new approach to database management system architecture. *Proceedings of the VLDB Endowment*, 6(14):1942–1953.

Diaconu, C., Freedman, C., Ismert, E., Larson, P.-A., Mittal, P., Stonecipher, R., Verma, N., and Zwilling, M. (2013). Hekaton: Sql server's memory-optimized oltp engine. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1243–1254. ACM.

Eldawy, A., Levandoski, J., and Larson, P.-Å. (2014). Trekking through siberia: Managing cold data in a memory-optimized database. *Proceedings of the VLDB Endowment*, 7(11):931–942.

Faerber, F., Kemper, A., Larson, P.-Å., Levandoski, J., Neumann, T., Pavlo, A., et al. (2017). Main memory database systems. *Foundations and Trends® in Databases*, 8(1-2):1–130.

Gomes, D. B. (2019). https://github.com/davibrg/nvrec_ycsb. Acessado em: 26/08/2019.

Huang, J., Schwan, K., and Qureshi, M. K. (2014). Nvram-aware logging in transaction systems. *Proceedings of the VLDB Endowment*, 8(4):389–400.

Kimura, H. (2015). Foedus: Oltp engine for a thousand cores and nvram. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 691–706. ACM.

Maciejewski, M. (2017). https://pmem.io/2016/02/22/pm-emulation.html. Acessado em: 27/05/2019.

Malviya, N., Weisberg, A., Madden, S., and Stonebraker, M. (2014). Rethinking main memory oltp recovery. In *2014 IEEE 30th International Conference on Data Engineering*, pages 604–615. IEEE.

Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., and Schwarz, P. (1992). Aries: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Transactions on Database Systems (TODS)*, 17(1):94–162.

A DBMS-Based Framework for Content-Based Retrieval and Analysis of Skin Ulcer Images in Medical Practice

Mirela T. Cazzolato[1], Lucas S. Rodrigues[1], Lucas C. Scabora[1], Guilherme F. Zabot[1], Guilherme Q. Vasconcelos[1], Daniel Y. T. Chino[1], Ana E. S. Jorge[2], Robson L. F. Cordeiro[1], Caetano Traina-Jr.[1], Agma J. M. Traina[1]

[1] Institute of Mathematics and Computer Sciences
University of São Paulo (USP) - São Carlos, Brazil

[2]Department of Physical Therapy
Federal University of São Carlos (UFSCar), São Carlos, Brazil

{mirelac, lucas_rodrigues, lucascsb, zabot, gui.queirozv}@usp.br
{chinodyt, robson, caetano, agma}@icmc.usp.br, anajorge@alumni.usp.br

Abstract. *Bedridden patients with skin lesions (ulcers) often do not have access to specialized clinic equipment. It is important to allow healthcare practitioners to use their smartphones to leverage information regarding the proper treatment to be carried. Existing applications require special equipment, such as heat sensors, or focus only on general information. To fulfill this gap, we propose* `ULEARn`*, a DBMS-based framework for the processing of ulcer images, providing tools to store and retrieve similar images of past cases. The proposed mobile application* `ULEARn-App` *allows healthcare practitioners to send a photo from a patient to* `ULEARn`*, and obtain a timely feedback that allows the improvement of procedures on therapeutic interventions. Experimental results of* `ULEARn` *and* `ULEARn-App` *using a real-world dataset showed that our tool can quickly respond to the required analysis and retrieval tasks, being up to 4.6 times faster than the specialist' expected execution time.*

1. Introduction

Medical facilities, such as hospitals and healthcare centers, generate large amounts of data daily, which is generally stored in a centralized system, allowing physicians and specialists to query for historical data. Exams such as Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) provide different support to the specialist during the diagnosis [García-Zapirain et al. 2018]. However, not every medical practice have access to such specialized equipment, such as home bedridden wounded patients, which may have limited movement and should be treated at home, or wounded patients in remote regions, where a shortage of specialists exist [Gamus et al. 2019]. Taking advantage of the increasing access to high-quality cameras, health care personnel can take photos of patients' wounds without depending on specialized equipment, and provide decentralized care with enough quality.

Nowadays, mobile devices can acquire high-quality images and act as a feasible, convenient, and cheap alternative for image acquisition [Dorileo et al. 2010]. One scenario where these images are especially useful is in the analysis of chronic skin wounds, often referred to as ulcers [García-Zapirain et al. 2018]. Chronic ulcers often

coincide with other morbidities, adding physical and psychological strain, further limiting a patient's ability to make their way to a specialist, usually located in urban centers [Gamus et al. 2019]. Skin ulcers occur due to reduced blood circulation in lower extremities, injuries, infections, tumors, and other skin conditions [Dorileo et al. 2010]. The visual appearance of these wounds provides clinical signs that may help healthcare practitioners in the diagnosis and treatment. The analysis focusing on the images' content can benefit specialists, providing clues based on past cases, helping them during diagnosis and follow-up procedures. Evaluating whether ulcers contain granulation, fibrin or necrosis allows the healthcare practitioner to analyze the evolution of the patient's condition.

Mobile devices lack on storage and processing power. Having the patients' data and images stored in a database allows not only to follow the patients' condition, but also to provide further knowledge to improve their treatment and the organization of the health system as a whole. Traditional data, such as text and numbers, can be queried using the traditional Database Management Systems (DBMS). On the other hand, considering the content of complex data such as images, videos, and audios require special treatment to store, represent and query them [Traina-Jr. et al. 2000]. DBMS support the analysis of complex data by providing a fast, self-organized, and scalable operations to store and retrieve complex objects and associated information. The most employed query type performed over complex data in DBMS are the similarity queries. They consider the object content, and retrieves the most similar objects in the database, given a similarity threshold or a number of similar objects to retrieve [Lu et al. 2017]. Similarity queries require representing each complex object as low-level feature vectors, retrieved by executing Feature Extraction Methods (FEM). A similarity measure computes the similarity between pairs of objects, often done by a Distance Function (DF) that gives the dissimilarity between pairs of (feature vectors from the) objects.

We are particularly interested in skin ulcer images. The literature has explored representing ulcer using color and texture features for classification purposes [Chino et al. 2018]. Other studies have proposed specific measures to improve the content-based image retrieval and classification of ulcers [Goyal et al. 2017]. However, to the best of our knowledge, every study has focused on the analysis of images using specific applications, without maintaining historical data. There is a lack of approaches with a practical application, considering real-world scenarios, and there is no proper DBMS support to carry similarity-based queries nor to store and use past cases. Past successful cases can be useful to empower the analysis of new patients' data. Existing practical approaches to support ulcer patients' care in remote areas or at home focus on exchanging textual information regarding their conditions [Pedro et al. 2011]. Alternative solutions provide simple information regarding an input image, such as highlighting the wound area using a temperature-based image processing [Fraiwan et al. 2018], but requires specialized equipment such as a mobile thermal camera.

In this work, we propose **ULEARn**, a framework to support health professionals in the analysis of skin ulcer. **ULEARn** is a simple, but accurate and cheap solution for ulcer image analysis, storage and organization. Figure 1 shows how the physician interacts with the system. Briefly, the contributions of this work are:

- The DBMS-based **ULEARn** framework, employed to store images and related information, which supports content-based image retrieval tasks and analysis appli-

Figure 1. Interaction between the physician, `ULEARn-App` and `ULEARn`.

cations with indexing support and a self-organizing structure.
- An ulcer image analysis application, which communicates with **`ULEARn`**, and provides helpful information regarding the ulcer images taken by the physician.
- The mobile application **`ULEARn-App`**, which allows the use of **`ULEARn`** in the medical practice.
- Analysis of a case study that validates **`ULEARn`** and **`ULEARn-App`**.

Paper outline. Section 2 describes relevant background. Section 3 presents related work. Section 4 introduces **`ULEARn`**, the proposed framework to analyze ulcer images. Section 5 presents a experimental analysis. Finally, Section 6 gives the conclusions.

2. Background and Basic Concepts

This section is divided in two parts. First we describe the content-based comparison of images, which can be carried within a database management system. Then we cover image analysis approaches to infer knowledge from ulcers' visual characteristics.

2.1. Content-based comparison of images

Traditional DBMS deals with scalar data such as numbers and dates, which are compared by order $(<, \leq, \geq, >)$ and identity relations $(=, \neq)$ [Traina-Jr. et al. 2000]. However, they have very little use to compare images, where similarity-based operators are better suited. Similarity is studied when performed over a dataset of complex objects. To use those concepts in a DBMS, we assume that the data to be searched is the set of complex values stored in an attribute of a relation. This approach is advantageous, as it allows storing structured information about each object in other attributes in the same tuple.

To make the comparison of complex objects possible, we employ Feature Extractor Methods (FEMs) to represent them and Distance Functions (DFs) to compare the objects. FEMs outputs feature vectors as signatures of the objects' content, with low-level descriptions of the complex data. For instance, to represent images' visual content we can use color, shape and/or texture. A DF is a metric used to compute the (dis)similarity between pairs of complex objects. Examples of DFs are those from the Minkowski family, which includes the Euclidean and Manhattan distances. Let the feature vectors of two complex objects s_i and s_j be respectively v_i and v_j. A DF δ computes the distance between pairs of complex objects $< s_i, s_j >$, such that $v_i = FEM(s_i), v_j = FEM(s_j)$ and $\delta : \mathbb{R}^m \times \mathbb{R}^m$. The DF is a metric if it complies with the following properties:

- *Non-negativity:* $\delta(s_i, s_j) \geq 0$;
- *Identity of the indiscernible:* $\delta(s_i, s_j) = 0 \Rightarrow s_i = s_j$;
- *Symmetry:* $\delta(s_i, s_j) = \delta(s_j, s_i)$;
- *Triangular inequality:* $\delta(s_i, s_j) \leq \delta(s_i, s_k) + \delta(s_k, s_j)$.

Similarity search in datasets only requires defining the search operators, usually the range and k nearest neighbors queries. However, similarity search in DBMS also requires defining the underlying comparison operators, namely Range and k-Nearest Neighbors (k-*NN*) comparisons. Let s_1 and s_2 be two complex objects. The range comparison $s_1\ Rng(\delta, \xi)\ s_2$ returns *TRUE* if and only if $\delta(s_1, s_2) \leq \xi$, where ξ is a similarity threshold (or similarity radius), and δ measures the similarity between complex objects. The k-*NN* comparison $s_1\ k\text{-}NN(\delta, k)\ s_2$ returns *TRUE* when s_1 is one of the k nearest elements to s_2 in the attribute's active domain, according to the similarity measure δ.

We are particularly interested in retrieving images within a similarity threshold from an image used as a query center. Content-based image retrieval (CBIR) systems aim at retrieving images using similarity-based queries. Examples of well-known FEMs for images are those defined by the MPEG-7 standard [MultiMedia 2002], which includes, among others, the Color Layout, Color Structure, and Edge-Histogram FEMs. Other examples, employed on the ulcer analysis task [Chino et al. 2018], are the Local Binary Patterns (LBP) and the Color Histogram. Let $\mathbb{S}$ be an image domain, $S \in \mathbb{S}$ be the set of stored images, $s_i, s_j \in S$ be two image objects, and φ be the FEM employed to obtain the image's signature. When employed over s_i, φ outputs an m-dimensional feature vector $v_i = (f_1, f_2, \ldots, f_m) \in \mathbb{R}^m$, where each dimension $f_j \mid 1 \leq j \leq m$ represents a visual feature, and is represented as an attribute.

A Metric Space (MS) is defined by a pair $< \mathbb{S}, \delta >$, where $\mathbb{S}$ represents the domain of valid objects and δ is a metric [Ciaccia et al. 1997]. This definition allows using Metric Access Methods (MAMs) to index complex objects, speeding-up similarity queries. MAMs consist of indexing structures for complex data storage and retrieval, and examples of well-known MAMs are the Slim-Tree [Traina-Jr. et al. 2000] and the M-Tree [Ciaccia et al. 1997]. We employed the Slim-Tree to index images in this work.

2.2. Image analysis approaches

An image I consists of a set P of n pixels. Let R, G and B be respectively the color channels red, green and blue in the RGB color space. A pixel $p_i \in P$ is a tuple $p_i = (R_i, G_i, B_i)$, which is the pixel intensity of each color channel of RGB color space. A superpixel S_j corresponds to a subset of w pixels $S_j = \{p_j \mid 1 \leq j < w, w \leq n\}$, obtained from a region from I, such that $S_j \subseteq I$, and $w = n$ if $S_j = P$. Superpixels are usually used to subdivide images in color-coherent regions, supporting classification and segmentation tasks. Examples of superpixel methods from literature are SLIC, Felzenswalb and Quick Shift [Achanta et al. 2012].

Let $\mathcal{C}$ be a set of categorical classes, $\mathcal{V} \subset \mathbb{R}^m$ be a set of vectors that represent a particular set of images S_i by using FEM φ_i, and $\mathcal{T}$ be the training set of images, previously classified by an expert with classes from $\mathcal{C}$. A supervised classifier ζ, trained over $\mathcal{T}$, builds a model able of predicting a class $c \in \mathcal{C}$ for each given unclassified image I_u, considering its corresponding feature vector v_u. Simply putting, the classifier corresponds to a function ζ that maps an image feature vector v to a class $c \in \mathcal{C}$. ζ can also be used to classify a superpixel S using the set of available classes. In this work, we consider a set of classes $\mathcal{C}_{ulcers} = \{normal, granulation, fibrin, necrosis\}$, which is useful to the ulcer application scenario. The classifiers used in our analysis and retrieval modules are the Naive-Bayes, K-Means, IBL, and MLP approaches [Zaki and Meira Jr. 2014]. We combined these classifiers with SLIC method to segment and classify skin ulcer images.

A feature vector with the content of the entire image is a global representation of the image. In contrast, feature vectors from specific image regions, such as superpixels, are region-based representations, or local features. Examples of local features are the Bag-of-Visual-Words (BoVW) technique and its variations. BoVW assigns the local features extracted from an image region into one or more visual words in a dictionary. In [Chino et al. 2018] the authors proposed ICARUS, a highly-accurate approach based on BoVW to identify patterns in skin ulcers and improve the content-based retrieval task. We use ICARUS to retrieve similar wound regions of skin ulcer images.

3. Related Work

Existing works related to our approach refers to (i) DBMS-like systems, supporting content-based image retrieval and associated tasks, as well as (ii) existing applications focused on the remote support and treatment of patients with skin ulcer conditions.

Regarding (i), the benefits of CBIR and knowledge discovery on complex data through similarity queries can be applied to several real-world applications, including medicine. However, when CBIR uses a DBMS, the latter is limited to supply the CRUD operations (`CREATE`, `READ`, `UPDATE` and `DELETE`). As a consequence, queries based on predicates over the image attribute and scalar attribute associated to it – such as the capture date or the author – must execute the similarity predicate (over the image) in a separated query engine and then combine it with the result of the scalar predicate from the DBMS. By allowing the DBMS to perform CBIR, both similarity and scalar predicates are integrated, and the queries can be answered using optimized query processing techniques.

The inclusion of CBIR in DBMS requires, at the very least, the provision of FEMs and DFs. Additionally, MAMs can be employed to speed-up queries processing. In that regard, several works integrating CBIR in DBMS can be found in the literature. SIREN [Barioni et al. 2006] proposed a seamless integration of similarity and scalar predicates by extending the SQL to represent similarity operators and implementing a middleware API that combines similarity predicates executed externally with scalar predicates executed by the DBMS. However, the middleware is limited to a query plan that always performs similarity predicates first. RAFIKI [Nesso-Jr. et al. 2018] and FMI-SiR [Kaster et al. 2010] tend to overcome this limitation by implementing similarity retrieval resources through user-defined-functions and extensible interface provided by the DBMS, with RAFIKI being a solution for PostgreSQL and FMI-SiR for Oracle. Both solutions were tested on medical image retrieval scenarios. Since this work uses Oracle as its DBMS, we employed the FMI-SiR similarity retrieval solution to perform queries over the ulcer images. During the execution of a query that uses similarity predicates, the query processor changes the context from SQL to the solution, executes the user-defined function for similarity retrieval and returns information to the query executor. Since it is integrated into the DBMS, these functions can be invoked anywhere in the query plan.

Existing applications focused on the support and treatment of skin ulcers (ii) usually do not analyze the images sent by the users. Those are often referred to as telemedicine applications for wound care, which involves the application of telecommunication technologies to remotely exchange medical information [Chittoria 2012]. In [Fraiwan et al. 2018], the authors proposed a mobile application to work with feet skin ulcers, which uses temperature-based image processing to detect ulcers. However, they

limit their application to images of feet, and also require a mobile thermal camera for image acquisition. Additionally, all processing tasks are performed in the mobile system itself, working with only one image at a time. mULCER [Pedro et al. 2011] is a mobile application to help the ulcer patients' care. mULCER performs image analysis of a user photo, as well as related information regarding the resulting analysis. They do not provide information related to how the images and information are stored, maintained nor processed, and limit the analysis results to only informative texts regarding the sent photo. A systematic review of applications for the prevention of pressure ulcers is presented in [Marchione et al. 2015]. The majority of the reported studies work with sensors and specialized equipment, patient bed position control, and informative textual descriptions. None of the studies described in their work provide a fully-automated analysis of image ulcers to support physicians in daily medical practice.

In this work, we show a DBMS-based solution that support the processing, organization, and communication among different components of a framework to assist the analysis of ulcer images. The proposed image analysis tasks, connected with an similarity-aware extended DBMS, provide the proper tools to assist the medical practice.

4. ULEARn: The Proposed Framework

This section presents the **UL**cer Imag**E** **A**nalysis and **R**etrieval (**ULEARn**) framework, developed to analyze and store skin ulcer images and related information. Figure 2 depicts an overview of **ULEARn**. The framework supports healthcare practitioners through images' content-based retrieval and processing tasks. The professional takes a photo of the ulcer being treated and submits it for analysis. The framework stores relevant information, allowing him/her to follow the evolution of the patient. **ULEARn** controls the interaction among different modules that can be swiftly integrated to a DBMS to take into account data from a set of images, and not from just one, as is the current state-of-the-art. This work is the first effort in such integration, pointing out resources for storage and query processing required from the DBMS. The ability to easily integrate modules of customized image analysis to a CBIR system is our leading research contribution.

ULEARn's workflow is composed of four main layers: (I) Mobile Application and Data Exchange; (II) Application Manager; (III) Image Analysis; and (IV) Storage and Similarity-Based Retrieval. Layers III and IV are inside **ULEARn**, while Layers I and II work as services above the proposed framework, making use of its features and controlling the user/framework interaction. We explain each layer in the next sections.

4.1. Layer I: Mobile Application and Data Exchange

ULEARn provides an infrastructure for mobile applications to communicate with the database and the image analysis methods. As mobile devices have limited capability to store and process large amounts of information, the support of a DBMS to store and process the data is primordial. We implemented the mobile app **ULEARn-App** as a proof of concept to test the proposed framework (see implementation details in Section 5). **ULEARn-App** provides a proper authentication for physicians to access the app and past patients' information. As the application stores personal information, the access to the app is restricted for trusted users. **ULEARn-App**'s main task is sending an image taken from the patient. The user also stores textual information in the database, such as patients'

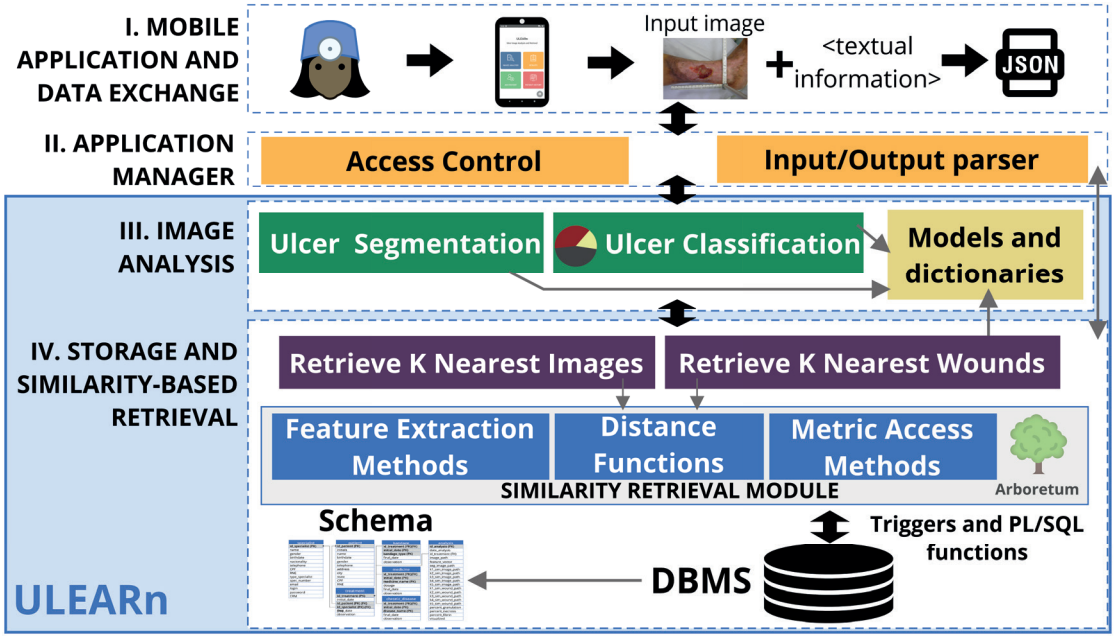


Figure 2. `ULEARn` framework overview.

data and treatment history. All data is encapsulated in a `JSON` file and sent to the server with a request for the analysis results. When the remaining layers finish processing, the app receives the results to be displayed for the physician, also as a `JSON` file.

4.2. Layer II: Application Manager

The application manager intermediates the communication between the app and the framework, and performs the access control and the input/output parser. The access control corresponds to authenticating the physician using the app, connecting to the framework, consulting, and inserting associated information. For security reasons, a physician can only access the information of patients that are linked to he/she. The input/output parser calls each of the image analysis and the storage/retrieval modules. Once the analysis are done, the parser sends the results back to Layer I.

4.3. Layer III: Image Analysis

In order to provide the relevant feedback to the physician, **`ULEARn`** implements two image analysis tasks, Ulcer Segmentation (*US*) and Ulcer Classification (*UC*). We analyzed *US* and *UC* with different off-the-shelf classifiers. MLP and 3NN presented the best F-Measure results for *US* and *UC* respectively. However, as **`ULEARn`** is a modularized solution, its customization by exchanging each functionality as a black box inside the framework is feasible and direct.

Ulcer Segmentation (US). Given the patient image as input, *US* segments the image into a set of superpixels $S_i \in \mathcal{S}$. *US* extracts the feature vector of each superpixel region using the Color Histogram FEM. Then, *US* classifies each feature vector according to the class set $\mathcal{C}_{wounds} = \{wound, not\_wound\}$. *US* uses the 3NN classifier and a pre-trained model stored in the framework. As a result, *US* composes a segmented image with only the regions classified as *wound* ones. Regions classified as *not_wound* are painted black, so

only the segmented wound will be depicted in the resulting image. The segmented image is used as input for the next step, the ulcer classification (*UC*).

Ulcer Classification (UC). Given the segmented image, *UC* extracts its feature vector with Color Histogram FEM. *UC* classifies the feature vector according to the different healing stages $\mathcal{C}_{stages} = \{fibrin, granulation, necrosis\}$. *UC* uses the MLP classifier, and outputs the probability of the image to contain each class in $\mathcal{C}_{stages}$.

4.4. Layer IV: Storage and Similarity-Based Retrieval

Following we detail how **ULEARn**: (i) stores all information regarding the physicians, the patients, the treatments, and the analysis results in a relational database; (ii) retrieves the most similar images (*kNI*) and wounds (*kNU*) to each query. **ULEARn** extends the DBMS Oracle, primarily used for storage purposes only, to also perform feature extraction, distance calculations, complex data indexing, and the execution of similarity queries. These functionalities correspond to the similarity retrieval module, depicted in Figure 2, provided by the Arboretum library[1]. **ULEARn** provides an interface to Arboretum, leaning on Oracle for calling the functions through triggers and PL/SQL functions.

(i) Storage. Figure 3 presents **ULEARn**'s data schema. Tables *specialist* and *patient* store personal information, and Table *treatment* links the specialist to the patient and stores specific information regarding its condition. A single treatment can have one or more dressings, medication, and chronic diseases registered, which allows the framework to maintain the medical records of the patient for further inquiries. Also, table *analysis* stores the patient's photos sent by the app, and all image analysis results performed in Layer III (Image Analysis).

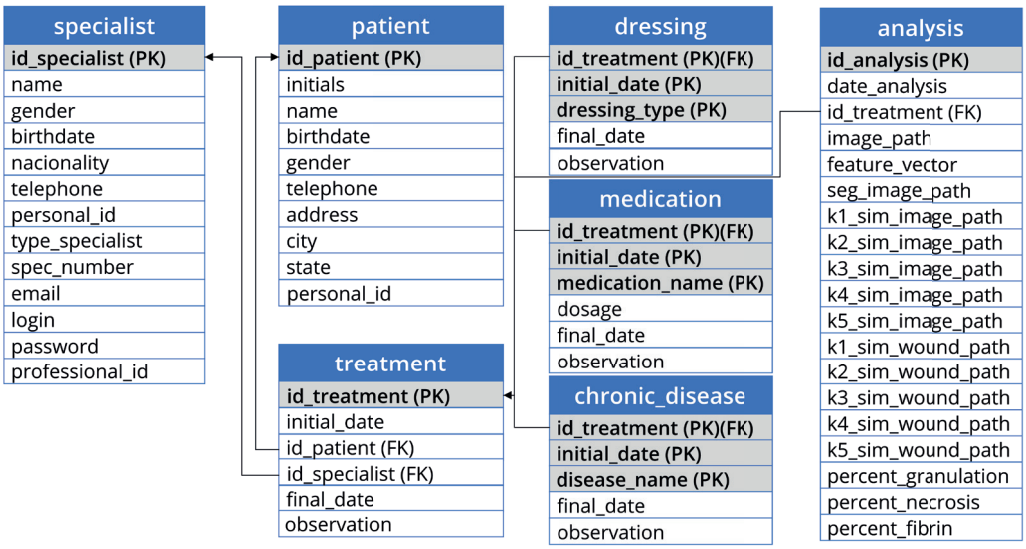


Figure 3. ULEARn schema, with information regarding the patients, professionals, treatment and analysis results.

(ii) Similarity-Based Retrieval. **ULEARn** relies on past cases to perform similarity-based queries for the two following tasks. They rely on the extensions performed over the DBMS, in order to allow the similarity-based representation and comparison of images.

[1]The Arboretum library: https://bitbucket.org/gbdi/arboretum/src/master/.

Retrieve K Nearest Ulcers (kNU). Given the patient's image as input, *kNU* segments the image into a set of superpixels, using an approach based on bag-of-visual-words to represent the regions' visual patterns. With the segmented image, *kNU* retrieves the k most similar images stored in the database, using the similarity retrieval module, which was included in the DBMS as depicted in Figure 2.

Retrieve K Nearest Images (kNI). Given the patient's image as input, *kNI* inserts the received image into the DBMS, which performs a kNN query using past cases, supported by a metric access method to speed-up the task. This is also relevant because the wound heals/reacts differently in different body parts, and considering the skin around the wound area is relevant to evaluate the patients' condition. The k most similar images are returned to the user, considering the entire image's similarity to the previously stored ulcer images, from other patients.

For *kNU*'s and *kNI*'s similarity retrieval tasks, the DBMS is in charge of receiving the image from the application manager layer of **ULEARn**, extracting its feature vector and storing it as a `BLOB` type attribute. The query of Statement 1 performs this step.

Statement 1. Receiving, processing and storing a new image.

```
DECLARE
   fvector BLOB;
   result PLS_INTEGER;
BEGIN
   dbms_lob.createtemporary(myblob, true);
   result := extractFeatureVector('image_path',fvector);
   INSERT INTO ANALYSIS(...,fvector,...) VALUES (...,fvector,...);
   dbms_lob.freetemporary(myblob);
END;
```

Statement 1 exemplifies feature extraction and storage, executed during a new image analysis, using the Color Layout FEM, which is executed by the similarity retrieval module. The corresponding feature vector is the *fvector* variable, which is then stored in the database along with other information (represented by '...'). To retrieve the k most similar images considering the Color Layout FEM and the Euclidean DF, **ULEARn** performs the query shown as Statement 2. Consider a newly inserted image using Statement 1, identified by *id_analysis*$= 10$ as the query center. The query in Statement 2 will retrieve its 5 most similar images considering Color Layout and the Euclidean distance. Remember from Section 3 that our employed Similarity Retrieval Solution implements its resources as user-defined-functions and extensible interfaces. Figure 4 shows the query workflow, for the insertion of a new image.

Statement 2. Retrieving the 5 most similar images from image 10.

```
SELECT id_analysis, image_path FROM analysis
WHERE euclidean_knn(fvector,(SELECT fvector FROM analysis
                             WHERE id_analysis = 10)) <= 5;
```

5. Experimental Analysis

Setup. We implemented **ULEARn-App** for Android OS using Kotlin, the web framework Flask to manage the communication between the app and the server, and all algorithms

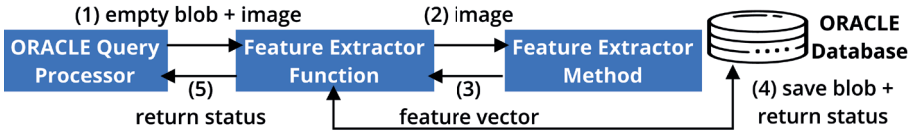


Figure 4. Interactions of the Similarity Retrieval Module to insert a new image.

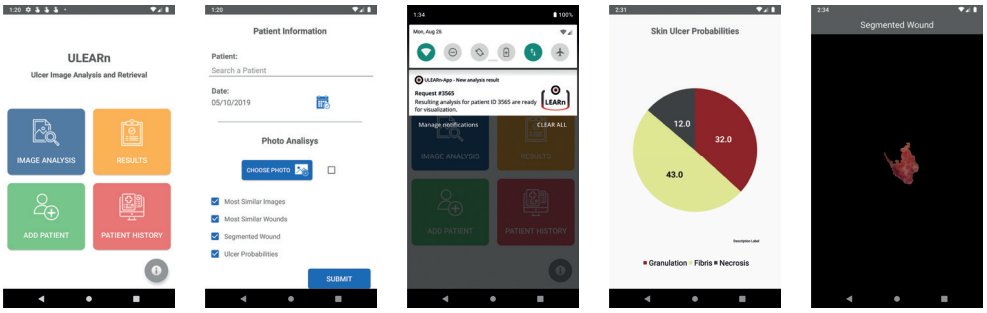


Figure 5. Screenshots of `ULEARn-App`.

were implemented in Python 3.6.5. The server is an Intel Xeon(R) X5650 2.67GHz 4vCPU, 16GB virtual machine, running Ubuntu 16.04 LTS, and the library *cx_Oracle* 7.1.3 to connect to the Oracle DB-12c - 64bit server.

Case Study. Figure 5 shows examples of screenshots taken from **`ULEARn-App`**[2]. In summary, the app has the following features implemented: login, physician, patient and treatment registration, analysis' results, and patient history, serving as a follow-up chart for the physician. We consider as the analysis focus the feedback from a physician from the Federal University of São Carlos, that works with ulcer patients. The ulcers specialist emphasizes that obtaining a response *within five minutes* is primordial. This allows them to take a photo from the patient, at the beginning of the visit, and obtain a proper response to improve the intervention while they are still working with the patient for treatment/therapy. Still, **`ULEARn-App`** is interactive, and while the physician waits for a response, the app will release its functionalities, not freezing up while **`ULEARn`** remotelly processes the analysis request. This way, the user can use their smartphone normally, and can also concomitantly send other images for analysis. The app will show a notification informing when the analysis is ready for visualization (3rd screen of Figure 5). We evaluate our framework using a physicians' provided dataset of ulcer images, testing the various components of **`ULEARn`** in terms of execution time. The dataset contains 216 images, consisting of ulcer photos taken by specialists from ulcer patients using smartphones.

Time Analysis. Figure 6 gives the execution time of each image analysis component, with different resolution (width varying from 320 to 3,264 pixels) using $k = 5$ for k-*NN* Retrieval. The segmentation and classification tasks (a) take up to 40.9 seconds but is faster as image resolution decreases. The wound retrieval (b) presents nearly constant execution time from the resolution of 1280 pixels, requiring at most 10.3 seconds. The image retrieval (c) used 50 random query objects, and it was much faster than the other analysis tasks, as it is performed in the DBMS using the Slim-Tree MAM. Its execution time was higher regarding the increased resolution, performing in up to 4.1 seconds.

Figure 7 shows the full processing execution time considering the sum of every

[2]The app and further details are given on https://github.com/mtcazzolato/ulearn-app.

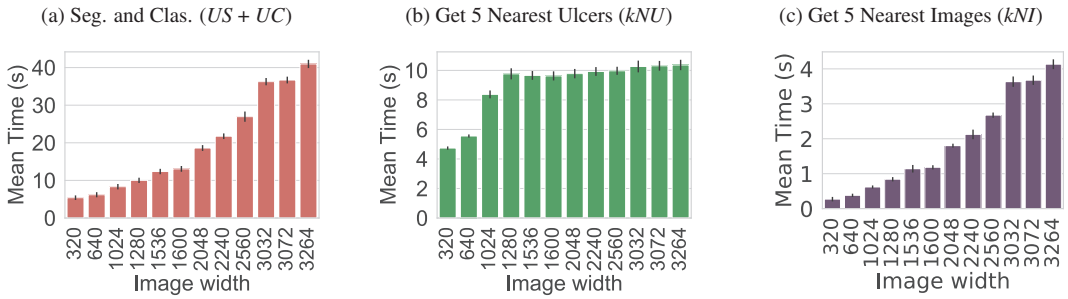


Figure 6. Mean execution time for the image analysis task of ULEARn.

analysis and database operations (d), and also considering the average app interaction and network latency times from **ULEARn** (e). In (d), we observe that the sum of every analysis algorithm shows a linear growing behavior on the image resolution. Now regarding **ULEARn-App** and **ULEARn** (e), we observe that the entire processing pipeline occurs within an execution time of up to 65.8 seconds, for high-resolution images. Notice that, part of this time corresponds to sending the image through the network. All in all, **ULEARn** has shown to perform in a feasible interval of time, serving its purpose and allowing the physician to obtain feedback within the desirable time frame.

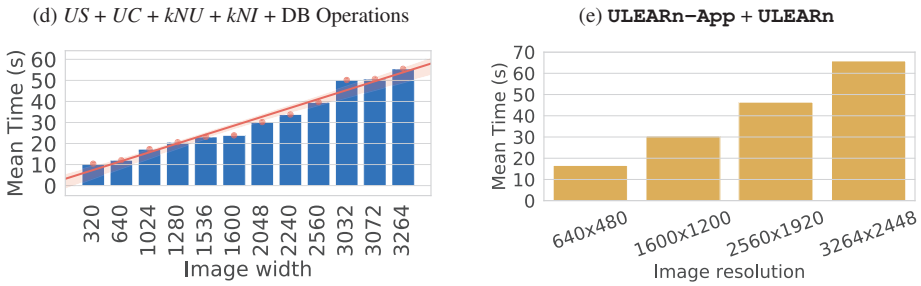


Figure 7. Mean execution time for all processing of ULEARn and ULEARn-App.

6. Conclusions

We presented **ULEARn**, a framework that supports the analysis and storage of skin ulcer images. **ULEARn** provides relevant feedback for health professionals regarding patients' ulcer, based on a DBMS to support the processing, organization and communication between its different components. **ULEARn-App** is an Android app that servers as a proof of concept regarding the usefulness of **ULEARn**. Experiments showed that **ULEARn** is up to *4.6 times faster* than the specialist's expectation, even when working with high resolution images. Specialists can use **ULEARn-App** in the day-to-day clinical scenario, for a patient checkup or in places that lack specialized wound care intervention. Future work include adding a temporal analysis on the patients' condition, and further test of **ULEARn-App** in real scenarios. Among the many motivations for proposing **ULEARn** and **ULEARn-App** is to allow physicians in the data acquisition, storage, and retrieval for analysis. We foresee that **ULEARn** will make it possible for physicians and data analists, in the near future, to collect a larger number of images and associated information, further improving the classification and segmentation models, including the use of Deep Learning approaches. Also, **ULEARn** can be easily adapted to other application contexts.

Acknowledgments

This research was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, by the São Paulo Research Foundation – FAPESP (grants #2018/24414-2, #2016/17078-0, #2016/17330-1, #2014/25125-3), and the National Council for Scientific and Technological Development (CNPq).

References

Achanta, R. et al. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *TPAMI*, 34(11):2274–2282.

Barioni, M. C. N., Razente, H. L., Traina, A. J. M., and Traina-Jr., C. (2006). SIREN: A similarity retrieval engine for complex data. In *VLDB*, pages 1155–1158.

Chino, D. Y. T. et al. (2018). Icarus: Retrieving skin ulcer images through bag-of-signatures. In *CBMS*, pages 82–87.

Chittoria, R. K. (2012). Telemedicine for wound management. *Indian J Plast Surg*, 45(2):412–417.

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.

Dorileo, E. A. G., Frade, M. A. C., Rangayyan, R. M., and Azevedo-Marques, P. M. (2010). Segmentation and analysis of the tissue composition of dermatological ulcers. In *CCECE*, pages 1–4.

Fraiwan, L., Ninan, J., and Al-Khodari, M. (2018). Mobile application for ulcer detection. *TOBEJ*, 12:16–26.

Gamus, A., Keren, E., Kaufman, H., and Chodick, G. (2019). Synchronous video telemedicine in lower extremities ulcers treatment: A real-world data study. *IJMI*, 124:31–36.

García-Zapirain, B., Elmogy, M., El-Baz, A., and Elmaghraby, A. S. (2018). Classification of pressure ulcer tissues with 3d convolutional neural network. *MBEC*, 56(12):2245–2258.

Goyal, M., Yap, M. H., Reeves, N. D., Rajbhandari, S., and Spragg, J. (2017). Fully convolutional networks for diabetic foot ulcer segmentation. In *SMC*, pages 618–623.

Kaster, D. S., Bugatti, P. H., Traina, A. J. M., and Traina-Jr., C. (2010). FMI-SiR: A flexible and efficient module for similarity searching on oracle database. *JIDM*, 1(2):229–244.

Lu, W., Hou, J., Yan, Y., Zhang, M., Du, X., and Moscibroda, T. (2017). MSQL: efficient similarity search in metric spaces using SQL. *VLDB J.*, 26(6):829–854.

Marchione, F., Araújo, L., and Araújo, L. (2015). Approaches that use software to support the prevention of pressure ulcer: A systematic review. *IJMI*, 84(10):725–736.

MultiMedia, I. (2002). Mpeg-7: The generic multimedia content description standard, part 1. *IEEE MultiMedia*, 9(2):78–87.

Nesso-Jr., M. R. et al. (2018). RAFIKI: retrieval-based application for imaging and knowledge investigation. In *CBMS*, pages 71–76.

Pedro, L. M. C. C., Rodrigues, J. J. P. C., and Martins, H. M. G. (2011). mUlcer – a mobile ulcer care record approach for integrative care. In *EIS*, pages 392–401.

Traina-Jr., C., Traina, A. J. M., Seeger, B., and Faloutsos, C. (2000). Slim-trees: High performance metric trees minimizing overlap between nodes. In *EDBT*, pages 51–65.

Zaki, M. J. and Meira Jr., W. (2014). *Data Mining and Analysis - Fundamental Concepts and Algorithms*. Cambridge University Press, New York, NY, USA.

A Parallel-based Map Matching Approach over Urban Place Records

Tiago Brasileiro Araújo[1], Carlos Eduardo Santos Pires[1], Demetrio Gomes Mestre[2], Andreza Raquel Monteiro de Queiroz[1], Veruska Borges Santos[1], Thiago Pereira da Nóbrega[2]

[1]Universidade Federal do Campina Grande (UFCG), Campina Grande, PB – Brasil

[2]State University of Paraíba (UEPB), Campina Grande, Brazil

{tiagobrasileiro@copin,cesp@dsc,andreza.queiroz@ccc,veruska

.santos@ccc}.ufcg.edu.br, {demetriogm,thiagonobrega}@uepb.edu.br

Abstract. *In the Smart Cities scenario, to avoid the conflicting geospatial records between official and non-official sources, it is necessary to detect the inconsistencies regarding the geospatial data provided by them. To this end, the map matching task, i.e., the task of identifying correspondent features between two geospatial data sources, should be applied. For spatial Big Data, the map matching task is confronted with challenges related to volume and veracity of the data. In this sense, we propose a Spark-based map matching approach, called MATCH-UPS. To evaluate, real-world data sources of New York (USA) and Curitiba (Brazil) were applied. The results showed that MATCH-UPS improved the precision by 26% and reduced the execution time by one third.*

1. Introduction

Geographical coordinates and maps (digital or paper-based) are a common element of our daily life which provide a two-dimensional representation of geographical features in the real world, such as parks, bus stops, roads, rivers, buildings, and places. Such information is often referred to as geospatial or geographical data and plays an essential role in many governmental, economic and social domains, such as disaster response, urban planning, and tourism [Du et al. 2017, Harb and Becker 2018]. In this sense, the large amount of data collected by municipalities and map projects (e.g., Open Street Map) may be used to improve the efficiency of public transportation and infrastructure investments. However, since geospatial data sources are prone to inconsistencies and quality issues, it is important to assess the data quality before using them to take valuable strategical decisions. In fact, an analysis based on incorrect information can lead to wrong decisions [Christen 2012].

For Smart Cities scenario, the municipalities benefit of the data collected from multiple sources, such as official documentation, third parties information, and environmental sensors [Bojic et al. 2015, Fan et al. 2014]. To avoid the conflicting records between official and non-official sources it is necessary to detect the inconsistencies regarding the geospatial data provided by them. To this end, the Map Matching task, i.e., the task of identifying correspondent features between two geospatial data sources [Du et al. 2017], can be applied. For instance, map matching can be used to identify geospatial records (e.g. parks, bus stops, and roads) from distinct data sources that refer to the same real-world place. It is an essential pre-processing step for data integration,

change detection, data updating, and data comparison [Fan et al. 2014]. Map matching is a powerful task to assist several projects interested in extract information from geospatial data (for instance, control of deforestation, fires and floods) and approaches related to smart cities [Bojic et al. 2015]. Several works report the necessity to apply the map matching task to solve diverse problems related to urban mobility and environment in different countries, such as USA [Pi et al. 2018], Chile [Arriagada et al. 2019], Sweden [Bouguelia et al. 2018], and Italy [Interdonato and Tagarelli 2017]. In this context, the present work has been conceived by analyzing the scenario and related issues addressed by an international cooperation project called EUBra-BIGSEA. This project aims to develop a cloud platform for data management and exploitation. In particular, the services are able to empower data analytics to support the development of data processing applications.

In the context of spatial Big Data, the map matching task is confronted with two main challenges: *volume*, as it handles a large amount of geospatial records (such as buildings, parks and bus stops of a megalopolis); and *veracity*, related to the reliability of the data provided by the source [Christen 2012]. The desktop GIS software typically takes hours to process (or compare) these massive geospatial data. Such a consumption of time is unacceptable for many applications, particularly for real-time policy decisions such as predicting which areas would be damaged by natural disasters. In this light, blocking techniques and parallel computing are applied in order to minimize the problems related to the large volume of data. Blocking techniques group similar records into blocks and perform comparisons of the records within each block. The map matching in parallel aims to reduce the overall execution time by distributing the comparisons between geospatial records among the various resources (e.g., computers or virtual machines) of a computational infrastructure. Regarding the veracity, the map matching task can be applied in order to assess the quality of data sources. For instance, by identifying conflicting and inconsistent records between them [Fan et al. 2014]. Problems related to veracity often occurs due to the data are collected through crowd-sourcing. For this reason, the OSM has often been denounced due to its heterogeneity in quality from the beginning of its development and needs to be evaluated by comparing with authority data [Fan et al. 2016].

In the literature, there are several map matching approaches [Fan et al. 2016, Fan et al. 2014, Du et al. 2017, Alarabi et al. 2017]. However, most of them address the problem of road network matching [Fan et al. 2014]. In this work, we concentrate on two open areas not addressed by the previously mentioned works concomitantly: map matching (involving points and polygons) and parallel computing, applying Apache Spark. Overall, the contributions of our work are the following: i) our approach, called Matching of Urban Places with Spark (MATCH-UPS), proposes the parallelization of the map matching approach proposed in [Fan et al. 2014]. The novel approach applies the Spark framework to parallel the execution of the map matching task; ii) we propose the application of the semantic attributes (e.g., names of the places) and context information (e.g., neighbouring streets) for the matching task in order to assist the comparison of geospatial records (i.e., polygons or points). This information provides additional evidence used to assist the definition of the corresponding places (i.e., match) in the map matching task; and iii) we propose the application of the blocking technique based on the geographical similarity (from the coordinates) to group similar geospatial records into the same block.

2. Background

Matching Geospatial Data. In map matching, it is necessary to compare the geospatial records of one data source with the geospatial records of the other one. To determine the similarity of a record pair, geospatial and semantic attributes (e.g., geolocations, place names, addresses, or postcodes) can be compared [Christen 2012]. If the records of a record pair present various similar attribute values, these records have high chances of being considered similar. According to the type of geospatial data (in this work, points and polygons), the geospatial information can be explored in different ways. To compare the geospatial information of two points (i.e., latitude and longitude of each point), the distance between them is commonly evaluated [Fan et al. 2014, Xavier et al. 2016]. The closer the points are, the greater the chances of them being considered as correspondents. Regarding the polygons, the geospatial information (i.e., a set of latitude and longitude for each polygon) can be evaluated based on the overlapping area of the polygon pair. To improve the efficacy of the map matching results, semantic attributes (e.g., names, descriptions, and annotations) of record pairs can also be compared beyond the evaluation based on the geospatial attributes [Christen 2012]. For instance, the names of the buildings (semantic attribute) are linguistically compared (using linguistic measures such as Jaccard or Levenshtein) to determine the similarity degree between them.

To determine whether a record pair is a correspondence or not, the similarity values can be combined (e.g., given by average or weighted average) or used individually to compound a rule of classification. The most two commonly classifiers applied in this context are: threshold-based and rule-based [Christen 2012]. The former combines the similarity values (e.g., geospatial similarity and semantic similarity) and determines as a correspondence the record pairs that have a final similarity value higher than a given threshold. Related to the latter, a record pair can be classified not only as match or non-match. Thus, a record pair is classified as one of the predetermined classification profiles (for instance, match, possible match, or non-match) according to the rule, taking into account all the similarity values (e.g., geospatial similarity and semantic similarity). In this work, we applied rule-based and threshold-based classifiers according to the number of attributes assessed (consequently, the number of similarity values) in each scenario (i.e., involving polygons or points), as will be described in Section IV.

3. Related Work

The survey [Xavier et al. 2016] highlights several works which propose map matching approaches in the context of geographical data. In the work [Fan et al. 2016], a polygon-based approach is proposed to match roads and urban blocks provided by Open Street Map (OSM) and official data sources. The algorithm represents the urban blocks (i.e., elements of urban planning) as polygons (e.g., surrounding buildings) and lines (e.g., surrounding streets). Thus, the algorithm is able to match urban blocks evaluating their spatial topologies based on the polygons. To determine the similar polygons (which model the urban blocks), the algorithm evaluates the overlapping areas between the polygons.

Similar to [Fan et al. 2016], the work [Fan et al. 2014], which inspired our work, proposes a method for matching building footprints (i.e., polygons), in order to assess the quality of the data provided by OSM. The similarity between polygons is defined by the percentage of overlapping area between them, using 30% as the threshold for

considering a polygon pair as a match. However, the work [Fan et al. 2014] presents limitations when the same building is represented as two disjoint polygons (commonly in scenarios where the polygons have a small area) in OSM data and authoritative data [Du et al. 2017]. Regarding context-based map matching, the work [Zhang et al. 2014] affirms that in cases when there is ambiguity in the correspondences, the similarity of geographical features depends on the context. This work proposes a triangulation-based approach to define the neighbourhood of urban places, considering a continuous influence from the closest places.

In these terms, although the works previously discussed address the map matching task, none of them proposed approaches to address map matching and parallel computing simultaneously. Despite the work [Alarabi et al. 2017] apply parallel computing over the geospatial records management, it does not address the map matching task. Thus, applying parallelism/cloud computing in the context of geographical Big Data is treated as an open research area [Zhang et al. 2015, Xavier et al. 2016]. The work [Zhang et al. 2015] also highlights the lack of parallel-based map matching approaches. Therefore, our work emerges as a bridge between map matching and parallel computing since we propose a Spark-based approach that parallelizes the map matching task. In addition to assessing the geographical similarity of the geospatial records (provided by the overlapping area or the distance between the records), our work takes into account other attributes (e.g. place names) of the records and applies context-based information. Regarding context-based map matching, our work considers the streets as context information and applies rules (instead of triangularization) based on the distance between bus stops/streets to determine which bus stops are considered correspondents.

4. The MATCH-UPS Approach

4.1. Overview

Given two sets of geospatial records, represented by D_1 and D_2, the map matching task consists in identifying all correspondences between records. We denote the schema followed by the records of D as $E = (a_1, a_2, \ldots, a_n)$, in such a manner that each a_i corresponds to an attribute (e.g., name, category, and geographical coordinates). Therefore, the input data sources D_1 and D_2 contain a finite set of records denoted as a pair of $\langle attribute, value \rangle$: $r = [\langle a_1, v_1 \rangle, \langle a_2, v_2 \rangle, \ldots, \langle a_n, v_n \rangle]$. Let $sim(r_1, r_2)$ be the similarity measure between records r_1 and r_2, Φ_{max} the maximum threshold that defines whether r_1 matches r_2, and Φ_{min} the minimum threshold that defines the pair r_1 and r_2 as non-match. Thus, the map matching can classify the pairs of records as $Match(M) = \{(r_i, r_k) \mid r_i \in D_1, r_k \in D_2 \; and \; sim(r_i, r_k) \geq \Phi_{max}\}$, $Non - Match(NM) = \{(r_i, r_k) \mid r_i \in D_1, r_k \in D_2 \; and \; sim(r_i, r_k) \leq \Phi_{min}\}$ and $PotentialMatch(PM) = \{(r_i, r_k) \mid r_i \in D_1, r_k \in D_2 \; and \; \Phi_{min} < sim(r_i, r_k) < \Phi_{max}\}$.

In this work, we propose a Spark-based map matching approach, denoted as MATCH-UPS[1], for dealing with records represented by spatial geometries (e.g., polygons and points). Thus, the MATCH-UPS approach performs pairwise comparisons between geospatial records, determining the similarity between them. To classify a record pair, linguistic and geographical matchers are applied. A linguistic matcher explores the textual attributes of the record (e.g., name and description) to determine the linguistic similarity

[1]https://github.com/brasileiroaraujo/GeoMatch-MATCH-UPS-.

Table 1. Classification rules.

Rule	Classification
LS >*LST* and *GS* >*GST*	*Match*
LS <*LST* and *GS* <*GST*	*Non-Match*
(*LS* >*LST* and *GS* <*GST*) or (*LS* <*LST* and *GS* >*GST*)	*Potential match*

(*LS*) of a record pair. To this end, algorithms such as Jaccard and Levenshtein distance are used. A geographical matcher explores the coordinates of both geospatial records in order to identify the overlapping area or the distance between them. The proportion of the overlapping areas (for polygons) or the distance (for points) represent the geographical similarity (*GS*) between the records of a pair. Thus, the record pairs are categorized according to the classification rule show in Table 1 which considers the similarity values as well as a linguistic similarity threshold (*LST*) and a geographical similarity threshold (*GST*). The record pairs are classified into threes categories: *match*, *non-match*, and *potential match*.

4.2. Geospatial Polygons

Regarding the polygons (for instance, buildings, residential regions, parks, and forests), the similarities between the geospatial records can be measured through the linguistic and geographical matchers. As previously mentioned, the linguistic matcher can apply algorithms such as Jaccard and Levenshtein distance to determine the linguistic similarity between the records. To measure the geographical similarity, the overlapping area between the polygons of the geospatial records is evaluated. In this sense, works such as [Fan et al. 2014] consider that a pair of geospatial records with an overlapping area above 30% are classified as a match. Therefore, the geographical similarity threshold (*GST*) applied to polygons (records) pairs is 0.3 (30%). Equation 1 denotes the rule to measure the similarity of two polygons (p_1 and p_2):

$$sim(p_1, p_2) = min(\frac{overlappedArea(p_1, p_2)}{area(p_1)}, \frac{overlappedArea(p_1, p_2)}{area(p_2)}) \quad (1)$$

4.3. Geospatial Points

To perform the map matching task over points records (for instance, bus stops, points of interest, and vehicle coordinates), linguistic and geographical matchers can be applied. In this work, we apply only a geographical matcher since the records provided by the data sources do not contain relevant linguistic attributes. The geographical matcher considers as *match* the record pairs with a distance (in meters) lower than a certain *GST* (also given in meters). Otherwise, the pair of records is considered as *non-match*. In this work, we apply a *GST* of 20 meters, since the work [Yang et al. 2014] proved experimentally that a pair of geospatial points with a distance lower than 20 meters can be considered as a match. On the other hand, if we consider only the geographical distance, two problems can be highlighted. Since several bus stops may be close to each other in the real world, a bus stop can be classified as match more than once with different representations of bus stops contained in the other data source. Moreover, the application of the geographical

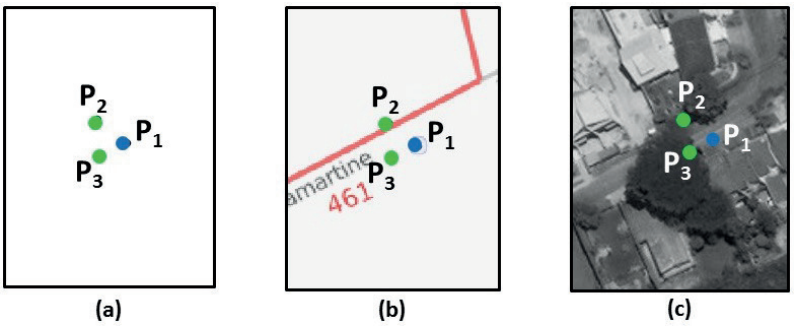


Figure 1. Applying the context information over bus stops data sources.

matcher can only reduce the reliability of the results since the bus stops may be close but might not represent the same bus stop. In order to minimize these problems, a context-based strategy was proposed.

Context-based Map Matching. Since the attributes of records are exploited in the map matching task in order to determine the similarity between a pair of records, the lack of comparable attributes reduces the confidence of the map matching results. The matching of urban areas is complex due to the ambiguities involved such as many-to-many correspondences, the positional discrepancy between geographical objects in two data sources, distinct map scales and objects with simplified descriptions or the absence of comparable attributes. This requires the application of context-based matchers [Zhang et al. 2014]. Hence, the map matching approaches can apply contextual information (e.g., surrounding geographical information such as streets, buildings, and urban blocks) to improve the accuracy of the results. The context information can be provided by external data sources in order to support the map matching task.

In this sense, we propose a context-based MATCH-UPS to compare geographical points (e.g., bus stops records). Considering that the only comparable attribute between two data sources is the georeferenced point (latitude and longitude), determining whether or not a pair of records (i.e., points) is a match turns out a challenging and imprecise task. For instance, Figure 1(a) depicts three geographical points (P_1, P_2 and P_3), where each point represents a bus stop. The bus stop P_1 (in blue) is provided by the municipality data source whilst bus stops P_2 and P_3 (in green) are provided by OSM. Since the distances (in meters) from bus stop P_1 to bus stops P_2 and P_3 are equivalent, it is not possible to determine which bus stop (P_2 or P_3) correspondends to bus stop P_1. Thus, the context of the bus stops (in this case, the surrounding streets) is applied to the map matching task in order to assist the matchers in the classification of pairs of records. In this case, the pair $\langle P_1, P_3 \rangle$ is considered a correspondence since the bus stops P_1 and P_3 are at located the same side of the street while the bus stop P_2 is positioned at the other side of the street, as illustrated in Figure 1(b) and 1(c).

4.4. An Efficient Approach for Map Matching Task

In this section, we describe the workflow of the MATCH-UPS approach, which combines blocking techniques and parallel computing to enhance the efficiency of the map matching task. The former groups similar records into blocks and perform comparisons within each block. In this work, we apply part of the geographical coordinates (i.e., blocking key) to block the records. In other words, records that share the same blocking key (based on the

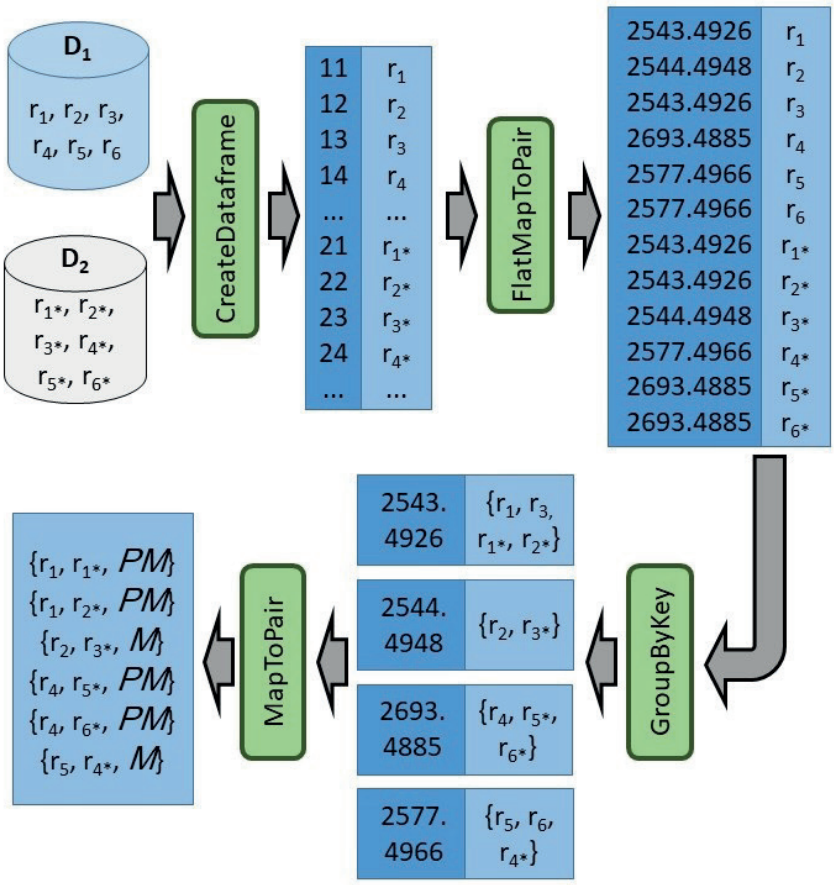


Figure 2. The MATCH-UPS workflow.

geographical coordinates) are grouped into the same block. Since a pair of records located at a long distance from each other has few chances to be similar, the blocking technique prevents that this pair of records from being compared. Regarding the map matching task in parallel, the Spark framework is applied to reduce the overall execution time by distributing the comparison between geospatial records among the available resources. Figure 2 depicts the whole workflow of the MATCH-UPS approach.

The proposed approach receives as input a set of records provided by two data sources D_1 and D_2. To a better understanding, the records (r) provided by D_2 are marked by the symbol "*". Initially, the records (r) are mapped as a key-value pair (i.e., FlatMapToPair operator), such that the key is formed by the data source ID (which provides the record) concatenated with the record ID whilst the value is the record itself. For instance, record r_1 provided by D_1 is mapped to the pair $\langle 11, r_1 \rangle$. These key-value pairs are stored in a DataFrame structure. Posteriorly, the first "n" numbers (in this example, $n = 4$ was applied) of the latitude and longitude coordinates are extracted from each record (if the record is a polygon, the coordinates are extracted from the centroid point). These numbers are concatenated in order to generate the blocking keys. Therefore, each record generates a blocking key composed of the first four numbers of the latitude and longitude coordinates. In the example, the first four numbers of the latitude ("*2543*") and longitude ("*4926*") coordinates of the record r_1 are concatenated to generate the pair $\langle 2543.4926, r_1 \rangle$. In the next step, all records sharing a

common blocking key are grouped into the same block (i.e., GroupByKey operator). For this reason, the records r_1, r_3, r_1* and r_2* are grouped into the same block since all records share the key "*2543.4926*". The blocks determine which records should be compared (by the matchers). Notice that each block is sent to an available resource, where the comparisons between records are performed. Finally, after comparison, the record pairs considered as a match or potential match are joined to compose the MATCH-UPS output (i.e., MapToPair operator). In Figure 2, the output generated for the example is $\{r_1, r_1*, PM\}, \{r_1, r_2*, PM\}, \{r_2, r_3*, M\}, \{r_4, r_5*, PM\}, \{r_4, r_6*, PM\}$ and $\{r_5, r_4*, M\}$.

Load balancing. In the map matching task, the most costly step (in terms of computational cost) is the comparison step, where the matchers are applied to compare the attribute values of each record pair. First of all, it is important to highlight two points: i) all records contained in a block are sent together to a specific resource (e.g., node) to be compared; and ii) since the blocks can have different number of records, the blocks can generate a different amount of comparisons between records. Due to these two points, the comparison step may suffer from load imbalancing problems. Load imbalancing occurs when some nodes execute comparisons for a long time while other nodes remain idle [Araújo et al. 2016].

For instance, in Figure 2, four blocks were generated $b_1 = \{r_1, r_3, r_1*, r_2*\}$, $b_2 = \{r_2, r_3*\}$, $b_3 = \{r_4, r_5*, r_6*\}$ and $b_4 = \{r_5, r_6, r_4*\}$, which generate 4, 1, 2 and 2 comparisons, respectively. Assuming that there are two nodes available, a scheduler can send randomly the blocks b_1 and b_3 to the first node and blocks b_2 and b_4 to the other node. As a result, the task will suffer from the load imbalancing problem since one node will perform six comparisons (4 + 2) while the other node will perform three (1 + 2) comparisons. To minimize the load imbalancing problem, the greedy load balancing technique proposed in [Araújo et al. 2016] is applied. This load balancing technique takes into account the amount of comparisons to be performed in each block to guide the distribution of the blocks among the nodes. To this end, the blocks are ordered according to the amount of comparisons: b_1, b_3, b_4 and b_2. Posteriorly, the top block is removed from the stack and sent to the node with fewer comparisons already allocated, applying a greedy algorithm. Therefore, the blocks b_1 and b_2 are sent to a node and the blocks b_3 and b_4 to the other node. Thus, the load imbalancing is minimized since one node will perform five comparisons (4+1) while the other node will perform four (2+2) comparisons.

5. Evaluation

In this section, we evaluate the effectiveness and efficiency of the MATCH-UPS approach using a cluster infrastructure with five nodes. Each node has 1 core, an Intel(R) Xeon(R) 1.0GHz CPU, 7GB memory, and runs the 64-bit Debian GNU/Linux OS with a 64-bit JVM and Apache Spark 2.0[2]. For the evaluation, we use real-world data sources[3] of Curitiba (Brazil) and New York (USA), provided by Open Street Map[4] and their respective municipalities. Table 2 depicts the number of geospatial records contained in each data source as well as the number of duplicate records (i.e., correspondences) present in each

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

[3]Available in the project's repository.

[4]https://www.openstreetmap.org/

Table 2. Data sources characteristics.

Pairs of Datasets	Municipality	OSM	Duplicates
Curitiba (Parks/Squares)	682	16,189	682
New York (Parks/Squares)	2,008	1,264,799	-
Curitiba (Bus Stops)	6,982	736	6,982
New York (Bus Stops)	3,365	74,140	-

pair of data sources. The data source provided by the Municipality of Curitiba was considered as the gold standard since this data source was cleaned and assessed by the Federal University of Technology (located in Curitiba) and the Institute of Research and Urban Planning of Curitiba (IPPUC[5]). However, we do not have enough information about the quality of the data source provided by the municipality of New York to consider it as the gold standard. Therefore, the cells of the New York data sources do not have value for the "Duplicates" column.

Effectiveness Results. This experiment evaluated the effectiveness of the MATCH-UPS approach over the data sources of Curitiba since there is a gold standard to validate the results. In this sense, three metrics are used to measure the effectiveness: recall, precision and F-measure. It is important to highlight the MATCH-UPS approach applies the same rule, proposed in [Fan et al. 2014], to determine whether or not a pair of geospatial records is considered a match. Therefore, the effectiveness results of the MATCH-UPS approach (without applying the blocking technique) are exactly the same as the results achieved by the approach proposed in [Fan et al. 2014].

Figures 3 (a) and (b) depict the effectiveness of the MATCH-UPS approach over the data sources that store the squares/parks and bus stops of the Curitiba. The goal of this experiment is to evaluate the impact of the blocking technique and context information on the effectiveness of the proposed approach. Regarding the squares/parks data sources, the following MATCH-UPS variations were evaluated: i) MATCH-UPS based on [Fan et al. 2014]; and ii) MATCH-UPS applying the blocking technique. Similarly, concerning the bus stops data sources, the following MATCH-UPS variations were evaluated: i) MATCH-UPS based on [Yang et al. 2014]; ii) MATCH-UPS applying the context information; and iii) MATCH-UPS combining the application of context information and blocking technique.

Based on the achieved results, it is possible to infer that the application of the blocking technique described in Section IV does not affect significantly the MATCH-UPS effectiveness. This behavior demonstrates that the blocking techniques just discarded comparisons with low chances of resulting in matches. Thus, the application of the blocking technique emerges as a useful step to perform the map matching task. Furthermore, we can highlight the 26% increase in the precision results achieved by the MATCH-UPS when the context information was applied, as shown in the Figure 3(b). It occurs due to the fact that this information assists the proposed approach to better classify the pairs of geospatial records. Concerning the recall metric depicted in Figure 3(b), the proposed approach presents low values since the intersection rate between the OSM data source and the municipality data source is only 10%, as described in [Araújo et al. 2017]. In other

[5]http://www.ippuc.org.br/

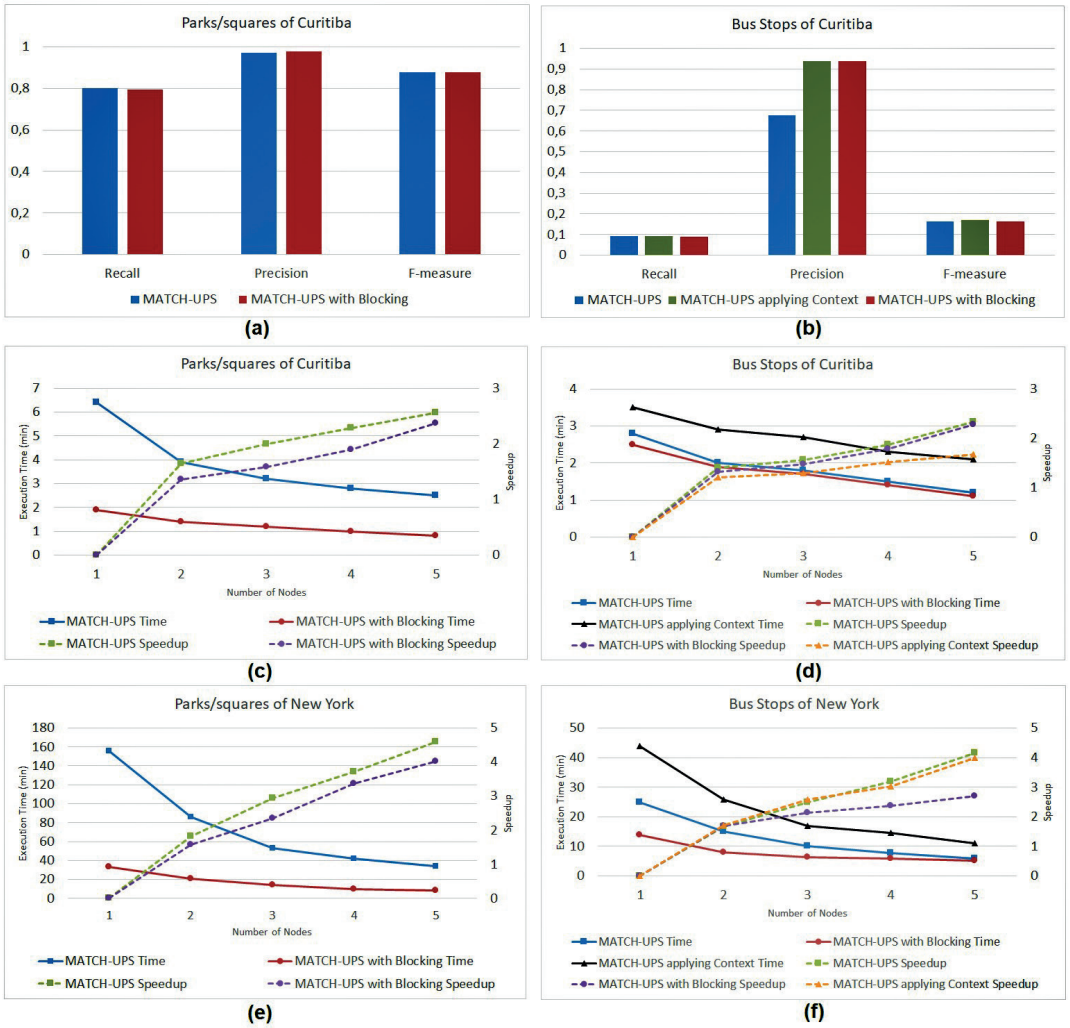


Figure 3. Evaluation of MATCH-UPS approach: (a) Effectiveness results for parks and squares of Curitiba; (b) Effectiveness results for bus stops of Curitiba; (c) Efficiency results for parks and squares of Curitiba; (d) Efficiency results for bus stops of Curitiba; (e) Efficiency results for parks and squares of New York; and (f) Efficiency results for bus stops of New York

words, the intersection between the OSM and the municipality data sources contains only 696 records (of 6,982 records stored in the municipality data source).

Efficiency Results. In this experiment, we evaluated the efficiency of the MATCH-UPS approach and its respective variations, similarly to the effectiveness experiment. It is important to notice that all variations of the MATCH-UPS approach use the load balancing technique described in Section IV. To measure the efficiency, the execution time (given by the average of three executions of each MATCH-UPS variation) and the speedup (which measures how much faster a process runs in parallel) are evaluated.

The efficiency results indicate that the application of parallel computing enhances the efficiency of the map matching task. Considering the MATCH-UPS approach (without blocking technique) for stand-alone mode (i.e., one node) as the baseline approach, it is possible to infer that the proposed Spark-based approach significantly reduces the execution time of the map matching task, as illustrated in Figures 3(c)-3(f). Although the

application of context information improves the effectiveness results, this strategy takes more time to be executed since it needs to process the context information besides to compare the geographical attributes of the records. Regarding the application of the blocking technique, it is important to highlight that the MATCH-UPS with the blocking technique achieved the best results regarding execution time for all experimental scenarios since the technique reduces the search space (i.e., number of comparisons between entities) of the map matching task, as depicted in Figures 3(c)-3(f). In Figure 3(c), with 1 node, the application of the blocking technique reduced the execution time by one third, without significant impact on the effectiveness (as shown in the Figure 3(a)).

Concerning the speedup metric, the MATCH-UPS approach achieved better results when large data sources (e.g. data source of New York) are submitted. It occurs due to the fact that Spark initialization time dominates the MATCH-UPS execution time for small (and medium) data sources [Araújo et al. 2016]. On the other hand, the speedup results of MATCH-UPS for large data sources (illustrated in Figures 3(e) and (f)) denote the scalability of the proposed approach.

6. Conclusion

This article presents MATCH-UPS, a Spark-Based approach to perform map matching in parallel. Context information and a blocking (indexing) technique are applied to enhance the effectiveness and efficiency of the approach, respectively. It is important to highlight that the MATCH-UPS approach can assist several smart cities approaches and environment projects from different countries that face common Big Data challenges, supporting the citizens and changing the way they live. Furthermore, other map matching approaches (e.g., the approaches proposed in [Fan et al. 2014, Araújo et al. 2017]) can benefit from the Spark-based workflow and the blocking technique proposed in this work to enhance the efficiency of them. Based on the experimental experiments, the results show that the MATCH-UPS applying the blocking technique improves the efficiency without significant impact on the effectiveness results. Moreover, the combination of context information and blocking technique enhances the efficiency and effectiveness results.

In future work, we intend to execute the proposed approach over other large geospatial data sources. Furthermore, we aim to extend the proposed approach in order to match line records (e.g., streets and trajectories). Another open area is to propose a map matching approach able to deal with streaming geospatial data.

References

Alarabi, L., Mokbel, M. F., and Musleh, M. (2017). St-hadoop: A mapreduce framework for spatio-temporal data. In *International Symposium on Spatial and Temporal Databases*, pages 84–104. Springer.

Araújo, T. B., Cappiello, C., Kozievitch, N. P., Mestre, D. G., Pires, C. E. S., and Vitali, M. (2017). Towards reliable data analyses for smart cities. In *Proceedings of the 21st International Database Engineering & Applications Symposium*, pages 304–308. ACM.

Araújo, T. B., Pires, C. E. S., da Nóbrega, T. P., and Nascimento, D. C. (2016). A fine-grained load balancing technique for improving partition-parallel-based ontology matching approaches. *Knowledge-Based Systems*, 111:17–26.

Arriagada, J., Gschwender, A., Munizaga, M. A., and Trépanier, M. (2019). Modeling bus bunching using massive location and fare collection data. *Journal of Intelligent Transportation Systems*, 23(4):332–344.

Bojic, I., Massaro, E., Belyi, A., Sobolevsky, S., and Ratti, C. (2015). Choosing the right home location definition method for the given dataset. In *International Conference on Social Informatics*, pages 194–208. Springer.

Bouguelia, M.-R., Karlsson, A., Pashami, S., Nowaczyk, S., and Holst, A. (2018). Mode tracking using multiple data streams. *Information Fusion*, 43:33–46.

Christen, P. (2012). *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media.

Du, H., Alechina, N., Jackson, M., and Hart, G. (2017). A method for matching crowd-sourced and authoritative geospatial data. *Transactions in GIS*, 21(2):406–427.

Fan, H., Yang, B., Zipf, A., and Rousell, A. (2016). A polygon-based approach for matching openstreetmap road networks with regional transit authority data. *International Journal of Geographical Information Science*, 30(4):748–764.

Fan, H., Zipf, A., Fu, Q., and Neis, P. (2014). Quality assessment for building footprints data on openstreetmap. *International Journal of Geographical Information Science*, 28(4):700–719.

Harb, J. G. and Becker, K. (2018). Emotion analysis of reaction to terrorism on twitter. In *Proceedings of the SBC Brazilian Symposium on Databases*, pages 97–108.

Interdonato, R. and Tagarelli, A. (2017). Personalized recommendation of points-of-interest based on multilayer local community detection. In *International Conference on Social Informatics*, pages 552–571. Springer.

Pi, X., Egge, M., Whitmore, J., Silbermann, A., and Qian, Z. S. (2018). Understanding transit system performance using avl-apc data: An analytics platform with case studies for the pittsburgh region. *Journal of Public Transportation*, 21(2):2.

Xavier, E., Ariza-López, F. J., and Ureña-Cámara, M. A. (2016). A survey of measures and methods for matching geospatial vector datasets. *ACM Computing Surveys (CSUR)*, 49(2):39.

Yang, B., Zhang, Y., and Lu, F. (2014). Geometric-based approach for integrating vgi pois and road networks. *International Journal of Geographical Information Science*, 28(1):126–147.

Zhang, C., Zhao, T., and Li, W. (2015). *Geospatial semantic web*. Springer.

Zhang, X., Ai, T., Stoter, J., and Zhao, X. (2014). Data matching of building polygons at multiple map scales improved by contextual information and relaxation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 92:147–163.

Towards a Technique for Extracting Relational Actors from Monolithic Applications

Rodrigo Laigner[1], **Sérgio Lifschitz**[1], **Marcos Kalinowski**[1], **Marcus Poggi**[1], **Marcos Antonio Vaz Salles**[2]

[1]Departamento de Informática – PUC-Rio, Brazil
{rlaigner,sergio,kalinowski,poggi}@inf.puc-rio.br

[2]Department of Computer Science (DIKU) – University of Copenhagen, Denmark
vmarcos@di.ku.dk

Abstract. *Relational actors, or reactors for short, integrate the actor model with the relational data model, providing an abstraction for enabling actor-relational database systems. However, as a novel model of computation for databases, there is no extensive work on reasoning about reactor modeling. To close this gap, this paper aims to propose as well as evaluate a technique to extract reactors from a monolithic system. For evaluation, we selected a REST-based open-source OLTP system in which a decomposition to microservices was conducted and applied our technique on its predecessor monolithic version. Our technique led to the same set of decisions, regarding table and behavior selection, taken by experts when decomposing the same system into microservices. The proposed technique can be seen as a first step towards supporting practitioners in decomposing OLTP systems into reactors.*

1. Introduction

Reactive systems and microservices constitute a new trend in the development of data-driven software applications [Boner et al. 2019, Hasselbring and Steinacker 2017]. To support these new application development needs, proposals have been put forward to integrate databases and actor systems [Bernstein et al. 2017, Shah and Salles 2017]. A key idea in these proposals is to support asynchrony and distribution in application design while still providing for desirable database functionality such as declarative querying and transactions.

In particular, actor-relational database systems aim at integrating actor constructs for encapsulation and concurrency at the level of the programming interface of a relational database management systems (RDBMS) [Shah and Salles 2017]. Relational actors (or reactors, for short) consist of a concrete programming model for actor-relational DBMS [Shah and Salles 2018]. A reactor encapsulates relations in its state, and allows for state manipulations to be executed only by asynchronous function calls returning futures. While reactors appear to be a promising programming model for actor-relational databases, there is limited guidance in how to design applications based on reactors. In [Shah and Salles 2018], the authors assert that "an interesting avenue for future research is to explore an analytical machinery for modeling and comparing the quality of reactor database designs."

Thus, there is a need for a systematic approach to derive a reactor database design. Towards this end, this work presents a technique for extracting reactors from a monolithic

application with Online Transaction Processing (OLTP) characteristics. The technique considers variables such as the access frequency of application entry points and coupling among tables. Importantly, we show that our technique can be used to suggest how to break a REST-based monolithic application into reactors.

The document is organized as follows. Section 2 provides background, presenting reactor database systems, the architectural style REST, and layered architecture. Section 3 discusses related work. Section 4 presents the technique proposed. Section 5 provides an evaluation of the technique based on an open-source OLTP application. The limitations are presented in Section 6. Finally, Section 7 presents concluding remarks.

2. Background

This section introduces background needed to contextualize our proposed technique.

Reactors. Stored procedures in RDBMS [Rowe and Stonebraker 1987] generally do not present constructs to explicitly parallelize operations on the database. This way, practitioners must rely on high level programming languages in the middle tier to handle parallelism in application logic. However, the complexity in source code entailed by the latter is substantial, raising the need for better abstractions [Sutter and Larus 2005]. In order to overcome this problem, [Shah and Salles 2018] proposed a relational actor (reactor) database system, which is derived from the actor model [Agha 1986].

In the actor model, [Shah and Salles 2018] argue that “communication is typically achieved by non-blocking send and blocking receive primitives.” In response to a received message, an actor can also send messages to other actors. Reactors abstract such messaging for concurrency and parallelism through asynchronous function calls that trigger operations on encapsulated state. At the same time, reactors provide high-level database abstractions for application programming. Importantly, “the state of each actor is abstracted by a set of relations and application functions employ declarative queries against relations” [Shah and Salles 2017]. Additionally, application functions can be executed transactionally with reactors.

An excerpt of interaction among reactors adapted from [Shah and Salles 2017] is shown in Figure 1. The figure depicts the use case of an order, triggered by the function call *add_items* in the *Cart* reactor, which transactionally retrieves customer information from a disjoint *Customer* reactor. After a separate transactional call to *checkout*, the reactor *Cart* then records a *store visit* in the same *Customer* reactor.

Representational State Transfer. Representational State Transfer (REST) is an architectural style described by Roy Fielding [Fielding 2000] aimed at designing distributed systems over the World Wide Web and has been established as a pattern in industry. According to Fielding [Fielding 2000], "the key abstraction of information in REST is a resource", on which "REST components perform actions on a resource by using a representation to capture the current or intended state of that resource". The action and the intended state of a resource can be understood as a contract established between two communicating parties.

Layered architecture. Layered architecture is a software architecture pattern aimed at distributing responsibilities over different components in an application in order to achieve high cohesion and low coupling among components [Fowler 2015]. According

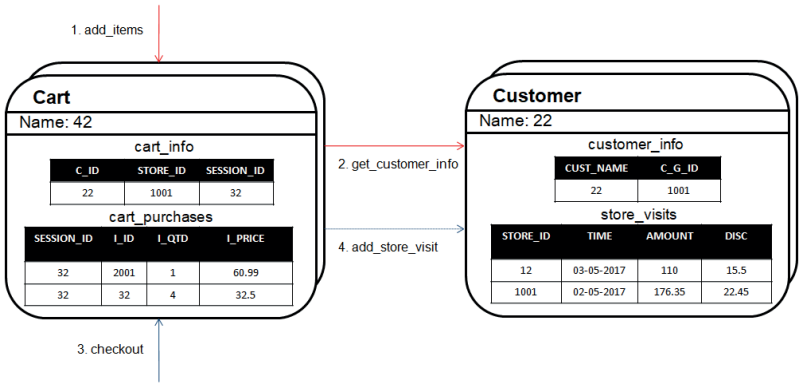


Figure 1. Communication between relational actors

to Richards [Richards 2015], "components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., business logic)". This way, a common approach for layered architecture is dividing the application into the four layers described below.

The **Presentation** layer encodes a set of rules that enable external clients to communicate with the system. This layer thus abstracts the *entry points* of the system, e.g., a REST API resource. The **Business** layer is composed of smaller subsystems. It is characterized as the core of the system, being responsible for handling application logic. The **Repository** layer is composed by a set of modules that primarily handle communication with the database system. This layer commonly deals with data access and data manipulation in SQL or through abstractions, such as provided by the Hibernate framework.[1] Finally, the **Database** layer is the DBMS.

Monolithic Applications. Monolithic systems adopt an architectural style in which modules and subsystems are integrated and cooperate in a centralized manner. By contrast, in a microservices architecture, the system is subdivided into several services, each of which usually represents a single concern within the system [Richards 2015].

3. Related Work

As there is no previous work on decomposing applications into reactors, we observe that the work most related to our proposal is on decomposing modules into thin services, the so-called microservices. As there is a substantial body of work on this topic, we focus on those studies that present similar approaches in relation to our proposed technique.

Mazlami et al. [Mazlami et al. 2017] define an extraction model where the starting point is the source code. Then, class files, change history, and developers that contributed to source code are identified. Employing different coupling strategies (e.g., terms are extracted from class files in order to enable a semantic coupling strategy), a graph representation is created representing the degree of coupling among classes in the monolithic system. A graph clustering algorithm is used to obtain candidates for microservices. The authors [Mazlami et al. 2017] argue that “the unsolved problem of how to share or assign pre-existing databases to different services remains a limitation” of their work.

[1] https://hibernate.org

Gysel et al. [Gysel et al. 2016] propose a service decomposition based on a set of coupling criteria and system specification artifacts, such as domain models and use cases. A supporting tool framework (Service Cutter) was developed and decomposed services were “represented as an undirected, weighted graph to find and score”. For instance, the authors [Gysel et al. 2016] define System Specification Artifacts, such as use cases and domain models, as an input of Service Cutter. We consider these artifacts insufficient to assess application workload, particularly in the context of OLTP applications. In contrast to our approach, their technique [Gysel et al. 2016] does not present primitives for considering the workload of the application neither extraction of application logic.

Levcovitz et al. [Levcovitz et al. 2016] present a technique for decomposing microservices from a monolithic application that is organized into three parts: a client side user interface, a server side application, and a database. The work of Levcovitz et al. [Levcovitz et al. 2016] decomposes the application based on business areas, which they describe as being "responsible for a business process". We consider this an inadequate selection criterion since a business unit is an abstract concept and might not lead to optimal decomposition, as our work does, in regard to workload and behavior distribution.

In addition to the above, numerous approaches have investigated horizontal and vertical partitioning in object-oriented [Bellatreche et al. 2000] or relational databases [Pavlo et al. 2012]. In contrast to our proposal, these approaches focus exclusively on database access logic and do not consider layered architectures with application code across tiers. Lastly, Wang et al. [Wang et al. 2019] discuss the modeling of Internet-of-Things applications with actor-oriented databases. However, the work focuses on actor databases with an object-oriented data abstraction, being thus only partially applicable to relational actors. Furthermore, the modeling guidelines presented are aimed at the development of new applications, as opposed to decomposition of existing monoliths.

4. Proposal

This work aims to extract reactors from a monolithic system. Therefore, we rely on the concept of *entry points* of a system, which are contracts that external clients should follow in order to communicate with the application. Several approaches, such as RPC and SOAP, have been defined to enable distributed systems communication. Thus, in order to establish a common pattern for systems communication, our technique targets OLTP web applications that communicate through REST APIs.

To formalize the system under analysis, we extend the system formalization of Levcovitz et al. [Levcovitz et al. 2016]. A monolithic system S is represented by a quadruple (I;B;R;D), where I = { int_1, int_2, ..., int_n } is a set of interfaces (REST resources) from the presentation layer, B = { bf_1, bf_2, ..., bf_n } is a set of business functions, R = { rf_1, rf_2, ..., rf_n } is a set of repository functions, and T = { tb_1, tb_2, ..., tb_n } is a set of database tables. Figure 2 depicts a representation of the layered architecture considered in this work. Our technique for extracting reactors from a monolithic application comprises five phases, which are described as follows.

System dependency graph construction. This phase comprises building a dependency graph G = (V, E) representing the execution flow for each interface described in the presentation layer. An execution flow represents the branches associated with a given interface, where vertices represent available functions and resources.

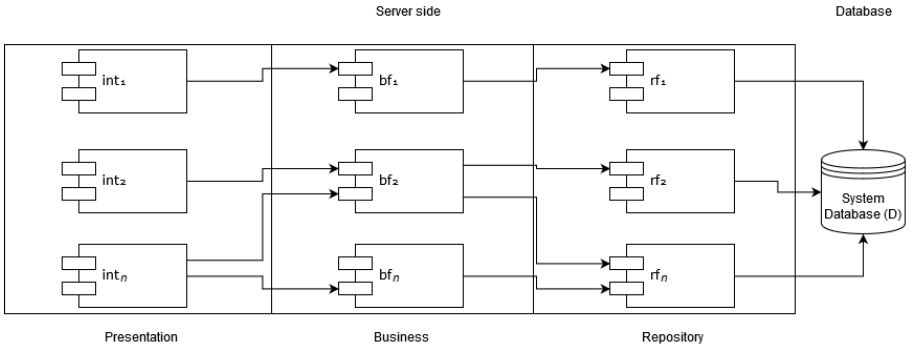


Figure 2. Layered architecture considered in this work

Vertices are represented by the following elements: (i) interfaces ($int_i \in$ I) $\forall i$; business functions ($bf_i \in$ B) $\forall i$; repository functions ($rf_i \in$ R) $\forall i$; database tables ($tb_i \in$ T) $\forall i$. Furthermore, the edges represent: (i) interface calls to business functions; (ii) calls among business functions; (iii) calls from business functions to repository functions; (iv) calls among repository functions; and (v) table accesses from repository functions.

Profile data collection. After building the dependency graph, it is important to know how frequently each REST resource (interface) is called. Profiling can be achieved by introducing dynamic instrumentation in the application or by techniques such as aspect-oriented programming, in a non-intrusive manner [Kiczales et al. 1997]. When such profiling data are not available, we suggest reasoning about the resources and assigning weights related to their expected frequency. Based on this step, vertices representing interfaces are weighted by a frequency cost, the so-called *access frequency*.

Table coupling identification. In the context of software development, coupling is commonly used for assessing source code structural quality [Olbrich et al. 2010]. On the other hand, since relational actors operate on data, it is necessary to define coupling in the context of tables.

Table coupling, as used in this work, thus concerns associations between tables. Two tables are associated by a foreign key (FK) if a FK is present in one or both relations. In case of a many-to-many association, we assume the FK is present on both relations and discard the join table built in consequence of physical design. A formalization of the information on table coupling is provided below.

$$coup_{i,j} = \begin{cases} 1, & \text{if tables } i \text{ and } j \text{ are associated} \\ 0, & \text{otherwise} \end{cases} \qquad \forall (i,j) \in T \times T$$

Reactor table identification. Towards identifying tables that would compose a reactor, the information collected in previous steps is employed to identify an optimal distribution of tables and interfaces among clusters. The formulation is based on the clustering problem found in optimization studies [Du and Pardalos 1998], in which the goal is to identify a set of clusters that respect a given connectivity measure among vertices.

It is necessary to establish a limit on the amount of workload a reactor can sustain while properties, such as coupling, are maintained within reactors. Given the access frequency, the table coupling, and the maximum workload limit, our technique aims to

divide workload among clusters by properly allocating tables and interfaces to reactors. It is noteworthy that this work only considers interfaces that enable GET and POST operations. The parameters and the decision variables are defined as follows.

Parameters

$access_i$ access frequency for interface i;
$coup_{i,j}$ coupling degree between tables i and j;
Q the maximum access frequency load a reactor can sustain.

$$post_{i,j} = \begin{cases} 1, & \text{if POST for table } j \text{ is fulfilled through interface } i \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in I,\ \forall j \in T$$

$$get_{i,j} = \begin{cases} 1, & \text{if GET for table } j \text{ is fulfilled through interface } i \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in I,\ \forall j \in T$$

Decision variables

$tb_{k,i}$ equals 1 if table *i* is allocated to reactor type *k*, and 0 otherwise;
$int_{k,i}$ equals 1 if interface *i* is allocated to reactor type *k*, and 0 otherwise.

Based on the parameters and decision variables, we introduce the model designed to optimize the definition of reactors.

$$\max \quad \sum_{k=1}^{n} \omega_k \tag{1}$$

The objective function (1) maximizes the coupling level among tables allocated within each cluster. In order words, we aim to keep associated tables together as maximum as possible in order to reduce communication costs in case of JOIN operations.

$$\sum_{k=1}^{n} tb_{k,i} = 1 \qquad \forall i \in T \tag{2}$$

$$\sum_{k=1}^{n} int_{k,i} = 1 \qquad \forall i \in I \tag{3}$$

Constraints (2) and (3) limit the maximum number of clusters each table and interface are allowed to be allocated to, respectively. We aim to allocate a given table (or interface) to only one cluster.

$$int_{k,i} - tb_{k,j} = 0 \qquad where \quad post_{i,j} = 1 \quad \forall k = 1, ..., n \tag{4}$$

$$int_{k,i} - tb_{k,j} = 0 \qquad where \quad get_{i,j} = 1 \quad \forall k = 1, ..., n \tag{5}$$

$$\sum_{k=1}^{n} \sum_{i \in I} \sum_{j \in T} int_{k,i} + tb_{k,j} = 2 \qquad where \quad post_{i,j} = 1 \tag{6}$$

$$\sum_{k=1}^{n} \sum_{i \in I} \sum_{j \in T} int_{k,i} + tb_{k,j} = 2 \qquad where \quad get_{i,j} = 1 \tag{7}$$

Constraints (4)-(7) force the table that represents a resource to be allocated in the same cluster that its respective GET and POST operations are allocated to (through respective interfaces). In addition, constraints (4)-(7) force GET and POST interfaces to be allocated in the same cluster.

$$\alpha_k = \sum_{i \in T} tb_{k,i} \qquad \forall k = 1, ..., n \tag{8}$$

$$\omega_k = \sum_{i \in T} \sum_{j \in T/i} tb_{k,i}.tb_{k,j}.coup_{i,j} \qquad \forall k = 1, ..., n \tag{9}$$

$$\alpha_k \leq \omega_k + 1 \qquad \forall k = 1, ..., n \tag{10}$$

$$\sum_{i \in I} int_{k,i}.access_i \leq Q \qquad \forall k = 1, ..., n \tag{11}$$

$$\sum_{i \in I} int_{k,i} \geq \sum_{j \in T} tb_{k,j} \qquad \forall k = 1, ..., n \tag{12}$$

Constraints (8) represent the total number of tables a given cluster holds. Constraints (9) represent the level of coupling a given cluster holds. Constraints (10) force tables in a cluster to have a positive coupling level. In other words, if there is more than a table in the cluster, the tables must be associated through FK. Constraints (11) limit the maximum access frequency a cluster can sustain. Constraints (12) restrict the existence of a cluster with tables and no interfaces.

$$int_{k,i} \in \{0, 1\} \tag{13}$$

$$tb_{k,i} \in \{0, 1\} \tag{14}$$

Constraints (13-14) are integrality constraints.

Reactor function extraction. In order to identify application logic that would be more efficiently executed by the DBMS, Cheung et al. [Cheung et al. 2012] assert that "programmers must identify sections of code that make multiple (or large) database accesses and can be parameterized by relatively small amounts of input". Based on this observation, this step aims at identifying source code lines with: (*i*) high degree of data access and manipulation, and (*ii*) low complexity. Thus, we seek the identification of application logic in the business layer to be migrated to a reactor function (*RAF*). Equation 15 exhibits the formula for detecting methods from source code for migration.

$$RAF(M) = \begin{cases} 1, \text{ CYCLE}(M) < HIGH_1 \wedge \text{NOAV}(M) < MANY \wedge \\ \qquad \text{DDLOC}(M) \geq HIGH_2 \\ 0, \text{otherwise} \end{cases} \tag{15}$$

In the equation above, *M* is the business function being inspected; CYCLE(*M*) is the cyclomatic complexity of *M*; NOAV(*M*) is the Number of Accessed Variables of *M*; DDLOC(*M*) is the degree of Data-Driven Lines of Code (DDLOC) in *M*, which are those lines of code that make access to and manipulation of data.

As mentioned by Cheung et al. [Cheung et al. 2012], "identifying sections of application logic that are good candidates for conversion [...] is tricky". Thus, we suggest that the thresholds $HIGH_1$, $HIGH_2$, and $MANY$ must be adapted for each application,

taking into consideration characteristics such as average lines of code (LOC) of methods present in each module of the business layer.

Reactor function allocation. Once reactor functions are identified, based on the locality of their source methods in the dependency graph, it is possible to assign a function that operates on a given table to the cluster holding it. For example, if table tb_x is assigned to reactor type R_k and method M_y manipulates tb_x, then M_y is allocated to R_k. In case of a method manipulating multiple tables, then the method is assigned to the reactor type with the respective highest *access frequency* for the given interface.

5. Evaluation

Our technique is applied to the monolithic version of Petclinic,[2] an OLTP open-source demonstration project of the Spring Framework.[3] The system adopts a three-tier layered architecture and has been under development since 2016.

System dependency graph construction. Due to space constraints, we provide a partial representation of the dependency graph for Petclinic. Figure 3 depicts the execution path for the */visits/new* resource regarding a POST operation. A node is represented by a rectangle, in which the header is the class name followed by method name. The complete dependency graph for Petclinic can be accessed online.[4]

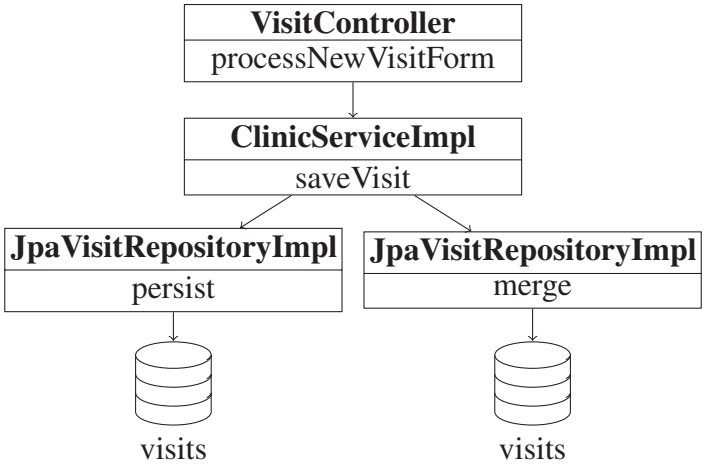


Figure 3. Dependency graph of */visits/new* interface POST operation in Petclinic

Profile data collection phase. Since Petclinic is a demonstration project, this study relies on an artificial workload that aims at reproducing a real-world scenario for the Petclinic domain. The workload decisions were taken based on characteristics of Petclinic, e.g., the insertion rate into the *pets* table cannot be lower than that of owners (a *pet* cannot exist without an *owner*), and table *visits* must incur the highest access frequency. Note that access frequency was normalized to range from 1 to 100 per interface.

Table coupling identification. The entity-relationship (ER) diagram for the Petclinic application is depicted in Figure 4. As can be seen, the relationships with coupling equal to 1 are: types and pets, owners and pets, and visits and pets.

[2]https://github.com/spring-petclinic/spring-framework-petclinic
[3]https://spring.io
[4]https://zenodo.org/record/3237968

Operation	Interface	Table	Access frequency
GET	/owners/ownerId	owners	10
GET	/owners/ownerId/edit	owners	10
PUT	/owners/ownerId/edit	owners	10
GET	/owners	owners	60
POST	/owners/new	owners	20
GET	/owners/ownerId/pets/new	pets	25
POST	/owners/ownerId/pets/new	pets	25
GET	/owners/ownerId/pets/petId/edit	pets	10
PUT	/owners/ownerId/pets/petId/edit	pets	10
GET	/vets	vets	10
GET	/owners/ownerId/pets/petId/visits/new	visits	100
POST	/owners/ownerId/pets/petId/visits/new	visits	100

Table 1. Workload scenario

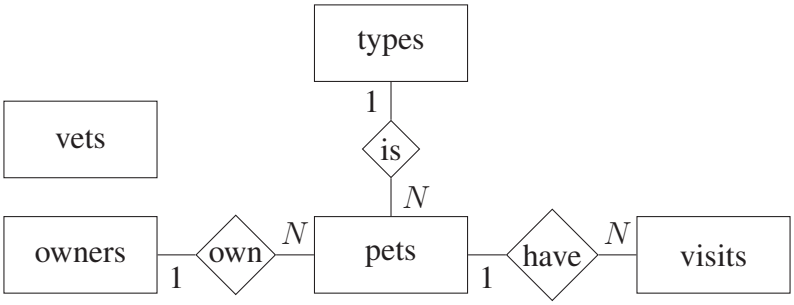


Figure 4. Petclinic ER diagram

Reactor table identification. In order to execute the model, the parameter Q was set to 200, corresponding to the sum for the resource with the highest access frequencies (*visits*). We aim to distribute workload among reactors, avoiding two or more data-intensive entry points to be allocated to the same reactor type. Figure 5 exhibits the result of the optimization allocating tables to clusters. The result of the allocation of interfaces to clusters can be accessed online.[4]

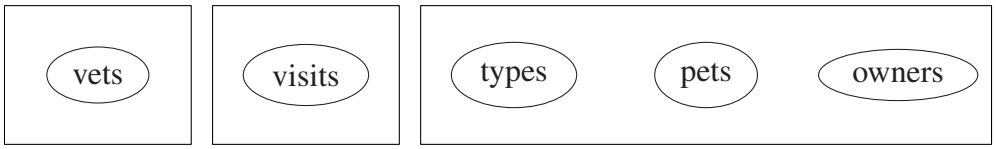


Figure 5. Optimization problem output

Reactor function extraction. For this step, a source code inspection was performed in Petclinic. Based on the strategy discussed in Section 4, we have employed a heuristic to define the thresholds: As the average Petclinic business layer LOC is 1, for each 3 lines of code, $HIGH_1$ is incremented. Also, as the average LOC of the repository layer in Petclinic is 2, DDLOC must be greater than or equal 1. An excerpt of an extracted reactor function based on the resource */visits/new* is shown in Figure 6. The remaining functions are available online.[4]

```
@Service
public class ClinicServiceImpl implements ClinicService {
   // code ommitted for brevity
   private VisitRepository visitRepository;
   // code ommitted for brevity
   @Override
   @Transactional
   public void saveVisit(Visit visit) throws
         DataAccessException {
      visitRepository.save(visit);
   }
}
```

```
@Repository
public class JpaVisitRepositoryImpl implements
      VisitRepository {
   // code ommitted for brevity
   @Override
   public void save(Visit visit) {
      if (visit.getId() == null) {
         this.em.persist(visit);
      } else {
         this.em.merge(visit);
      }
   }
}
```

```
void upsert_visit(visit){
   if visit.id IS NULL then
      INSERT INTO
               visits
      VALUES   (visit.date,
               visit.description,
               visit.pet_id);
      return;
   end if;

   SELECT id
   FROM visits
   INTO v_id
   WHERE visit.id = id;

   if v_id IS NULL then
      abort;
   end if;

   UPDATE visits
   SET date = visit.date,
      description = visit.description,
      pet_id = visit.pet_id
   WHERE visit.id = id;
}
```

Figure 6. Application logic (left) extracted to a reactor function (right)

Reactor function allocation. Following the technique, the method depicted in Figure 6, for example, is assigned to the cluster that holds the *visits* table. Information regarding the allocation of all identified methods can be accessed online.[4]

Based on the results, it is possible to correlate the distribution of relational actors and the respective tables and interfaces to the project Petclinic microservices version.[5] Table 2 depicts the tables presented in each microservice in the Petclinic microservices application. We observe that the decomposition of tables provided by the expert developers of Petclinic microservices is the same as the decomposition provided by our technique in terms of the tables and methods selected for each microservice.

Microservice	Tables
customers-service	owners, types, and pets
vets-service	vets
visits-service	visits

Table 2. Tables presented in each microservice for Petclinic

6. Limitations

One of the limitations of our technique concerns the assumption that the system adopts a three-tiered layered REST-based architecture. However, this architecture is widely adopted in industrial settings. Also, while our technique currently only considers interfaces that enable GET and POST operations, we do not anticipate significant issues in extending it with DELETE and PUT operations.

Regarding the evaluation, the prepared artificial workload may not provide sufficient coverage for all cases in which the technique could be applied. Nevertheless, the workload was defined based on reasoning about the application domain. It is noteworthy to mention that the workload, its limits, and the source code metrics were verified by three independent researchers. Additionally, to test the sensitivity of our model, we have also

[5]https://github.com/spring-petclinic/spring-petclinic-microservices

applied it to a different hypothetical workload, allowing us to observe sensitivity of model output due to changes in the input specifications.

Finally, we chose a specific Java software project for applying our technique. However, we believe the technique is generic enough to be applied to other object-oriented programming languages (e.g., C#) and frameworks (e.g., .NET Core).

7. Concluding Remarks

This study proposes an exact optimal approach for allocating relational tables, REST interfaces, and application logic to clusters, represented by reactors. To the best of our knowledge, there is no identical work in the literature. Although our system formalization is based on the work of Levcovitz et al. [Levcovitz et al. 2016], it goes much beyond in both presenting a MIP-solver-based automatic method for distribution among clusters as well as considering heuristics to identify application logic in source code to be extracted.

For evaluation purposes, the technique was applied to a REST-based application and exhibited appropriate results. Indeed, the output yielded by our technique consisted in a closely related reactor decomposition when compared to a microservices decomposition conducted by the developers of the original application example.

It is noteworthy to mention that modules in larger applications might require different workload constraints. For instance, a module that presents heavily accessed REST endpoints might need a higher workload threshold, when compared to those modules that are composed by REST endpoints that are infrequently accessed. This observation calls for not using a single parameter Q. Therefore, for large applications, we argue that the restriction over the maximum workload per cluster should be defined per use case. For instance, the maximum workload allowed in a cluster that primarily deals with payment transactions (due to intensive insert rate) should be higher than a cluster that handles historical analysis of user behavior (infrequent query-based workload).

Thus, as future work, we aim at conducting more in-depth empirical evaluations of our technique to further understand its benefits and limitations, including applying it to real-world systems.

References

Agha, G. (1986). *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA, USA.

Bellatreche, L., Karlapalem, K., and Simonet, A. (2000). Algorithms and support for horizontal class partitioning in object-oriented databases. *Distributed and Parallel Databases*, 8(2):155–179.

Bernstein, P. A., Dashti, M., Kiefer, T., and Maier, D. (2017). Indexing in an actor-oriented database. In *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*.

Boner, J., Farley, D., Kuhn, R., and Thompson, M. (2019). The Reactive Manifesto. `https://www.reactivemanifesto.org/`.

Cheung, A., Madden, S., Arden, O., and Myers, A. C. (2012). Automatic partitioning of database applications. *Proceedings of the VLDB Endowment*, 5(11).

Du, D.-Z. and Pardalos, P. (1998). *Handbook of Combinatorial Optimization. Combinatoral Optimization in Clustering*. Springer, New York, NY.

Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine.

Fowler, M. (2015). Presentation domain data layering. `https://martinfowler.com/bliki/PresentationDomainDataLayering.html`.

Gysel, M., Kölbener, L., Giersche, W., and Zimmermann, O. (2016). Service cutter: A systematic approach to service decomposition. In *Service-Oriented and Cloud Computing*, pages 185–200. Springer International Publishing.

Hasselbring, W. and Steinacker, G. (2017). Microservice architectures for scalability, agility and reliability in e-commerce. In *IEEE ICSA Workshops*, pages 243–246.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., and Irwin, J. (1997). Aspect-oriented programming. In *ECOOP'97 — Object-Oriented Programming*, pages 220–242, Berlin, Heidelberg. Springer Berlin Heidelberg.

Levcovitz, A., Terra, R., and Valente, M. T. (2016). Towards a technique for extracting microservices from monolithic enterprise systems. *CoRR*, abs/1605.03175.

Mazlami, G., Cito, J., and Leitner, P. (2017). Extraction of microservices from monolithic software architectures. In *International Conference on Web Services*. IEEE.

Olbrich, S. M., Cruzes, D. S., and Sjøberg, D. I. (2010). Are all code smells harmful? a study of god classes and brain classes in the evolution of three open source systems. *IEEE International Conference on Software Maintenance*.

Pavlo, A., Curino, C., and Zdonik, S. B. (2012). Skew-aware automatic database partitioning in shared-nothing, parallel OLTP systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 61–72.

Richards, M. (2015). *Software Architecture Patterns*. O'Reilly, 1st edition.

Rowe, L. A. and Stonebraker, M. (1987). The POSTGRES data model. In *VLDB'87, Proceedings of 13th International Conference on Very Large Data Bases, September 1-4, 1987, Brighton, England*, pages 83–96.

Shah, V. and Salles, M. A. V. (2018). Reactors: A case for predictable, virtualized actor database systems. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD, Houston, TX, USA, June 10-15, 2018*, pages 259–274.

Shah, V. and Salles, M. V. (2017). Actor database systems: A manifesto. *CoRR*, abs/1707.06507.

Sutter, H. and Larus, J. R. (2005). Software and the concurrency revolution. *ACM Queue*, 3(7):54–62.

Wang, Y., dos Reis, J. C., Borggren, K. M., Salles, M. A. V., Medeiros, C. B., and Zhou, Y. (2019). Modeling and building iot data platforms with actor-oriented databases. In *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*, pages 512–523.

Classificação de Estados Epilépticos em Sinais de EEG Utilizando Detecção de Anomalias

Lucas Cabral[1], **Guilherme A. Barreto**[2], **José Maria Monteiro**[1]

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

[2]Departamento de Engenharia de Teleinformática – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brazil

lucascabral@alu.ufc.br, gbarreto@ufc.br, monteiro@dc.ufc.br

Abstract. *Epilepsy is a neurological disorder characterized by an abnormal electrical disturbance in the brain, causing recurrent seizures. The most commonly used exam to diagnose epilepsy it the electroencephalogram (EEG), where a patient's brain electrical activity is measured and visually analyzed. However, identifying epileptic patterns in the EEG signal through visual inspection is a time-consuming and exhaustive task for professionals in the field, motivating the development of algorithms that can identify these patterns, aiding the medical diagnosis. In this work, we propose three models based on anomaly detection. The results obtained demonstrate high performance and noise robustness in relation to results found in the literature.*

Resumo. *A epilepsia é um distúrbio neurológico caracterizado por uma perturbação elétrica anormal no cérebro, causando convulsões recorrentes. O exame mais utilizado no diagnóstico da epilepsia é eletroencefalograma (EEG), onde a atividade elétrica cerebral de um paciente é mensurada e analisada visualmente. Contudo, identificar os padrões epilépticos no sinal de EEG através de inspeção visual é uma tarefa demorada e exaustiva para profissionais da área. Assim, o desenvolvimento de algoritmos que possam identificar esses padrões de forma automática, auxiliando o diagnóstico médico, tornou-se um importante desafio. Neste trabalho, propomos três modelos de classificação, baseados em detecção de anomalias. Os resultados obtidos demonstram alto desempenho e robustez a ruídos em relação resultados encontrados na literatura.*

1. Introdução

A epilepsia é uma desordem crônica do cérebro que afeta cerca de 50 milhões de pessoas ao redor do mundo, sendo um dos distúrbios neurológicos mais frequentes [World Health Organization 2017]. Ela é caracterizada por crises recorrentes onde ocorre uma disfunção temporária da atividade elétrica dos neurônios, onde os sintomas de cada crise dependerão das partes do cérebro envolvidas na disfunção [Kanashiro 2006]. A crise convulsiva é a forma mais conhecida e é identificada comumente como "ataque epiléptico". Nesse tipo de crise, o portador de epilepsia pode cair ao chão, apresentar contrações musculares em todo o corpo, perda de consciência, mordedura da língua, salivação intensa, respiração ofegante, perda do controle da função intestinal e da bexiga.

Portadores de epilepsia tendem a ter risco sofrer fraturas e contusões de lesões relacionadas a convulsões, bem como taxas mais altas de condições psicológicas como ansiedade e depressão. Tais pacientes apresentam um risco de morte prematura cerca de três vezes maior que a população em geral.

Uma importante ferramenta para o diagnóstico e a investigação da epilepsia é o eletroencefalograma (EEG) de escalpo, que consiste no registro da atividade elétrica cerebral captada por eletrodos posicionados no couro cabeludo. O EEG permite visualizar os comportamentos anormais característicos dos diversos tipos de crises e síndromes epilépticas, sendo o método mais específico para o diagnóstico das epilepsias [Niedermeyer and da Silva 2001]. Embora técnicas mais recentes de neuroimagem tenham trazido grandes avanços, o EEG continua sendo bastante utilizado, devido a sua simplicidade técnica e ao seu baixo custo.

Contudo, a atividade cerebral do ser humano apresenta uma infinidade de padrões de atividade, caracterizados pela soma de diferentes estados mentais em um instante de tempo, resultando em um sinal naturalmente complexo e não-linear [Adeli and Ghosh-Dastidar 2010]. Ademais, os exames de EEG geram uma enorme quantidade de dados, onde a duração de uma crise convulsiva é curta em relação a duração total do exame. Em exames com horas de duração os períodos de crise podem durar apenas alguns minutos ou segundos. Neurologistas precisam analisar centenas de horas de exame buscando esses padrões. Assim, o diagnóstico da epilepsia é um procedimento lento e sujeito a falhas, devido à fadiga visual e à subjetividade de cada especialista.

Neste contexto, algoritmos que classifiquem de forma automática os padrões relacionados à uma crise convulsiva são extremamente importantes para auxiliar o diagnóstico da epilepsia e facilitar o trabalho dos neurologistas. Porém, para que os algoritmos de classificação por aprendizado supervisionado apresentem um bom desempenho, o número de exemplos por classe deve ser balanceado [Webb 2002]. No caso dos sinais de EEG, existe um grande desbalanceamento entre a quantidade de dados referentes a crises convulsivas e dados associados a comportamentos não-convulsivos. Essas condições motivam o uso de técnicas de detecção de anomalias por aprendizado não-supervisionado, com a finalidade de classificar sinais EEG em convulsivo e não-convulsivo.

Neste trabalho, propomos e avaliamos o desempenho de três modelos de classificação de estados convulsivos em EEG baseados na detecção de anomalias e utilizando atributos no domínio da frequência de Densidade Espectral de Potência (PSD). Os resultados mostram um desempenho semelhante aos melhores resultados encontrados na literatura, possuindo robustez a sinais ruidosos, além de simplicidade de treino e baixo custo computacional.

2. Trabalhos Relacionados

São encontradas na literatura diferentes técnicas de classificação utilizando aprendizado supervisionado e diferentes métodos de extração de atributos com o objetivo de classificar corretamente os sinais de EEG em convulsivo e não-convulsivo. Não existe um método padrão de extração de atributos para sinais de EEG, e a combinação efetiva de atributos e classificador ainda é um problema em aberto. A maioria dos trabalhos encontrados na literatura é paciente-específico, ou seja, os modelos são treinados e testados com dados de um paciente por vez, devido a alta variabilidade de atividade cerebral en-

tre pacientes. Subasi and Ercebeli 2005 obtiveram taxas de acerto em torno de 89,3 à 93% utilizando redes neurais artificiais e regressão logística. Chan et al. 2008 apresentaram uma sensibilidade na classificação em torno de 84,89 à 94% utilizando a extração de atributos no domínio da frequência e o classificador do tipo *Support Vector Machine* (SVM). Khan et al. 2012 usaram extração de atributos com Wavelet, produzindo acurácia de 91.8%, sensibilidade de 83.6% e especificidade de 100%. Acharya et al. 2011 utilizaram *Recurrence Quantification Analysis* (RQA) como método de extração de atributos e testou com sete diferentes classificadores: SVM, *Gaussian Mixture Model* (GMM), *Fuzzy Sugeno Classifier*, *K-Nearest Neighbor* (KNN), *Naive Bayes Classifier* (NBC), *Decision Tree* (DT), e *Radial Basis Probabilistic Neural Network* (RBPNN). Os melhores resultados foram obtidos com SVM, com acurácia de 96%, sensibilidade de 96% e especificidade de 97%. Shoeb and Guttag 2010 usaram decomposição Wavelet multi-nível para extrair atributos que capturam a morfologia e distribuição espacial do EEG, classificando com SVM e obtendo acurácia de 96%.

Fergus et al. 2016 apresentaram um método para generalizar a detecção de crises convulsivas entre diferente pacientes, uma tarefe de maior dificuldade. Os vetores de atributos em seu trabalho foram gerados com os dados de 23 pacientes, usando *Peak Frequency*, *Median Frequency*, variância, *root mean squares* (RMS), *sample entropy*, assimetria e curtose. Foi apresentado um resultado de 88% de acurácia, 88% de sensibilidade e 93% de especificidade, utilizando um classificador KNN. Ainda considerando o problema de classificação inter-pacientes, Liang et al. 2019 apresentaram uma abordagem baseada em aprendizado profundo, utilizando uma combinação das redes neurais Convolutional Neural Network (CNN) e *Long Short-Term Memory Network* (LSTM), denominada *Long-Term Convolutional Network* (LTCN). Uma das vantagens da LTCN é o fato de não ser necessária uma etapa de manual de extração de atributos, pois esta trabalha diretamente com o sinal. Essa proposta apresentou acurácia de 84%, sensibilidade de 99% e especificidade de 99%.

3. Modelos Propostos

A detecção de anomalias refere-se ao problema de encontrar dados que não estejam de acordo com o comportamento esperado. A detecção de anomalias encontra uso extensivo em uma ampla variedade de aplicações, como detecção de fraude para cartões de crédito, seguros ou cuidados de saúde, detecção de intrusão para cibersegurança, detecção de falhas em sistemas e diagnóstico de patologias [Varun Chandola 2009]. Em geral, em métodos de detecção de anomalias calcula-se uma medida de dissimilaridade entre os dados e as anomalias são encontradas através da comparação dessa medida com um limiar de decisão.

Neste trabalho, propomos três modelos de detecção de anomalias para detectar sinais convulsivos: Distâncias Ao Centroide Não-Convulsivo (DCNC), Distâncias Ao Centroide Convulsivo (DCC) e Distâncias aos Centroides de K-*Clusters* (DCKC).

3.1. Modelo das Distâncias Ao Centroide Não-Convulsivo (DCNC)

Uma medida de dissimilaridade bastante utilizada é a distância euclidiana ao centroide da massa de dados. Usualmente, um vetor com um valor alto para a distância ao centroide é considerado uma anomalia [Knorr 2000]. O valor do limiar de decisão usualmente é escolhido como o 95 ou 99 percentil das distâncias ao centroide [A. Zimek and Kriegel 2012].

O modelo das Distâncias Ao Centroide Não-Convulsivo utiliza apenas os dados não-convulsivos para definir o comportamento esperado, calculando o centroide da massa de dados não-convulsivos e calculando um vetor de distâncias de cada observação para o centroide. O limiar de decisão pode então ser escolhido como o 95 percentil desse vetor de distâncias. Esse modelo parte da suposição de que os dados possuem uma distribuição normal em torno do centroide, e pontos muitos distantes deste centroide são anomalias. Devido a alta dimensionalidade dos vetores, a visualização de sua distribuição espacial é complexa, sendo necessário testar a hipótese de sua distribuição normal. O Algoritmo 1 detalha esse método.

Algoritmo 1: Modelo de Distâncias ao Centroide Não-Convulsivo

```
   Entrada: matrizDeTreino, matrizDeTeste
   Saída: vetorDePredições
1  início
2      vetorCentroide = media(matrizDeTreino);
3      //Calcula o vetor médio dos dados de treino;
4      //Esses dados são formados apenas por sinais não-convulsivos;
5      repita
6          distanciasDeTreino[i] = distancia(matrizDeTreino[i],centroide);
7          //Para cada vetor de treino, calcula a distância até o centroide;
8      até i = quantidadeVetoresDeTreino;
9      limiar = percentil(distanciasDeTreino,95);
10     //Calcula o limiar de decisão como o 95-percentil das distâncias;
11     repita
12         distanciasDeTeste[i] = distancia(matrizDeTreino[i],centroide);
13         //Os vetores de teste são formados por todos os vetores convulsivos;
14         //e por alguns vetores não-convulsivos;
15         //Para cada vetor de teste, calcula a distância até o centroide;
16         //e compara com o limiar;
17         if distanciasDeTeste[i] > limiar then
18             vetorDePredições[i] = -1;
19             //Se a distância é maior que o limiar, classifica como convulsivo;
20         else
21             vetorDePredições[i] = +1;
22         //se a distância é menor que o limiar, classifica como normal;
23     até i = quantidadeVetoresDeTeste;
```

3.2. Modelo das Distâncias Ao Centroide Convulsivo (DCC)

O Modelo das Distâncias Ao Centroide Convulsivo (DCC) tem uma abordagem similar ao modelo anterior, porém utiliza apenas os dados convulsivos para modelar o comportamento esperado, calculando o centroide convulsivo e identificando os pontos que estejam muito distantes como não pertencentes ao conjunto de dados. Ou seja, treina-se o modelo utilizando os vetores convulsivos, calculando o limiar como o 95-percentil das distâncias ao centroide convulsivo e classificando como não-convulsivos os vetores cuja distância ao centróide está acima deste limiar. Conceitualmente, pode-se entender esse modelo como uma detecção de novidades. O algoritmo desse modelo segue passos idênticos ao Algoritmo 1, com duas importantes diferenças: os dados de treino são formados exclusivamente por dados convulsivos e o critério de classificação é invertido: as distâncias abaixo do limiar são classificadas como dados convulsivos e as acimas do limiar como não-convulsivos.

3.3. Modelo da Distâncias aos Centroides de K-*Clusters* (DCKC)

Métodos baseados na distância ao centroide têm bom desempenho na identificação de anomalias aleatoriamente distribuídas em torno da massa principal de dados. Entretanto, quando as anomalias estão concentradas, ou seja, se apresentam na forma de um ou mais agrupamentos localizados regiões específica, a distância ao centroide não os detecta satisfatoriamente. Visando tornar a detecção mais robusta pode-se utilizar técnicas de agrupamento (*clustering*). Essas técnicas comumente são desenvolvidas a partir da suposição de que instâncias normais dos dados estão â pequenas distâncias do centroide do *cluster* mais próximo, enquanto anomalias estão a grandes distâncias do centroide do *cluster* mais próximo.

O Modelo da Distâncias aos Centroides de K-*Clusters* (DCKC) utiliza uma abordagem de agrupamento. Considerando os dados não-convulsivos, define-se uma quantidade k de *clusters*, cujos centroides são calculados através dos algoritmo K-Médias [MacQueen 1967]. A quantidade ótima de *clusters* é estimada através do índice Davies-Bouldin [Davies and Bouldin 1979]. A menor distância de cada observação aos centroides é calculada, formando um vetor de distâncias a partir do qual pode ser escolhido o limiar de decisão.

Algoritmo 2: Modelo da Distâncias aos Centroides de K-*Clusters*

```
   Entrada: matrizDeTreino, matrizDeTeste
   Saída: vetorDePredições
1  início
2      K = DaviesBouldin(matrizDeTreino);
3      //Estima a quantidade ótima de clusters dos dados de treino;
4      //Esses dados são formados apenas por sinais não-convulsivos;
5      matrizCentroides = KMedias(matrizDeTreino,K);
6      //Calcula os centroides dos K clusters;
7      repita
8          distanciasDeTreino[i] = menorDistancia(matrizDeTreino[i],matrizCentroides);
9          //Para cada vetor de treino, calcula a menor distância até um dos K centroides;
10     até i = quantidadeVetoresDeTreino;
11     limiar = percentil(distanciasDeTreino,95);
12     //Calcula o limiar de decisão como o 95-percentil das menores distâncias;
13     repita
14         distanciasDeTeste[i] = menorDistancia(matrizDeTreino[i],matrizCentroides);
15         //Os vetores de teste são formados por todos os vetores convulsivos;
16         //e por alguns vetores não-convulsivos;
17         //Para cada vetor de teste, calcula a distância até o centroide;
18         //e compara com o limiar;
19         if distanciasDeTeste[i] > limiar then
20             vetorDePredições[i] = -1;
21             //Se a distância é maior que o limiar, classifica como convulsivo;
22         else
23             vetorDePredições[i] = +1;
24         //se a distância é menor que o limiar, classifica como normal;
25     até i = quantidadeVetoresDeTeste;
```

4. Avaliação Experimental

Neste seção, serão discutidos todos os detalhes relacionados aos dados e configurações utilizados nos experimentos realizados. Os algoritmos propostos foram implementados

em *MATLAB*® e estão disponibilizados, assim como os dados pré-processados, no repositório: `https://github.com/cabrau/anomaly-detection-eeg`.

4.1. Dados Utilizados

O conjunto de dados utilizados neste trabalho é o mesmo utilizado nos trabalhos relacionados e consiste em gravações contínuas do EEG de escalpo, realizados em 24 pacientes pediátricos do Hospital Infantil de Boston, após a retirada da medicação para a avaliação de cirurgia de epilepsia. Os arquivos utilizados pertencem ao projeto CHB-MIT Scalp EEG Database e podem ser acessados através do website: `https://www.physionet.org/pn6/chbmit/`.

Cada exame EEG apresenta uma taxa de amostragem de 256 Hz, utilizando-se 23 canais, com duração aproximada de 1 hora. Contudo, existem gravações de até quatro horas, dependendo do quadro de crises epiléticas do paciente. Foram realizados vários exames em cada paciente, gerando um total de 686 arquivos (aproximadamente 32Gb de dados). Os arquivos estão catalogados em com convulsões e sem convulsões, onde os momentos em que ocorrem crises foram previamente classificados por especialistas. Em todos os arquivos, foram detectados 197 convulsões distribuídas em 141 arquivos, totalizando um total de 195,5 minutos para todos os pacientes.

4.2. Extração de Atributos

O EEG é composto de uma ampla faixa de componentes de frequência onde se destacam ritmos cerebrais associados a certos estados do cérebro. Portanto, atributos que descrevam as frequências do sinal são significativos para a identificação de crises epilépticas [Sanei and Chambers 2007]. Nesse trabalho, optamos por utilizar como atributos a Densidade Espectral de Potência do Sinal (PSD)[Hayes 1996], uma função que descreve como a energia de um sinal se distribui em suas frequências. Esse método de extração de atributos é comumente utilizado na literatura com bons resultados. Sabe-se que o sinal EEG é não-estacionário, portanto seu espectro muda com o tempo. Tal sinal pode ser aproximado como estacionário por partes, ou seja, uma sequência de segmentos de sinal estacionários independentes. Pode-se assumir que a duração de um intervalo estacionário mínimo seja de 2 segundos [McEwen and Anderson 1975]. Durante o desenvolvimento deste trabalho, também foi experimentada uma técnica de extração de atributos no domínio do tempo, através da estimação de coeficientes de Codificação Linear Preditiva (LPC) [O'Shaughnessy 1988]. Este método utiliza um modelo autorregressivo para parametrizar o sinal, e utiliza os coeficientes como atributos. Destaca-se como característica fundamental de um processo autorregressivo o fato da observação atual estar correlacionada com a observação anterior, ou seja, assume-se uma correlação significativa entre as observações anteriores, o que é uma suposição forte. Para os métodos de detecção propostos, essa técnica de extração de atributos resultou em um baixo desempenho, com acurácia abaixo de 50%, e não serão apresentados.

4.3. Construindo os Vetores de Atributos

Na base de dados utilizada, o EEG foi captado utilizando 23 canais. Portanto, cada exame possui 23 sinais, amostrados com uma frequência de 256Hz. Para cada canal de EEG, vetores de atributos PSD são extraídos em janelas de tempo (épocas) de 2 segundos, intervalo no qual o sinal é considerado aproximadamente estacionário. Portanto, dada a taxa de amostragem de 256Hz, cada época contém 521 amostras.

Para extrair os atributos PSD, foram realizados os seguintes passos:

- **Passo 1 -** Para a época corrente (janela de 2 segundos), aplicamos o método de periodograma de Welch, utilizando uma janela Gaussiana de tamanho 128. Esse procedimento foi repetido para todos os 23 canais.
- **Passo 2 -** Aplicamos uma escala logarítmica aos valores PSD resultantes para convertê-los em decibéis (dB). Ou seja, $PSD(dB) = 10\log_{10}(PSD)$.
- **Passo 3 -** Segmentamos os valores PSD (em dB) em 8 bandas de frequência, indo de 0.5Hz a 25Hz, e calculamos a média de cada banda. Para cada canal k, esse procedimento leva à computação de 8 atributos $x_{1,k}, x_{1,k}, ..., x_{M,k}$. Esses atributos são correspondentes a energia distribuída nos principais ritmos cerebrais.
- **Passo 4 -** Para cada época t, concatenou-se as 8 médias espectrais extraídas de cada um dos 23 canais. Esse processo forma um vetor de atributos X_t com dimensão $M \times N$ = 184, definido como

$$X_T = [x_{1,1}, x_{2,1}, ..., x_{M,1}|...|x_{1,N}, x_{2,N}, ..., x_{M,N}]^T \quad (1)$$

Esse procedimento foi realizado para todos os sinais de um determinado paciente. Assim, para cada paciente, temos uma matriz de atributos. Em seguida, os dados correspondentes às crises convulsivas, já previamente classificados por especialistas, foram separados, produzindo assim duas matrizes de atributos: uma com os dados referentes às crises convulsivas e outra com os dados não-convulsivos. Em geral, a quantidade instâncias na matriz com dados convulsivos corresponde a menos de 1% da quantidade de instâncias não convulsivas, evidenciando o grande desbalanceamento entre as classes.

4.4. Métricas de Desempenho

Tratando-se de um problema de classificação binária, utilizamos como métricas de desempenho a acurácia (AC), sensibilidade (SB) e a especificidade (EP). Estas métricas são amplamente utilizadas na literatura consultada e permitem uma comparação de resultados.

- **acurácia** (AC): a proporção de predições corretas, sem levar em consideração o que é positivo e o que é negativo. Esta medida é altamente suscetível a desbalanceamentos do conjunto de dados e pode facilmente induzir a uma conclusão errada sobre o desempenho do sistema:

$$AC = \frac{(V_P + V_N)}{TOTAL} \quad (2)$$

- **sensibilidade** (SB): a proporção de verdadeiros positivos, ou seja, a capacidade do sistema em predizer corretamente a condição para casos que realmente a têm;

$$SB = \frac{V_P}{(V_P + F_N)} \quad (3)$$

- **especificidade** (EP): A proporção de verdadeiros negativos, ou seja, a capacidade do sistema em predizer corretamente a ausência da condição para casos que realmente não a têm.

$$EP = \frac{V_N}{(V_N + F_P)} \quad (4)$$

É importante ressaltar que devido ao alto desbalanceamento das classes, a acurácia por si não representa uma métrica de desempenho significativa, visto que existem poucos dados convulsivos. Considerando apenas a acurácia, um classificador cuja todas predições fossem não-convulsivas ainda teria um desempenho alto, mesmo errando todas as crises convulsivas. Considerando a aplicação desejada, que é auxiliar neurologistas na detecção de crises em longos exames, o custo de um falso-negativo é maior do que o de um falso-positivo, pois o especialista poderia identificar e descartar o último, mas um falso-negativo poderia passar despercebido. Logo, a sensibilidade é uma métrica muito significativa.

5. Resultados

Esta seção tem por finalidade avaliar o desempenho dos três métodos proposto. Para isso, os métodos concebidos foram aplicados em três pacientes distintos, 01, 05 e 08, onde o critério de seleção foram os resultados encontrados na literatura para cada paciente, buscando, assim, criar um subconjunto com diversidade de padrões de EEG. Na literatura, a análise dos sinais do paciente 01 possui, em geral, bom desempenho. O paciente 05 apresenta desempenho intermediário e o paciente 08 quase sempre apresenta o pior desempenho. Foram realizadas 20 rodadas de treino e teste para cada um dos três pacientes, gerando as métricas de desempenho. Em cada rodada os dados de treino e teste foram escolhidos aleatoriamente. Ao final, para cada paciente e para cada uma das três métricas utilizadas (AC, SB e EP), registramos o melhor valor, o pior valor e a média dos resultados obtidos nas 20 rodadas de treino e teste.

5.1. Método de Distância ao Centroide Não-Convulsivo

Para este método, na fase de treinamento, foram utilizadas 98% das instâncias não-convulsivas, gerando um modelo que representa os dados "normais", ou seja, sem convulsão. Já na fase de teste, utilizamos os 2% restantes das instâncias não-convulsivas e 100% das instâncias convulsivas (o que corresponde a cerca de 1% do total de instâncias). Esta abordagem foi aplicada para cada paciente. Assim, criamos um modelo preditivo distinto para cada paciente. Para o paciente 1, por exemplo, 60420 instâncias não-convulsivas foram utilizadas na fase de treinamento. Já na fase de teste, foram utilizadas 611 instâncias não-convulsivas e 199 instâncias convulsivas. Assim, na fase de teste foram utilizadas ao todo 810 instâncias.

A Tabela 1 apresenta os resultados obtidos pelo Método da Distância ao Centroide Não-Convulsivo. Para cada paciente, destacamos o melhor valor obtido, o pior valor encontrado e a média dos resultados para cada uma das três métricas selecionadas (AC, SB e EP).

Tabela 1. Resultados do teste de desempenho para o Método de Distância ao Centroide Não-Convulsivo.

	Paciente 1			**Paciente 5**			**Paciente 8**		
	Melhor	**Pior**	**Média**	**Melhor**	**Pior**	**Média**	**Melhor**	**Pior**	**Média**
AC	96.39	93.63	95.37	97.10	93.57	95.26	95.72	92.43	94.10
SB	96.88	96.88	96.88	97.26	97.26	97.26	92.68	91.87	92.36
EP	96.31	93.12	95.14	97.07	92.91	94.90	98.34	92.27	95.28

Observa-se que, apesar de considerar uma modelagem simples da distribuição dos dados, o modelo gerou bons resultados utilizando os atributos PSD. Portanto, a distribuição de energia entre as frequências do sinal segue a hipótese inicial assumida de que os dados se distribuem ao redor do centroide. De fato, os ritmos cerebrais de uma crise convulsiva são compostos por frequências que, em geral, são distinguíveis do comportamento normal.

Observa-se também que os resultados do paciente 8 apresentam um desempenho ligeiramente inferior ao dos demais pacientes. Possivelmente este resultado deve-se ao tipo de crise epiléptica do paciente 08 ter um padrão similar ao padrão normal. De fato, nos resultados encontrados na literatura o paciente 08 obtém sempre o pior desempenho. No trabalho de Liang et al. 2019, por exemplo, o paciente 8 atingiu sensibilidade de 60%. Encontra-se na literatura médica casos onde não há alteração perceptível no EEG durante crises, dificultando tanto o diagnóstico quanto a detecção por algoritmos. Uma vez que trata-se de uma condição onde os padrões podem variar enormemente entre pacientes, esse tipo de caso pode ocorrer. Neste cenário, apesar de inferior aos pacientes 1 e 5, as médias do paciente 8 ficaram sempre acima de 90%, indicando a robustez desse método mesmo para um caso mais complexo.

5.2. Método de Distância ao Centroide Convulsivo

Com a finalidade de avaliar o Método de Distância ao Centroide Convulsivo, 70% dos dados convulsivos foram selecionados aleatoriamente para formar o modelo, na fase de treinamento. Já na fase de teste foram utilizados os 30% restantes dos dados convulsivos juntamente com 1% dos dados não convulsivos. Essa proporção difere consideravelmente do método anterior devido a diferença entre as quantidades de dados de crises disponíveis.

A Tabela 2 apresenta os resultados obtidos pelo Método da Distância ao Centroide Convulsivo. Para cada paciente, destacamos o melhor valor obtido, o pior valor encontrado e a média dos resultados para cada uma das três métricas selecionadas (AC, SB e EP).

Tabela 2. Resultados do teste de desempenho para o Método de Distâncias ao Centroide Convulsivo.

	Paciente 1			**Paciente 5**			**Paciente 8**		
	Melhor	**Pior**	**Média**	**Melhor**	**Pior**	**Média**	**Melhor**	**Pior**	**Média**
AC	100	98.66	99.44	99.12	93.83	96.06	97.03	69.64	93.83
SB	100	90.00	97.00	100	86.36	97.05	94.31	91.87	92.54
EP	100	99.02	99.68	99.51	93.17	95.95	100	52.78	94.72

Observa-se que esse modelo consegue altas taxas de acerto nos melhores casos, conseguindo até 100% no paciente 1, mas também apresenta uma maior variância. Os piores casos do paciente 8 apresentam valores bastante inferiores em relação aos cenários apresentados anteriormente, que obtiveram resultados mais consistentes. Novamente, é importante salientar que o EEG de cada paciente possui características únicas, o que torna métodos de reconhecimento de padrões nesse sinal altamente dependentes do paciente. Em comparação com o modelo DCNC, o modelo DCC é treinado com uma quantidade consideravelmente inferior de dados, uma vez que existem menos dados convulsivos dis-

poníveis, o que dificulta a generalização da classificação e pode explicar o seu desempenho inferior nos piores casos.

5.3. Método de Distâncias aos Centroides de K-*Clusters*

Por fim, avaliamos o Método de Distâncias aos Centroides de K-*Clusters*, no qual são calculadas as menores distâncias aos centroides de uma quantidade ótima de *clusters* envolvendo os dados não-convulsivos. Assim como no experimento com o modelo DCNC, foram utilizadas 98% das instâncias não-convulsivas para treino. O conjunto de teste é formado por 2% das instâncias não-convulsivas e por todas as instâncias convulsivas de cada paciente.

Pelo índice Davies-Bouldin, a quantidade ótima de *clusters* encontrada foi 3. A Tabela 3 apresenta os resultados obtidos pelo Método de Distâncias aos Centroides de K-*Clusters*. Para cada paciente, destacamos o melhor valor obtido, o pior valor encontrado e a média dos resultados para cada uma das três métricas selecionadas (AC, SB e EP).

Tabela 3. Resultados do Método de Distâncias aos Centroides de K-*Clusters*

	Paciente 1			**Paciente 5**			**Paciente 8**		
	Melhor	**Pior**	**Média**	**Melhor**	**Pior**	**Média**	**Melhor**	**Pior**	**Média**
AC	97.03	93.42	94.98	96.68	93.57	94.77	98.03	94.41	96.15
SB	96.88	90.63	96.56	95.89	94.52	95.75	97.56	92.68	97.32
EP	97.05	93.12	94.73	97.07	93.15	94.60	98.34	92.82	95.36

Esse método obteve médias bastante elevadas, próximas ou superiores aos resultados dos métodos apresentados anteriormente. Além disso, o Método de Distâncias aos Centroides de K-*Clusters* manteve uma consistência de resultados entre os pacientes. No caso do paciente 8, ele obteve os melhores resultados. De fato, a vantagem desse método é que ele utiliza uma representação mais fiel da distribuição dos dados, considerando vários centroides ao invés de um, sendo, portanto, mais robusto a ruídos. Porém, esse método tem como desvantagem o alto custo computacional em relação aos anteriores, principalmente na etapa de treino, onde é preciso executar o algoritmo K-Médias diversas vezes para um grande volume de dados. Adicionalmente, a etapa de teste também apresenta um custo maior, uma vez que é preciso testar a distância de cada instância a vários centroides ao invés de somente um.

5.4. Análise Comparativa

Por fim, realizamos uma análise comparativa entre os três modelos propostos neste artigo e as abordagens encontradas na literatura que utilizaram a mesma base de dados e as mesmas métricas de desempenho. A Tabela 4 ilustra a média dos valores obtidos para as três métricas selecionadas (AC, SB e EP), para cada um dos métodos analisados. A coluna Tipo indica se o classificador foi treinado e testado para cada paciente separadamente (paciente específico), como realizado neste trabalho, ou foi treinado e testado com os dados de todos os pacientes (inter-pacientes, IP) em conjunto.

De forma geral, os três modelos propostos obtiveram bons resultados, dentro das condições estabelecidas. Apesar de serem utilizados métodos relativamente simples, o desempenho foi comparável aos melhores resultados do estado da arte, que utilizam técnicas

Tabela 4. Análise Comparativa

Autor, Ano	Classificador	Tipo	AC %	SB %	EP %
[Shoeb and Guttag 2010]	SVM	Paciente-específico	96	-	-
[Acharya et al. 2011]	SVM	Inter-pacientes	96	96	97
[Fergus et al. 2016]	KNNC	Inter-pacientes	88	88	93
[Bhattacharyya and Pachori 2017]	Random Forest	Paciente-específico	94.4	97.9	99.5
[Liang et al. 2019]	LTCN	Inter-pacientes	84	99	99
Neste trabalho	DCNC	Paciente-específico	95.3	96.1	95
Neste trabalho	DCC	Paciente-específico	96.4	95.5	96.7
Neste trabalho	DCKC	Paciente-específico	95.3	96.5	96.6

de classificação mais sofisticadas baseadas em aprendizado supervisionado. Isso demonstra a relevância da abordagem de detecção de anomalias no problema de identificação de crises convulsivas em EEG.

6. Conclusões e Trabalhos Futuros

A principal contribuição desse trabalho foi a concepção e avaliação de três modelos distintos para identificar crises convulsivas, utilizando detecção de anomalias e atributos de frequência. Não encontramos na literatura nenhum trabalho que tenha investigado essa extratégia. Os resultados experimentais mostraram que os modelos propostos apresentam um desempenho semelhante aos melhores resultados encontrados na literatura. Além disso, os modelos propostos possuem robustez para lidar com sinais ruidosos, simplicidade de treino e baixo custo computacional. Como trabalhos futuros, pretendemos investigar a aplicação destes métodos na classificação inter-pacientes.

7. Agradecimentos

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro concedido na forma de bolsa de mestrado que tornou este trabalho possível.

Referências

[A. Zimek and Kriegel 2012] A. Zimek, E. S. and Kriegel, H. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387.

[Acharya et al. 2011] Acharya, U. R., Sree, S. V., Chattopadhyay, S., YU, W., and ANG, P. C. A. (2011). Application of recurrence quantification analysis for the automated identification of epileptic eeg signals. *International Journal of Neural Systems*, 21(03):199–211. PMID: 21656923.

[Adeli and Ghosh-Dastidar 2010] Adeli, H. and Ghosh-Dastidar, S. (2010). *AUTOMATED EEG-BASED DIAGNOSIS OF NEUROLOGICAL DISORDERS: Inventing the Future of Neurology.* New York: CRC Press.

[Bhattacharyya and Pachori 2017] Bhattacharyya, A. and Pachori, R. B. (2017). A multivariate approach for patient-specific eeg seizure detection using empirical wavelet transform. *IEEE Transactions on Biomedical Engineering*, 64(9):2003–2015.

[Chan et al. 2008] Chan, A. M., Sun, F. T., Boto, E. H., and Wingeier, B. M. (2008). Automated Seizure onset detection for accurate onset time determination in intracranial EEG. *Clinical Neurophysiology*, 119. pp. 2687-2696.

[Davies and Bouldin 1979] Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227.

[Fergus et al. 2016] Fergus, P., Hussain, A., Hignett, D., Al-Jumeily, D., Abdel-Aziz, K., and Hamdan, H. (2016). A machine learning system for automated whole-brain seizure detection. *Applied Computing and Informatics*, 12(1):70 – 89.

[Hayes 1996] Hayes, M. H. (1996). *Statistical Digital Signal Processing and Modeling*. USA: John Wiley and Sons.

[Kanashiro 2006] Kanashiro, A. L. A. N. (2006). EPILEPSIA: prevalência, características epidemiológicas e lacuna de tratamento farmacológico. . 2006. 135 f. Master's thesis, Tese (Faculdade de Ciências Médicas da Universidade Estadual de Campinas).

[Khan et al. 2012] Khan, Y. U., Rafiuddin, N., and Farooq, O. (2012). Automated seizure detection in scalp eeg using multiple wavelet scales. In *2012 IEEE International Conference on Signal Processing, Computing and Control*, pages 1–5.

[Knorr 2000] Knorr, E.; NG, R. T. T. V. (2000). Distance-based outliers: Algorithms and applications. *VLDB Journal*, 8(3):237–253.

[Liang et al. 2019] Liang, W., Pei, H., Cai, Q., and Wang, Y. (2019). Scalp eeg epileptogenic zone recognition and localization based on long-term recurrent convolutional network. *Neurocomputing*.

[MacQueen 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. Math. Stat. Probab. Vol. 1 Stat.*, pages 281–297, Berkeley, Calif. University of California Press.

[McEwen and Anderson 1975] McEwen, J. A. and Anderson, G. B. (1975). Modeling the stationarity and gaussianity of spontaneous electroencephalographic activity. *IEEE Transactions on Biomedical Engineering*, BME-22(5):361–369.

[Niedermeyer and da Silva 2001] Niedermeyer, E. and da Silva, F. L. (2001). *Electroencephalography – Basic Principles, Clinical Applications and Related Fields*, volume 1. Williams Williams.

[O'Shaughnessy 1988] O'Shaughnessy, D. (1988). Linear predictive coding. *IEEE Potentials*, 7(1):29–32.

[Sanei and Chambers 2007] Sanei, S. and Chambers, J. A. (2007). *EEG Signal Processing*. England: John Wiley and Sons.

[Shoeb and Guttag 2010] Shoeb, A. and Guttag, J. (2010). Application of Machine Learning To Epileptic Seizure Detection. *Appearing in Proceedings of the 27th International Conference on Machine Learning , Haifa, Israel*.

[Subasi and Ercebeli 2005] Subasi, A. and Ercebeli, E. (2005). Classification of EEG signal using neural network an logistic regression. *Computer Methodis and Programs in Biomedicine*, 78. pp. 87-99.

[Varun Chandola 2009] Varun Chandola, Arindam Banerjee, V. K. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3).

[Webb 2002] Webb, A. (2002). *Statistical Pattern Recognition*. John Wiley Sons.

[World Health Organization 2017] World Health Organization (2017). Epilepsy fact sheet.

34th Brazilian Symposium on Databases

October 7-10, 2019
Fortaleza - CE - Brazil

SBBD PROCEEDINGS
SHORT PAPERS

Promotion

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

Organization

Departamento de Computação, Universidade Federal de Ceará– UFC

Program Chair

Fábio Porto, LNCC

34th Brazilian Symposium on Databases

October 7-10, 2019
Fortaleza - CE - Brazil

Promotion

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

Organization

Departamento de Computação, Universidade Federal de Ceará– UFC

SBBD Steering Commitee

Ângelo Brayner (UFC)
Bernadette Lóscio (UFPE) coordenadora da CEBD
Carina Dorneles (UFSC)
Sérgio Lifschitz (PUC-Rio)
Fábio Porto (LNCC)
Carmem Hara (UFPR)

SBBD 2019 Commitee

Steering Committee Chair
Bernadette Lóscio (UFPE)

Local Chair
José Maria da Silva Monteiro Filho (UFC, Brazil)

Full Paper Chair
Carina F. Dorneles (UFSC, Brazil)

Short Paper Chair
Fábio Porto (LNCC, Brazil)

Demos and Applications Chair
Robson L. F. Cordeiro (ICMC-USP, Brazil)

Thesis and Dissertation Workshop Chair
Jonice Oliveira (UFRJ, Brazil)

Tutorials Chair
Altigran Soares da Silva (UFAM, Brazil)

Short course Chair
Maria Cláudia Cavalcanti (IME, Brazil)

Workshop Chair
José Antônio Macedo (UFC, Brazil)

Thesis and Dissertation Contest Chair
Caetano Traina Jr. (USP, Brazil)

Graduation Student Workshop Chair
Ticiana Linhares (UFC, Brazil)

Local Organization Committee
SBBD Local Chair: José Maria da Silva Monteiro Filho (DC/UFC)
Leonardo Oliveira Moreira (Instituto UFC Virtual/UFC)
Marum Simão Filho (UNI7)
Angelo Roncalli de Alencar Brayner (DC/UFC)
Javam de Castro Machado (DC/UFC)

Short Papers Program Committee
Alessandreia Oliveira (UFJF)
Altigran Soares da Silva (UFAM)
Ana Carolina Almeida (UERJ)
Anderson Ferreira (UFOP)
Angelo Brayner (UFC)
Bernadette F. Lóscio (UFPE)
Carlos Eduardo Pires (UFCG)
Carmem Hara (UFPR)
Cristina Ciferri (USP)
Damires Souza (IFPB)
Daniel de Oliveira (UFF)
Daniel Kaster (UEL)
Daniela Barreiro Claro (UFBA)
Deise Saccol (UFSM)
Denio Duarte (UFFS)
Duncan Ruiz (PUCRS)
Eduardo Borges (FURG)
Eduardo de Almeida (UFPR)
Eduardo Ogasawara (CEFET/RJ)
Elaine Sousa (USP)
Eveline Sacramento (FUNCEME)
Fernanda Baião (UNIRIO)
Flávio R. C. Sousa (UFC)
Helena Ribeiro (UCS)
Humberto Razente (UFU)
João Eduardo Ferreira (IME/USP)

Jonas Dias (DELL EMC)
Jonice de Oliveira Sampaio Oliveira (IM/UFRJ)
José Antonio Macêdo (UFC)
José de Aguiar Moraes Filho (UNIFOR)
José Monteiro (UFC)
José Palazzo Moreira de Oliveira (UFRGS)
Karin Becker (UFRGS)
Kelly Braghetto (IME/USP)
Luciano Barbosa (UFPE)
Luiz Celso Gomes Jr (UTFPR)
Luiz Manoel Rocha Gadelha Júnior (LNCC)
Maria Camila Nardini Barioni (UFU)
Maristela Holanda (UnB)
Michele Brandão (UFMG)
Mirella Moro (UFMG)
Moisés Carvalho (UFAM)
Pedro Eugenio Rocha Pedreira (Facebook Inc.)
Raquel Stasiu (PUCPR / UTFPR)
Raqueline Penteado (UEM)
Rebeca Schroeder (UDESC)
Renata Galante (UFRGS)
Renato Fileto (UFSC)
Robson Cordeiro (USP)
Robson Fidalgo (UFPE)
Sergio Lifschitz (PUC-Rio)
Sergio Mergen (UFSM)
Thiago Henrique Silva (UTFPR)
Ticiana Coelho da Silva (UFC)
Valéria C. Times (UFPE)
Vanessa Braganholo (UFF)
Vaninha Vieira (UFBA)

ADDITIONAL REVIEWERS
Marcelo Iury S. Oliveira (UFRPE)
Eduardo Pena (UFPR)
Guilherme Queiroz Vasconcelos (USP)
Nielsen Luiz Rechia Machado (PUCRS)

Table of Contents (Short Papers)

Artigo Visão: Processamento de Banco de Dados em Memória

Tiago R. Kepe[1,2], **Eduardo C. Almeida**[1], **Marco A. Z. Alves**[1]

[1]Universidade Federal do Paraná (UFPR)
[2]Instituto Federal do Paraná (IFPR)

{trkepe,eduardo,mazalves}@inf.ufpr.br

Abstract. *Neste artigo, apresentamos nossa visão de como os Sistemas Gerenciadores de Bancos de Dados (SGBD) podem integrar Processamento-em-Memória (PIM) em processamento de consultas. PIM promete mitigar os problemas clássicos de "memory-wall" e "energy-wall" presentes nas arquiteturas de computadores que são amplificados pela movimentação de dados na hierarquia de memória por aplicações de análise de dados. Compartilhamos com a comunidade uma análise empírica dos prós e contras do uso de PIM em três principais operadores da álgebra relacional: seleção, projeção e junção. Com base nos resultados obtidos começamos a desenvolver um escalonador de consultas PIM-consciente que apresenta resultados promissores chegando a reduzir em 3× o tempo de execução de consultas e o consumo de energia em pelo menos 25%. Concluímos nossa visão com uma discussão sobre os desafios e oportunidades para impulsionar pesquisas na concepção de um SGBD com PIM.*

1. Introdução

PIM foi originalmente concebido no final da década de 60 [Kautz 1969] e consiste em adicionar unidades de processamento dentro do dispositivo de memória para uso eficiente da largura de banda interna. Ao longo dos anos, a ideia de PIM foi revisitada de acordo com o progresso das tecnologias de memória. Por exemplo, componentes de processamento foram adicionados em discos magnéticos para executar certos algoritmos de processamento de consultas [Keeton and et al. 1998]. O mesmo ocorreu às tecnologias de memória RAM [Patterson and et al. 1997] que tentaram adicionar unidades lógicas dentro da DRAM para realizar computações específicas. Infelizmente, produtos comerciais não adotaram essas abordagens devido às limitações tecnológicas de *hardware* e, porque o crescimento contínuo de desempenho de CPU obedecia à lei de Moore e o escalonamento de Dennard. Soluções em discos *flash* também tentaram embutir unidades funcionais [Do et al. 2013]. Porém, essas soluções não foram exitosas devido à falta de uma interface de programação genérica e o custoso tratamento de erros de *hardware*.

Sistemas centrados em dados movem grandes quantidades de dados pela hierarquia de memória até a CPU. Por isso, novas arquiteturas PIM emergem devido à introdução das vias de silício em memória (*Through-Silicon Vias* (TSVs)) que tornaram as memórias 3D viáveis: 3D consiste na integração de *chips* de DRAM e células lógicas no mesmo dispositivo. Na prática, as memórias 3D invertem o modelo de processamento tradicional, i.e., elas movem a computação até os dados. Atualmente, as GPUs comerciais embutem memórias 3D nas tecnologias: Cubo Híbrido de Memória (HMC) [Jeddeloh and Keeth 2012] e Memória de Alta Largura de Banda (HBM) [Kim and Kim 2014].

Portanto, PIM surge como uma abordagem atrativa para reduzir a movimentação de dados em sistemas centrados em dados. Logo, em nossa visão, já é tempo de discutir

como SGBDs podem odotar PIM no processamento de consultas. Os maiores benefícios a serem explorados são a redução drástica no consumo de energia e a alta largura de banda interna dos novos dispositivos, pois estes fornecem um alto nível de acesso paralelo aos dados aliado ao processamento em memória. Atualmente, PIM tem sido usado de forma isolada como acelerador de operadores, tais como: seleção [Tome and et al. 2018] e junção [Mirzadeh et al. 2015]. Porém, notamos que o comportamento de cada operador depende de características do conjunto de dados e das configurações do sistema de caches. Neste artigo discutimos os seguintes aspectos para integração de PIM em SGBDs:

I. Idealizamos um processador de consulta PIM-consciente composto por otimizador e escalonador PIM-conscientes para usufruir do paralelismo intra-consulta entre CPU e PIM, e responder a pergunta de pesquisa: *Quando é melhor processar os dados no dispositivo de memória ao invés de movê-los pela hierarquia de memória até a CPU?*

II. Introduzimos um escalonador PIM-consciente como componente preliminar do processador de consulta que visa intercalar a execução intra-consulta entre CPU e PIM para explorar o potencial de cada dispositivo de processamento.

III. Divulgamos resultados preliminares sobre o impacto de PIM no processamento tradicional de consulta e discutimos a economia de energia com a redução drástica no tempo de execução além de uma ordem de magnitude para certos operadores.

IV. Compartilhamos uma lista de desafios e oportunidades para o desenvolvimento e integração de um SGBD-PIM.

2. Fundamentos de Arquiteturas PIM

As memórias 3D emergentes integram quatro ou oito *chips* de DRAM empilhados e interconectados via TSV até uma camada lógica na base. A pilha de *chips* de DRAM é organizada de forma vertical, formando partições chamadas "*vaults*" (Figura 1). As memórias 3D atuais possuem 32 *vaults* interdependentes para acessar e processar dados em paralelo alavancando a largura de banda interna em até 320 GB/s, e suportam instruções aritméticas, lógicas e binárias com operandos de até 16-*bytes*.

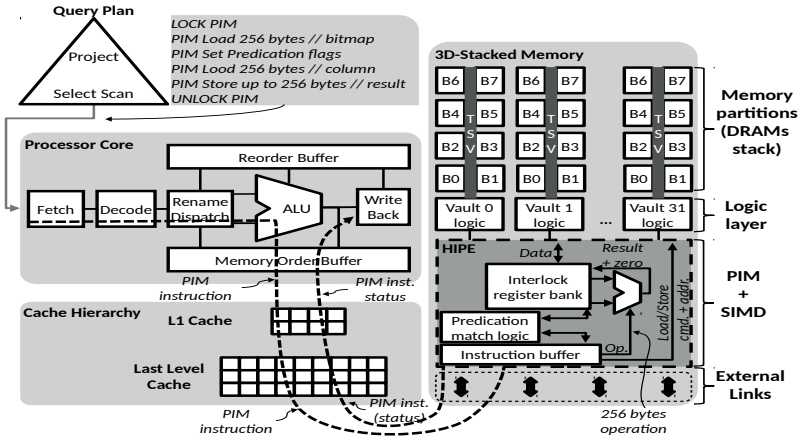


Figura 1. O fluxo de uma consulta na arquitetura von Neumann tradicional em conjunto com uma memória 3D com PIM e suporte SIMD.

O trabalho recente de [Tome and et al. 2018] apresenta uma extensão da arquitetura desses dispositivos para operar com instruções SIMD (*Single Instruction, Multiple*

Data) em vetores de 256-*bytes* para processamento de consultas, pois cada *vault* possui um *buffer* de *256-bytes* para acessar os bancos de memória correspondentes a sua partição. A Figura 1, lado direito, ilustra essa arquitetura PIM com suporte SIMD e, no lado esquerdo, a arquitetura von Neumann tradicional, i.e., o processador com a hierarquia de caches separados. No topo da Figura 1, o processador dispara instruções PIM diretamente ao dispositivo PIM contornando a hierarquia de caches. No final, o dispositivo PIM retorna somente o *status* da instrução à CPU para continuar a execução do *pipeline*. Esse mecanismo é a principal vantagem sobre atuais SGBDs (e.g. Netezza e Exasol) que também filtram dados em *hardware* antes de passar pela CPU. Porém, eles precisam condensar os dados filtrados em páginas para evitar a poluição do *bufferpool* que é custoso e propenso a erros. Por outro lado, os protocolos atuais de memória lidam com todas as idiossincrasias de instruções PIM, tais como: coerência de cache, correção de erros de códigos de memória (ECC) e acesso direto à memória (DMA). O fluxo de execução funciona no nível de instrução como o processamento tradicional da CPU (e.g., AVX/SSE): os programadores adicionam instruções PIM intrínsecas, como as *Intel Intrinsics*, e o compilador sinaliza-as como instruções de memória especiais.

3. Processamento de Consulta com PIM

Para avançarmos no desenvolvimento do processamento de consulta PIM-consciente, precisamos entender o impacto do uso do PIM na execução de consulta tradicional. Por isso, analisamos tal impacto em uma carga de trabalho analítica ao comutar o processamento entre CPU e PIM em termos do tempo de execução e consumo de energia dos operadores de projeção, seleção e junção. Realizamos nossa análise através do simulador SiNUCA devido às limitações de *hardware* e usamos os mesmos parâmetros aplicados por trabalhos relacionados [Tome and et al. 2018] e estimamos o consumo de energia com os mesmos parâmetros de DRAM do estado da arte [Jeddeloh and Keeth 2012]. Além de tudo, o SiNUCA tem sido extensivamente adotado em artigos científicos em arquitetura de computadores [Alves and et al. 2016] e banco de dados [Tome et al. 2018, Kepe 2018].

Avaliamos os operadores através do *benchmark* TPC-H de 1GB porque os dados de entrada cabem em cache que é o cenário mais propício para processamento em CPU. As implementações dos operadores usam operações SIMD de 64-*bytes* (CPU/AVX-512) e 256-*bytes* (PIM). Os resultados convergem para explorarmos nossa visão de SGBD-PIM.

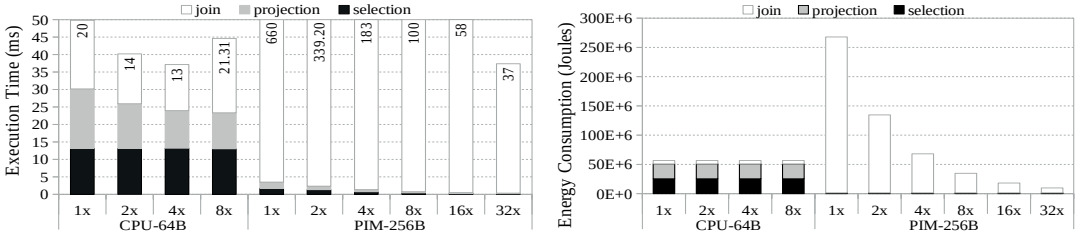


Figura 2. Tempo de execução e consumo de energia dos operadores da consulta 03 do TPC-H: seleção, projeção e junção em CPU-64B e PIM-256B com profundidades de *loop* desenrolados entre 1× até 32×.

Projeção: A Figura 2 mostra que a projeção em PIM reduz o tempo de execução em mais de uma ordem de magnitude, cerca de 60×. Também observamos uma redução de consumo de energia em 36×. **Seleção**: A seleção com PIM reduz o consumo de energia em 98% e o tempo de execução em 76× comparado a execução em CPU.

Os resultados são devidos ao comportamento em *streaming* dos operadores com PIM que causa baixo reuso de dados e pouca movimentação destes fora da memória principal. No processamento em CPU, as colunas são transferidas desde a memória principal passando por toda hierarquia de caches até a CPU que compara porções das colunas com os predicados de seleção e projeção. Contudo, todas as porções de dados transferidas não são reutilizadas durante a execução, inibindo o mecanismo de *caching*. Em contrapartida, nos operadores em PIM, a CPU envia apenas as instruções para o dispositivo PIM. Os operadores acessam e avaliam os dados com até 32 instruções PIM+SIMD de 256-*bytes*.

Junção: Em nossos experimentos, implementamos três algoritmos de junção, porém os algoritmos de Hash Join e Sort-Merge Join geram acessos aleatórios à memória que inibem o acesso paralelo aos dados e a alta largura de banda dos dispositivos PIM. Por isso, para fornecer uma percepção sobre os problemas de junção em PIM, nós analisamos a execução do Nested Loop Join (NLJ). O NLJ-PIM desenrola o *loop* interno até 32× para explorar os acessos paralelos aos dados. Porém, cada iteração desse *loop* causa instruções compulsórias de leitura e escrita na memória, pois a coluna de junção interna (*inner column* em cache) é reacessada em toda iteração do *loop*. Por isso, o processamento em CPU é melhor que a execução em PIM, chegando a ser 2,8× mais rápido, conforme mostra o gráfico da Figura 2. Mesmo o consumo de energia, o maior benefício de usar PIM, é 70% mais custoso em PIM. Entretanto, o NLJ-PIM torna-se viável quando a coluna interna de junção não cabe em cache o que inibe o reuso de dados. Neste ponto, o PIM obtém uma melhora de 1,38× comparado ao processamento em CPU.

4. Processamento de Consulta PIM-Consciente

Em nosso estudo preliminar sobre o impacto de PIM no processamento consulta, nós observamos que nem todo operador de banco de dados se beneficia do PIM. Portanto, a decisão sobre a arquitetura de processamento de um operador de consulta não é trivial. De fato, observamos que os operadores de consulta com alta reutilização de dados se beneficiam do mecanismo de *caching* e, assim, o processamento em CPU torna-se atraente, como no caso do NLJ. Por outro lado, os operadores com comportamento *streaming* (e.g., projeções e seleções) são mais adequados para o PIM. Outros operadores, como a agregação, podem ter um reuso de dados baixo ou médio e um comportamento de *streaming* parcial, o que torna pouco evidente qual é o melhor dispositivo de processamento.

Nossa investigação concentra-se em como intercalar a execução de consultas entre a CPU e o PIM, mas entendemos que um otimizador de planos PIM-consciente deve ser averiguado. Iniciamos nosso estudo com um escalonamento estático que usa um modelo de classificação baseado no perfil de operadores para decidir em qual arquitetura processar um dado operador. Diferente de trabalhos relacionados em escalonamento para GPU, esta estratégia não requer uma fase de calibração [Breß et al. 2012] nem heurística [Karnagel et al. 2014]. A Figura 3 apresenta três planos de execução de consulta a partir do plano ótimo: dois planos *hardware*-específicos, i.e., CPU e PIM, e um plano híbrido baseado no escalonamento estático de perfil. No plano híbrido, os operadores de seleção e projeção são processados em PIM, enquanto a junção é executada em CPU. Com o plano híbrido a execução de consulta tem uma melhora em quase 3× (Figura 3d), pois explora as vantagens de ambas as arquiteturas: operadores com alto reuso de dados são executados na CPU e operadores *streaming* são processados em PIM. Em termos de consumo de energia, o plano híbrido chega a reduzir em pelo menos 25%, Figura 3e.

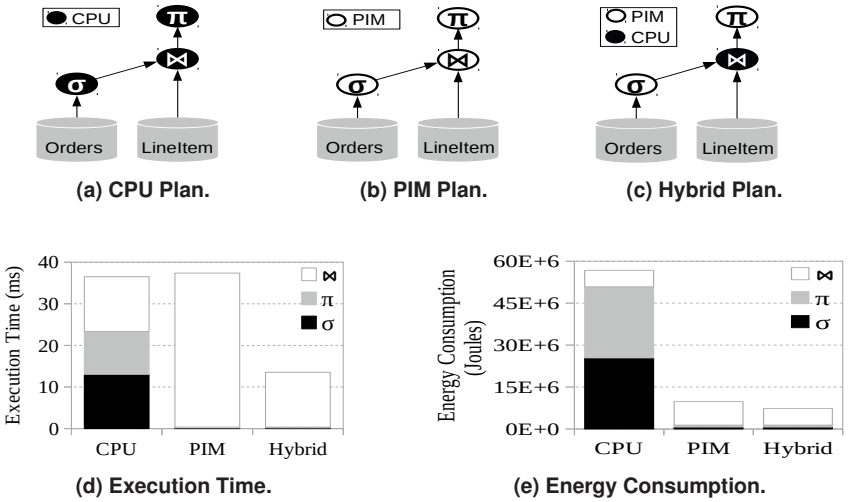


Figura 3. Planos de consulta: planos isolados CPU (a) e PIM (b), plano híbrido (c). Respectivos tempo de execução (d) e consumo de energia (e).

5. Desafios & Oportunidades

Coprocessamento Simultâneo: Em ambientes de processamento heterogêneos, o otimizador precisa identificar oportunidades de coprocessamento para evitar dispositivos ociosos ou consumo de energia ineficiente (e.g., o problema do muro de energia [Wang and Skadron 2013]). Além disso, a detecção de concorrência à memória é difícil de gerenciar no processamento simultâneo entre CPU e PIM.

Otimização do Plano de Consulta: O espaço de busca na otimização de planos de consulta já é um problema comum ao otimizador de consulta. A adição de planos híbridos aumenta ainda mais a complexidade para gerar planos de consultas candidatos. Desta forma, um otimizador de consulta PIM-consciente deve considerar características específicas de *hardware* para escolher o dispositivo de processamento apropriado.

Transações: Intrinsecamente, as instruções PIM aritméticas e lógicas são atômicas [HMC-Consortium 2015], o que proporciona oportunidades de pesquisa para transações em PIM e *Hybrid Transactional/Analytical Processing (HTAP)*. A ISA atual dos dispositivos PIM suportam instruções do tipo comparação-e-troca para teste de dados. Por conseguinte, instruções de atualização PIM podem ser sincronizadas em memória sem necessitar de checar as caches ou uso extra da banda de memória. A oportunidade é reduzir o *overhead* de bloqueio de dados e de estruturas (*locking* e *latching*) que correspondem a 30% das instruções OLTP [Harizopoulos and et al. 2008].

Adoção do SGBD: Em nossa visão SGBDs devem invocar instruções PIM no código dos operadores, similarmente as abordagens SSE e AVX. Nós também consideramos otimização de código para incluir funções intrínsecas na ISA dos dispositivos PIM.

6. Conclusão & Trabalho Futuro

Neste artigo apresentamos nossa visão de um SGBD que explora a largura de banda de memória sem precedentes dos novos dispositivos PIM aliada ao processamento em memória. Discutimos resultados promissores ao intercalar a execução de processamento intra-consulta entre PIM e CPU. Com nosso escalonador estático, planos de consulta

híbridos melhoraram o desempenho cerca de 3× e reduziram o consumo de energia em cerca de 25%. Apesar desses resultados promissores, sabemos que existem limitações a serem superadas. Por isso, nosso trabalho em andamento consiste em um escalonador dinâmico para criar e atualizar perfis e reescalonar operadores dinamicamente. Também consideramos investigar o otimizador de consulta PIM-consciente e os algoritmos Hash Join, Sort-Merge Join e agregações. Por último, compartilhamos com a comunidade uma lista de desafios e oportunidades abertas pela nossa visão para o *co-design* do SGBD-PIM.

AGRADECIMENTOS. Este trabalho foi parcialmente financiado pelo Instituto Serrapilheira (número de concessão Serra-1709-16621).

Referências

Alves, M. A. Z. and et al. (2016). Large vector extensions inside the HMC. In *DATE*.

Breß, S., Mohammad, S., and Schallehn, E. (2012). Self-tuning distribution of db-operations on hybrid CPU/GPU platforms. In *24th Grundlagen von Datenbanken*.

Do, J., Kee, Y., Patel, J. M., Park, C., Park, K., and DeWitt, D. J. (2013). Query processing on smart ssds: opportunities and challenges. In *SIGMOD*.

Harizopoulos, S. and et al. (2008). OLTP through the looking glass, and what we found there. In *SIGMOD*.

HMC-Consortium (2015). *Hybrid Memory Cube Specification 2.1*. HMC-30G-VSR PHY.

Jeddeloh, J. and Keeth, B. (2012). Hybrid memory cube new DRAM architecture increases density and performance. In *VLSIT*.

Karnagel, T., Habich, D., Schlegel, B., and Lehner, W. (2014). Heterogeneity-aware operator placement in column-store DBMS. *Datenbank-Spektrum*.

Kautz, W. H. (1969). Cellular logic-in-memory arrays. *IEEE Trans. Computers*.

Keeton, K. and et al. (1998). A case for intelligent disks (idisks). *SIGMOD Record*.

Kepe, T. R. (2018). Dynamic database operator scheduling for processing-in-memory. In *PhD@VLDB*.

Kim, J. and Kim, Y. (2014). HBM: memory solution for bandwidth-hungry processors. In *26th HCS*.

Mirzadeh, N., Kocberber, O., Falsafi, B., and Grot, B. (2015). Sort vs. hash join revisited for near-memory execution. In *ASBD@ISCA*.

Patterson, D. A. and et al. (1997). A case for intelligent RAM. *IEEE Micro*.

Tome, D. G. and et al. (2018). HIPE: HMC instruction predication extension applied on database processing. In *DATE*.

Tome, D. G., Kepe, T. R., Alves, M. A. Z., and de Almeida, E. C. (2018). Near-data filters: Taking another brick from the memory wall. In *ADMS@VLDB*.

Wang, L. and Skadron, K. (2013). Implications of the power wall: Dim cores and reconfigurable logic. *IEEE Micro*.

Explorando o uso de árvores B+ na Indexação de Dados por Similaridade

Jéssica N. B. de Farias[1], **Maria Camila N. Barioni**[1], **Humberto L. Razente**[1]

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

{jessica.farias,camila.barioni,humberto.razente}@ufu.br

Abstract. *This paper presents a new metric indexing method called GroupSim+ which allows narrowing the minimal cut-regions, when compared with related works. This characteristic of the method contributes to the optimization of similarity queries. The experiments performed with different datasets demonstrate the effectiveness of the approach adopted in the development of GroupSim+.*

Resumo. *Este artigo apresenta um novo método de indexação métrico denominado GroupSim+ que, quando comparado com trabalhos correlatos, permite um maior estreitamento das regiões mínimas de poda. Essa característica do método contribui para a otimização de consultas por similaridade. Os resultados dos experimentos realizados com diferentes conjuntos de dados reais demonstram a eficácia da abordagem adotada no desenvolvimento do GroupSim+.*

1. Introdução

Nas últimas décadas, o aumento do volume e da complexidade dos dados disponíveis para análise tem afetado o modo como os Sistemas de Gerenciamento de Bancos de Dados (SGBD) manipulam os dados. Esse fato gerou a necessidade do desenvolvimento de estruturas de indexação e operações de consulta próprias para atender essa mudança de paradigma. Dentro desse contexto, a otimização de consultas por similaridade tem sido uma questão de pesquisa importante na área de Bancos de Dados.

A otimização de consultas por similaridade, como as consultas por abrangência e aos k-vizinhos mais próximos, é normalmente realizada por meio da utilização de estruturas de indexação conhecidas como Métodos de Acesso Métricos (MAM). Nesses métodos, as comparações por similaridade requerem que os domínios de dados sejam representados em espaços métricos. Um espaço métrico é definido por um par $\langle \mathbb{S}, \delta() \rangle$, onde $\mathbb{S}$ é o domínio de dados e $\delta()$ é uma função de distância $\delta : \mathbb{S} \times \mathbb{S} \rightarrow R^+$ que obedece as propriedades de simetria, não-negatividade e desigualdade triangular. Essa última propriedade é especialmente importante para os MAM uma vez que permite podar regiões do espaço que certamente não contêm elementos que atendem as condições de uma consulta. Uma extensa revisão da área pode ser obtida em [Samet 2006].

Mapeamentos unidimensionais de dados têm sido empregados no desenvolvimento de MAM interessantes. A ideia básica desses métodos consiste em ordenar os elementos de dados considerando as distâncias em relação a um elemento de referência (pivô), como ilustrado na Figura 1. Isso permite a utilização de estruturas dinâmicas baseadas na relação de ordem total, como a árvore $B+$, para indexação dos dados. Exemplos de MAM baseados nessa estratégia são: *OmniB-Forest* [Traina-Jr et al. 2007], *iDistance* [Jagadish et al. 2005] e *GroupSim* [Razente et al. 2017]. A otimização das consultas por

similaridade de modo exato nesses métodos é baseada na propriedade de desigualdade triangular considerando cada pivô, o elemento de consulta e o raio de abrangência dado.

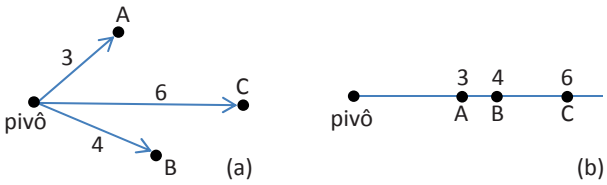


Figura 1. Pontos de um espaço bi-dimensional imersos em um mapeamento unidimensional. (a) Pontos 2D originais. (b) Mapeamento.

A principal característica do método *OmniB-Forest* é o emprego de uma árvore $B+$ para cada mapeamento unidimensional. O conjunto candidato de uma busca por abrangência nesse método é a intersecção dos anéis formados com base no elemento de consulta, o raio e o conjunto de pivôs. Já o método *iDistance* particiona o conjunto de dados considerando o conjunto de pivôs, armazenando-o em uma mesma árvore $B+$, por meio de uma constante para separar as partições. O método *GroupSim* emprega uma árvore $B+$ para indexação de um mapeamento unidimensional, entretanto, agrega as distâncias para os demais mapeamentos unidimensionais em cada entrada da estrutura, permitindo a computação de regiões mínimas de poda com um custo inferior aos outros métodos.

O trabalho apresentado nesse artigo visa contribuir para a otimização de consultas por similaridade por meio da exploração de uma nova abordagem baseada na utilização de árvores $B+$ na construção de índices dinâmicos. A abordagem adotada na proposta do novo MAM, denominado *GroupSim+*, explorou desvantagens identificadas nos MAM correlatos. Além disso, algoritmos eficientes de consulta por abrangência e aos k-vizinhos mais próximos também são propostos. Os experimentos iniciais ajudam a corroborar as hipóteses levantadas e a direcionar a realização de trabalhos futuros. O restante do artigo está organizado da seguinte forma. A Seção 2 descreve o problema e o novo MAM *GroupSim+*. A discussão a respeito dos resultados experimentais é apresentada na Seção 3. As conclusões e os trabalhos futuros são descritos na Seção 4.

2. O método GroupSim+

No método *OmniB-Forest* é necessário coletar os identificadores dos elementos até atingir o raio da consulta em todas as árvores $B+$ e então computar as intersecções entre essas regiões de abrangência enquanto no *iDistance* a poda é limitada apenas pelo anel do pivô de cada partição. Para tratar essas questões, o método *GroupSim* emprega uma árvore $B+$ para indexação das distâncias dos elementos em relação a um pivô e armazena um vetor de distâncias para os demais pivôs permitindo o descarte de identificadores pela região mínima de poda com custo menor. Entretanto, ao indexar os elementos de dados em relação a apenas um pivô, os anéis formados por um raio em uma consulta podem abranger um volume grande de identificadores. A hipótese para o desenvolvimento do método *GroupSim+* é que a combinação do particionamento do espaço com o armazenamento do vetor de distâncias para os demais pivôs resultará no estreitamento das regiões mínimas de poda em relação ao método *GroupSim*.

O *GroupSim+* é implementado por meio da extensão da árvore B+ de modo que os nós folha armazenem as seguintes entradas: $< chave, id, info[\,] >$, onde a ordenação

é dada pela $chave$, representada pela distância de um elemento ao pivô mais próximo, id é um identificador para o elemento e $info[\,]$ é um vetor de distâncias do elemento aos demais pivôs. Para a indexação de um conjunto dinâmico de elementos, seleciona-se inicialmente um conjunto estático de pivôs. A indexação de um elemento é feita da seguinte maneira. O elemento é inserido no final do arquivo de acesso aleatório (ORAF) e as distâncias para o conjunto de pivôs são calculadas. O particionamento dos dados na árvore $B+$ é dado pela adição de uma constante c multiplicada pelo número da partição à distância do elemento em relação ao pivô mais próximo, resultando na $chave$ de indexação. O id é a posição do elemento no ORAF e as distâncias para os demais pivôs são armazenadas em $info[\,]$. A constante c deve ser suficientemente grande para que não haja sobreposição entre os intervalos de diferentes partições.

Uma consulta por abrangência recebe como entrada um elemento de consulta q e um raio r e retorna uma lista de elementos que estão a até r de distância de q. No método *GroupSim+*, inicia-se procurando na árvore $B+$ pela chave mais próxima de $\delta(q, p_i) - r$ em cada partição i representada pelo pivô p_i. Todas as entradas com chaves no intervalo $\delta(q, p_i) - r$ a $\delta(q, p_i) + r$ devem ser verificadas (região em amarelo apresentada na Figura 2a), sendo que as distâncias armazenadas em $info[\,]$ permitem o descarte das entradas que certamente não fazem parte do conjunto resposta pela propriedade da desigualdade triangular (região em amarelo da Figura 2b), evitando assim a recuperação dos elementos do ORAF e a computação das distâncias em relação a q. As partições não cobertas pelo raio de consulta não precisam ser verificadas.

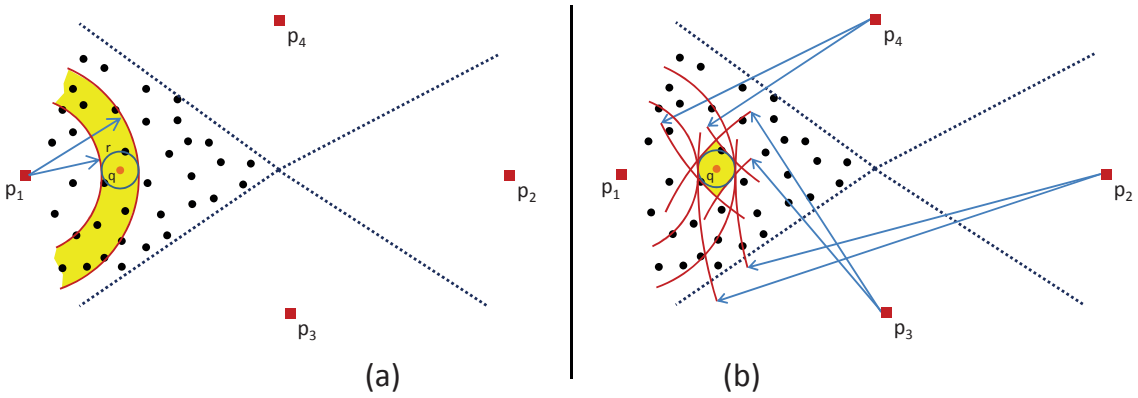


Figura 2. Região mínima de poda no *GroupSim+* para a partição representada pelo pivô p_1 com base no elemento de consulta q e raio r. (a) Anel restringe entradas da árvore $B+$ que precisam ser verificadas. (b) Distâncias armazenadas em $info[\,]$ permitem evitar a recuperação dos elementos do ORAF que não estiverem na região mínima de poda.

Uma consulta aos k-vizinhos mais próximos recebe como entrada um elemento de consulta q e um número k de elementos. Essa consulta retorna uma lista de até k elementos mais próximos, ordenados pela distância em relação a q. No método *GroupSim+*, inicia-se procurando na árvore $B+$ pela chave mais próxima de $\delta(q, p_i)$ na partição i representada pelo pivô p_i que contém o elemento de consulta q. A estratégia utilizada é a *start small and grow*, ou seja, uma constante Δ_r é utilizada como raio r inicial, verificam-se as entradas à esquerda e à direita até atingir o raio r, e caso decida-se continuar, adiciona-se Δ_r a r. Para as demais partições, se $\delta(q, p_i) - max_i \leq r$ na partição i representada pelo pivô p_i onde max_i é a maior distância entre os elementos da partição,

ou seja, se o raio da região de abrangência ultrapassa a partição que contém o elemento de consulta, devem ser verificadas as entradas em ordem decrecente a partir do limite max_i até $\delta(q, p_i) - r$. Para melhor desempenho, os nós folha da árvore $B+$ devem ser duplamente encadeados, para permitir percorrê-los tanto em ordem ascendente quanto descendente. Em todos os casos, as distâncias armazenadas em $info[\,]$ são utilizadas para verificar se cada elemento está dentro da região mínima de poda definida pela intersecção dos anéis obtidos a partir de cada pivô.

3. Experimentos

Os experimentos foram implementados[1] em C++ e executados em um computador equipado com CPU Intel Core i7-4770, 8 GB de memória principal e disco rígido de 1 TB. Os métodos foram avaliados em relação ao desempenho da realização de consultas por similaridade baseadas em raio e k-vizinhos mais próximos utilizando distância Euclidiana. Os conjuntos de pivôs foram selecionados pela heurística *MaxSum* [Socorro et al. 2011]. Os resultados para os seguintes conjuntos de dados são apresentados: Pendigits (10.992 elementos, 16 dimensões) e Covertype (581.012 elementos, 10 dimensões)[2]; Nasa (40.150 elementos, 20 dimensões) e Colors (112.682 elementos, 112 dimensões)[3].

O objetivo é avaliar o desempenho da nova estrutura em relação aos trabalhos correlatos considerando os seguintes critérios: o tempo, cálculos de distância e acessos ao disco. Para todos os métodos avaliados, a inclusão de um elemento envolve o cálculo das distâncias do elemento para cada elemento do conjunto de pivôs, a inclusão no final do arquivo ORAF, e a inclusão da entrada nas árvores $B+$. Assim, os métodos foram construídos com tempos de execução aproximadamente iguais. Foram selecionados aleatoriamente 100 elementos de cada conjunto de dados para realização das consultas.

Para os experimentos apresentados nas Figuras 3 e 4 os índices foram criados com a utilização de 6 pivôs. Na Figura 3 as consultas foram executadas com um raio variando de 1% a 10% da maior distância calculada entre todos os pares de elementos de cada conjunto (eixo das abcissas). Analisando esses gráficos é possível notar que considerando o número de acessos a disco, o método *iDistance* apresenta número de acessos inferior ao do *GroupSim+*. Entretanto, os métodos *OmniB-Forest*, *GroupSim* e *GroupSim+* apresentam comportamento similar, resultando em quantidades inferiores de cálculos de distância quando comparados com o *iDistance*. Isso ocorre devido ao fato que esses métodos utilizam todos os pivôs para definição da região mínima de poda enquanto o *iDistance* utiliza apenas um pivô em cada partição. Com relação ao tempo de execução das consultas, o *GroupSim+* apresentou redução de 41% para o raio $r = 1\%$ e 47% para o raio $r = 10\%$ em relação do *GroupSim* no conjunto Nasa.

Na Figura 4 as consultas foram executadas com $k = 10$ até 100 (eixo das abcissas). De maneira geral, o comportamento de todos os métodos na realização de consultas aos k-vizinhos mais próximos foi semelhante ao comportamento observado nas consultas por abrangência. Isso ocorre devido ao fato de que a otimização nesses algoritmos é realizada pela convergência da região de abrangência (definida por um raio) que contém k elementos.

[1]Biblioteca Arboretum: https://bitbucket.org/gbdi/arboretum
[2]UCI Machine Learning Repository: http://archive.ics.uci.edu/ml
[3]Metric Spaces Library: http://www.sisap.org/metricspaceslibrary.html

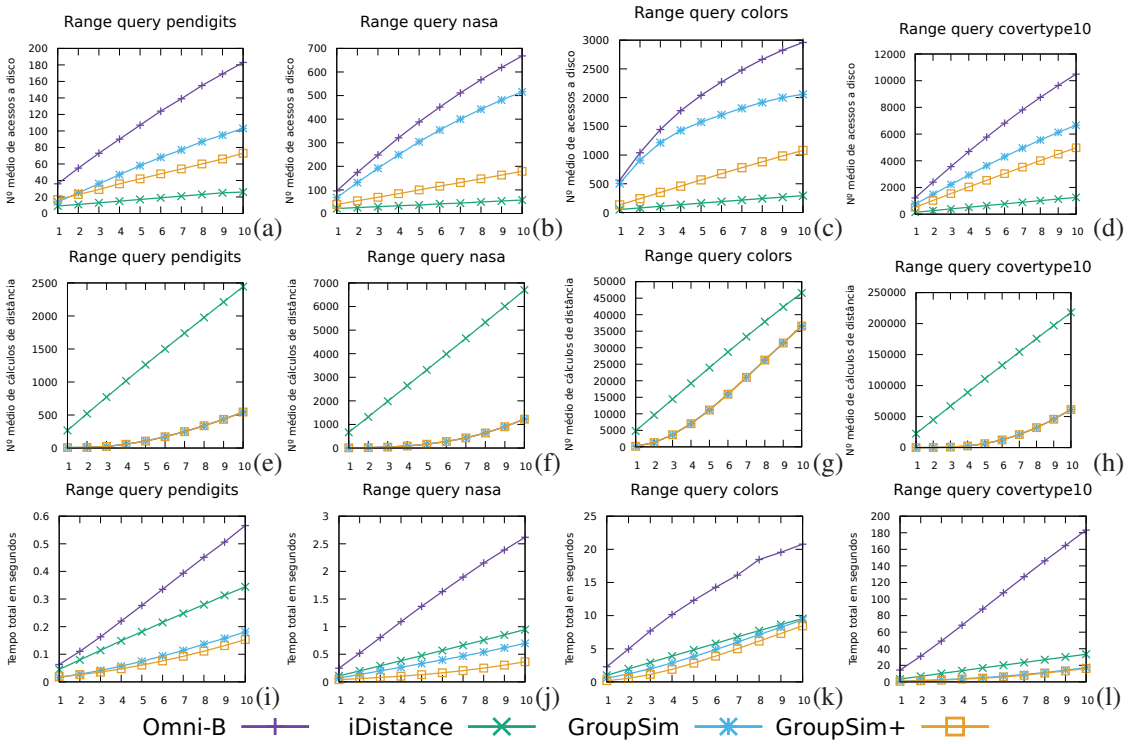


Figura 3. Consultas por abrangência. (a-d) acessos a disco. (e-h) cálculos de distância. (i-l) tempo total.

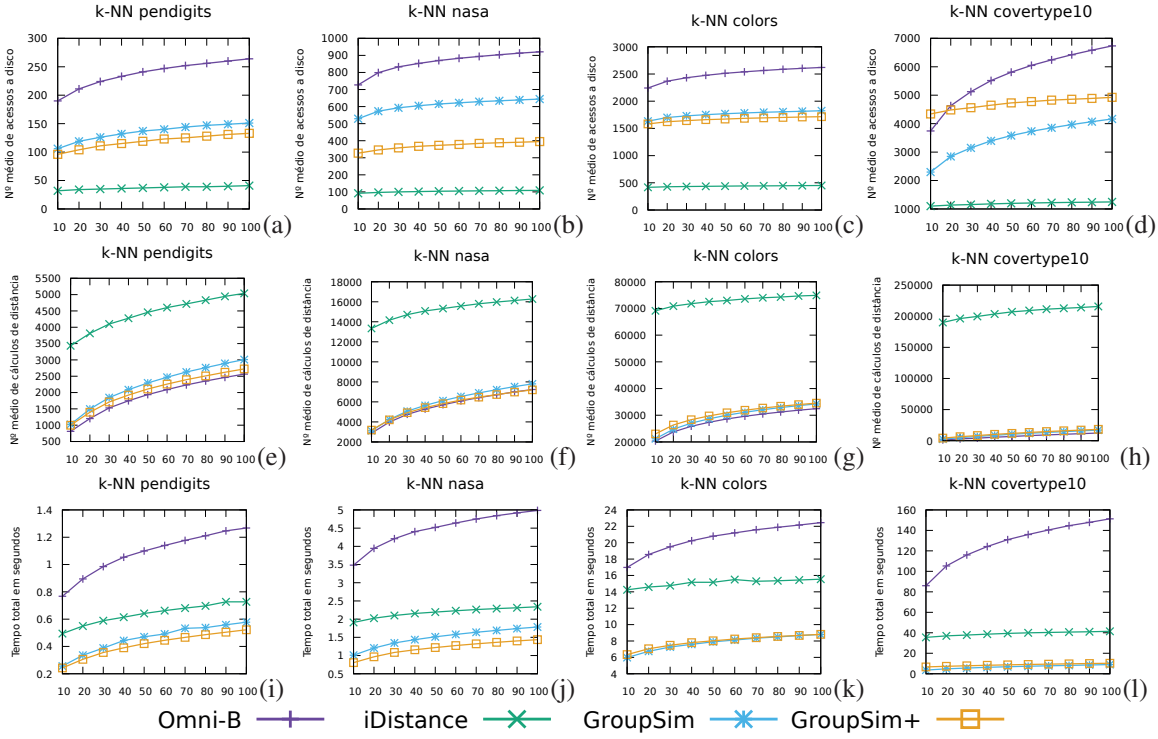


Figura 4. Consultas aos k-vizinhos mais próximos. (a-d) acessos a disco. (e-h) cálculos de distância. (i-l) tempo total.

A Figura 5 apresenta os resultados para consultas aos 10-vizinhos mais próximos executadas em índices construídos com 3 a 15 pivôs (eixo das abcissas). O aumento no número de pivôs impactou negativamente no tempo de execução do método *OmniB-Forest* para os conjuntos de dados testados, permitindo observar que a partir de 4 pivôs não houve diminuição nas regiões mínimas de poda e a inclusão de pivôs aumentou linearmente o tempo para computação da intersecção dos anéis. Os métodos *GroupSim* e *GroupSim+* apresentaram comportamento praticamente constante no tempo de execução para todos os conjuntos de pivôs testados. Apesar dos gráficos indicarem uma tendência de melhora no tempo de execução do *iDistance* com o aumento do número de pivôs, a inclusão de novos elementos em um cenário dinâmico pode reverter essa tendência. Pretende-se investigar essa hipótese em trabalho futuro.

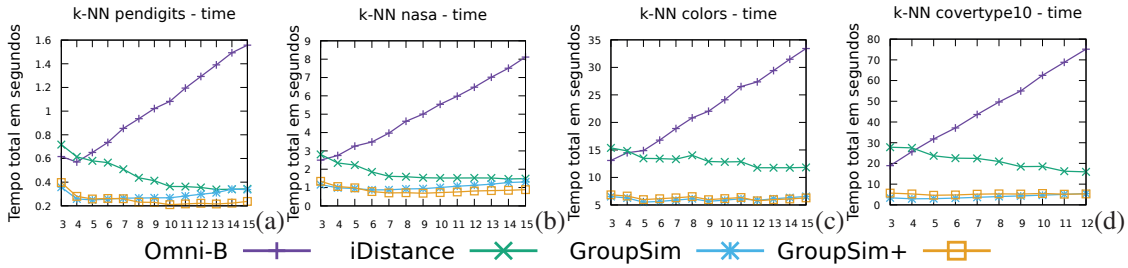


Figura 5. Consultas aos 10-vizinhos mais próximos executadas em índices construídos com 3 a 15 pivôs. (a-d) tempo total.

4. Conclusões

O *GroupSim+* considera as contribuições existentes no contexto de indexação de dados complexos utilizando mapeamentos unidimensionais e define um novo método de acesso métrico que visa aumentar o desempenho da recuperação por similaridade de elementos de dados por meio do estreitamento das regiões de busca e diminuição do número de cálculos de distância, a um mesmo custo de indexação. Como trabalho futuro, pretende-se explorar uma estratégia para redução dos acessos a disco.

Agradecimentos

Os autores agradecem ao CNPQ e a CAPES pelo apoio financeiro a este trabalho.

Referências

Jagadish, H., Ooi, B., Tan, K., Yu, C., and Zhang, R. (2005). iDistance: an adaptive b+tree based indexing method for nearest neighbor search. *ACM TODS*, 30(2):364–397.

Razente, H., Lima, R. B., and Barioni, M. C. (2017). Similarity search through one-dimensional embeddings. In *ACM SAC*, pages 874–879, Marrakech, Marrocos.

Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, San Francisco.

Socorro, R., Mico, L., and Oncina, J. (2011). A fast pivot-based indexing algorithm for metric spaces. *Pattern Recognition Letters*, 32(11):1511–1516.

Traina-Jr, C., Filho, R. F., Traina, A., Vieira, M. R., and Faloutsos, C. (2007). The omni-family of all-purpose access methods: A simple and effective way to make similarity search more efficient. *The VLDB Journal*, 16(4):483–505.

Fake News and Brazilian politics – temporal investigation based on semantic annotations and graph analysis

Luiz Gomes-Jr[1], **Gabriel Frizzo**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR) – Curitiba – Brazil

gomesjr@dainf.ct.utfpr.edu.br, gfrizzo@alunos.utfpr.edu.br

Abstract. *The widespread use of misleading news articles has been threatening democratic processes such as elections and referendums. Understanding how fake news address social entities (e.g. personalities and institutions) and how they react to social events are important factors in the fight against the trend. This paper employs a new approach based on semantic annotations and graph analysis to study fake news articles about Brazilian politics in a two year time span. We demonstrate how graph analysis can be used to track topic evolution and cluster related entities. A preliminary result also indicates that fake news tend to be influenced by public interest (and not the other way around).*

1. Introduction

Fake news has become a major source of concern in modern societies. The problem derives from the combination of cheap means of content publication provided by the Web with the easy and manipulable means of content promotion provided by social media. This scenario threatens democratic processes such as elections and referendums, and is characterized by a complex interaction between economic incentives and human vulnerabilities [Allcott and Gentzkow 2017].

The serious social threats and the difficulties in tackling the problem make it one of the most urgent research topics currently. Understanding the evolution of fake news over time and their interaction with social events can provide insights on the reasoning behind the publication of unreliable content. This understanding can then provide the basis for better algorithms for classification of non-credible sources and for better legal actions against such sources.

This paper investigates the topics and entities targeted by fake news in the Brazilian press in a two years time span. We employ semantic annotations and graph analysis models to assess, for the entities mentioned in the published news: the clusters formed (which can represent the latent topics, Section 4.1), the relative importance of entities through time (Section 4.2), and the interplay between the calculated metrics and social interest in the entities (Section 4.3). We discuss preliminary results of the techniques in Section 5.

The goal of this work-in-progress paper is twofold: (i) to provide a glimpse into the topics and entities discussed in the Brazilian fake news, and (ii) to demonstrate how semantic annotations and graph analysis can be employed for temporal analysis of text (not restricted to fake news analysis).

2. Related work

The term fake news has gained importance in recent years, mainly due to the widespread use of misinformation in democratic elections and referendums. Fake news can be defined as distorted signals that do not correlate with the truth [Allcott and Gentzkow 2017]. One important aspect of the fake news phenomenon is the mechanisms behind their dispersion. Allcott and Gentzkow [Allcott and Gentzkow 2017] analyse the influence of Social Media on the dissemination of fake news in the 2016 presidential election in the United States. The authors discuss the economics of fake news, how social media is a source of traffic for fake news websites, how the news favored the candidates and how education plays a role in people's vulnerability to false content.

Given the discussed importance of social media in the dissemination of fake content, it is important to understand the dynamics of the origins and spreading of misinformation. Jang et al. [Jang et al. 2018] analysed several tweets related to fake news stories in the 2016 US election employing a network science approach. The authors created phylogenetic trees for the tweets associated with each of the fake news topics. The analysis of the trees suggested interesting trends, like the tweets being promoted majoritarily by ordinary users and the tendency of fake news tweets mentioning non-credible sources.

Therefore, it is important to be able to detect non-credible sources. Linguistic patterns can provide valuable information about the likelihood of a content being fake. Monteiro et al. [Monteiro et al. 2018] produced the first reference corpus of fake news for the Portuguese language. The authors manually added fake news from online sources and aligned each article with a reliable story on the same topic. With the aligned corpus the authors were able to determine linguistic differences between fake and true content, such as the higher number of misspellings in fake news. The authors also set up a classification task and run several experiments to identify the best features to classify news as fake or true.

In this paper we use the corpus provided by Monteiro et al. but focus on a time analysis of the evolution of topics and entities cited in the fake news. Unlike Jang et al. [Jang et al. 2018], our focus is on the content produced by non-credible sources and not on its spread. We aim at understanding the interplay between fake news and the society in terms of political events and popular interest.

3. Data collection and cleaning

The corpus used in the analysis is the Fake.Br Corpus [Monteiro et al. 2018]. Only news classified as fake by the authors are used here. Each news article was annotated with DBPedia[1] entities using the SpotLight tool [Mendes et al. 2011]. For each article, a graph was formed with all entities mentioned as vertices. Edges in the graph connect all entities (complete graph), forming the article's co-citation graph. The graphs for the individual articles were used to build two distinct datasets: (i) a global graph for the entire time span of the corpus, and (ii) a series of graphs representing time partitions in the corpus.

For the global graph, the individual co-citation graphs were merged, with the number of occurrences of a given edge being used as its weight. Edges that occurred less than 20 times were omitted from the analysis. This dataset was used to identify clusters of

[1] https://dbpedia.org/

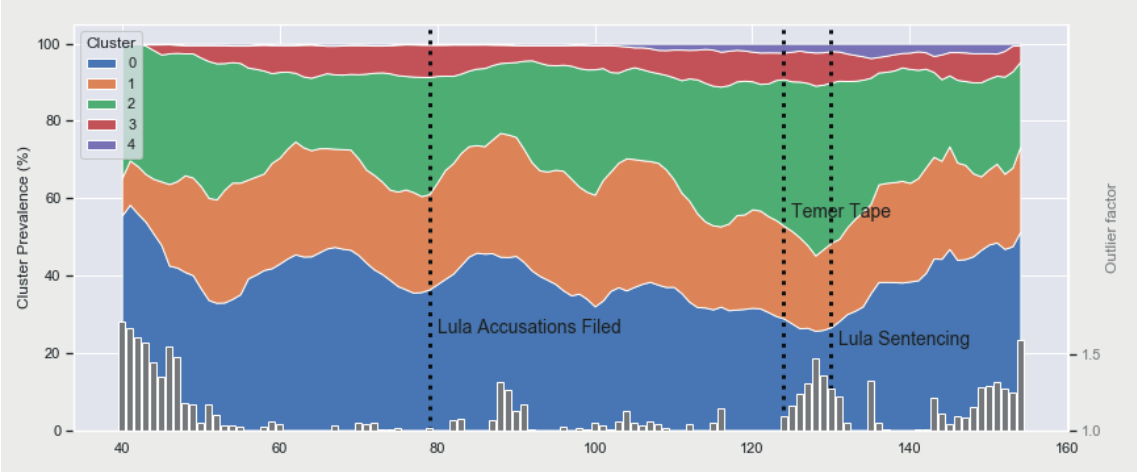


Figure 1. Prevalence of each cluster, outlier factors and events for reference

entities (Section 4.1). As for the time series of graphs, we defined time windows of 14 days with an overlap of 7 days (i.e. one 14 days window starts every week). Each window is considered a partition, which are numbered from 0 (starting on 2015-01-10) to 198 (ending on 2018-11-04). Only partitions between 38 and 155 are kept because the others have many gaps and much fewer collected articles. For each partition we merged the graphs of the respective articles and calculated several centrality and clustering metrics. This dataset was used to assess the evolution of graph measurements and other statistics over time (Section 4.2).

4. Data analysis

4.1. Cluster analysis

To define clusters of entities correlated in the graph, we applied the modularity algorithm (see [da F. Costa et al. 2007]) over the complete graph (all partitions merged – 154 nodes, 665 edges). The algorithm detected 5 distinct clusters, as shown below. The latent topic in each cluster seems to be (0): PT (workers party) politicians, (1): Lava Jato (car wash) investigations, (2): President Rousseff's Impeachment, (3): (residual cluster) Supreme Court, (4): (residual cluster) Bolsonaro and others.

Figure 1 shows the evolution of the prevalence of clusters through time. Each color represents a cluster with its width proportional to the percentage of entities of that cluster appearing in news in a given partition window. The time series for each clusters was smoothed to reduce noise and emphasize the trends. It can be seen, for example, how Lula's cluster (cluster 0, drawn in blue) becomes more prevalent after accusations were filed against him and also after he was considered guilty in the process.

To automatically detect important events in the period covered by the news, we employed outlier detection techniques. The algorithms are applied to single out partitions whose metrics deviate significantly from what would be expected.

The initial technique used is the Local Outlier Factor (LOF). It is applied to the data measuring the prevalence of each cluster (modularity class) in each partition. LOF is a multidimensional technique which in this case identifies isolated partitions in the 5-

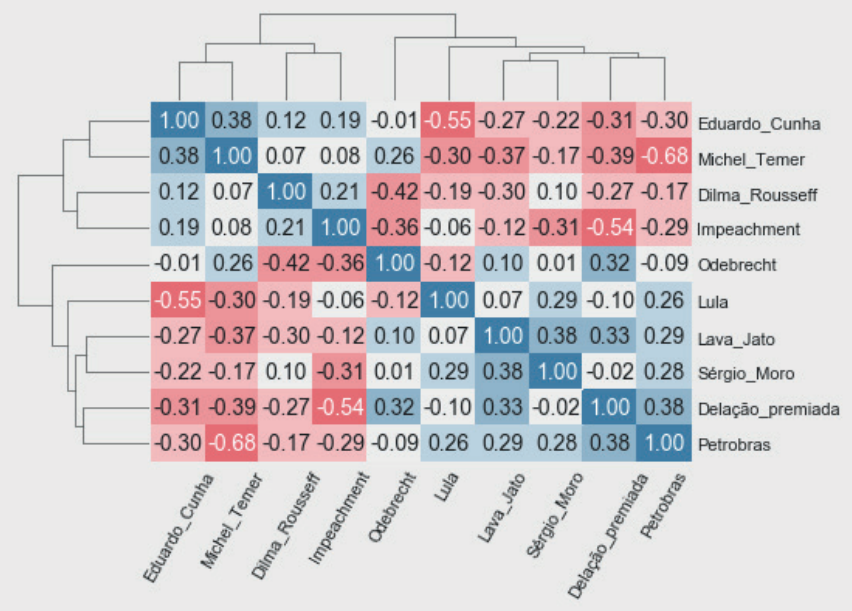


Figure 2. Correlations between top entities

dimensional space of cluster metrics. Intuitively, the algorithm will identify important shifts in discourse in the context of the news.

Figure 1 shows the Local Outlier Factor (bars in grey) representing how unexpected a given partition and its mix of clusters are. In the Figure, we overlaid three major political events in the period. There seems to be a correlation between the events and the distribution of the clusters, but this investigation is part of our ongoing efforts.

4.2. Time analysis of entities

The main focus of this paper is on understanding how the citation of individual entities vary with time. To assess the patterns, we need measurements of the importance or relevance of each entity in each time window. For each time window, we constructed its graph and calculated centrality measurements for the vertices (i.e. entities). The metrics used were *eigenvector centrality*, *closeness centrality*, and *betweenness centrality* (see [da F. Costa et al. 2007] for definitions).

The centrality measurements allow us to analyze the correlation of entities for the entire time period (all time windows). We can also cluster the entities based on spatial similarities of their vectors. Figure 2 shows the correlation matrix for top entities end their eigenvector centrality with dendograms displaying possible groupings. In the figure, it can be seen clusters of positive and negative correlations grouping entities related to the impeachment process and also with the car wash investigation. The eigenvector centrality metric demonstrated a clearly better separation among all metrics, but further analysis is needed to substantiate its superiority in this type of task.

4.3. Fake News vs. popular interest

One central aspect in understanding the mechanisms behind fake news is determining causal relationships between fake news cycles and popular interest. The basic question

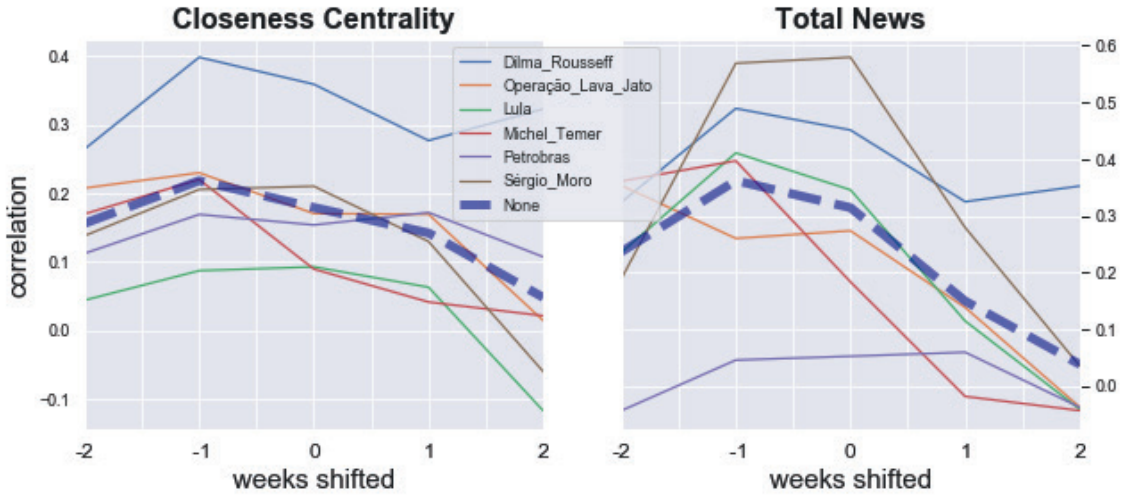


Figure 3. Correlation between metrics and Google Trends

here is whether fake news influence popular demand of topics or whether it feeds on the demand to promote their hidden agendas.

To make an initial assessment of this relationship, we first selected five top entities in our dataset, focusing on persons and entities that were part of the popular discussion in the period covered by the fake news corpus. The entities chosen were *Dilma Rousseff* (Brazil's impeached president), *Operação Lava Jato* (the codenamed 'car wash' investigation), Lula (ex-president and central theme in the investigations), Michel Temer (Rousseff's vice-president), Petrobras (Brazilian state-run oil company), Sérgio Moro (judge behind the car wash prosecutions).

For each of the chosen entities, we obtained their Google Trends[2] counts (representing search demands) for the weeks covered in the fake news corpus. We then analyzed the correlations between the Google Trends and the time partition measurements in our dataset. The correlation between total number of citation in fake news and Google Trends count reaches 0.58 for Sérgio Moro, which we consider high, since we believe that only the more polemical news would generate a web search response. The best centrality metric in terms of correlation was closeness centrality, reaching 0.36 for Dilma Rousseff.

The most interesting patterns appear when we time-shift the Google Trends counts. The idea is to correlate the entity measurements with their popular interest weeks before or ahead the time window of the fake news. Figure 3 (left) shows the variation of correlation for closeness centrality compared against the Google Trends for two weeks before, one week before, the same week, one week after, and two weeks after. The correlations tend to peak for the Google Trends of the week before (-1 in the x axis), indicating that the fake news tend to follow popular interest – at least in general terms. The dashed line shows the aggregated mean of the correlations, highlighting the aforementioned trend.

Figure 3 (right) shows the trends considering total mentions of entities in the fake news articles. The correlation is higher than that for closeness centrality, albeit more irregular (the Petrobras entity now has almost no correlation). Although more tests are

[2]https://trends.google.com/trends/

necessary, this result indicates that the graph metrics like closeness centrality might be more reliable indicators of entity relevance.

5. Discussion and Conclusion

This paper presented a graph-based approach to analyse the contents and evolution of fake news about Brazilian politics. The preliminary results presented here suggest that graph analysis can capture several aspects of the discourse space.

By using graph-based clustering we detected the main latent topics in the corpus. The goal was to match the evolution through time of the topic mix with real world events. As presented here, we are now starting to apply outlier detection algorithms to identify important periods. We are currently testing other outlier algorithms and developing a better methodology to match the outliers with real-world events. We expect that in the future this analysis will provide insights on the interplay between fake news and real events.

The analysis of the graph measurements for the entities over time also showed promising results. Using traditional clustering algorithms over the entity vectors containing the measured values, we were able to find clusters of related entities that correspond to real world political developments. We plan to also apply outlier detection over individual entities in the future.

Finally, in the most important preliminary result in this paper, we assessed how the graph measurements for some top entities correlate with popular interest (represented by Google Trends counts). The analysis shows a clear tendency of fake news on a given topic to vary based on recent popular interest. This suggest that fake news, at this moment, may be more of a reactive mechanism than a driver of the discourse. Our hypothesis is that real news would not show this marked trend, correlating more closely with popular interest, but we still do not have an adequate corpus to verify. If the hypothesis is verified, this information could be used to automatically identify unreliable sources of information.

References

Allcott, H. and Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36.

da F. Costa, L., Rodrigues, F. A., Travieso, G., and Boas, P. R. V. (2007). Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242.

Jang, S. M., Geng, T., Li, J.-Y. Q., Xia, R., Huang, C.-T., Kim, H., and Tang, J. (2018). A computational approach for examining the roots and spreading patterns of fake news: Evolution tree analysis. *Computers in Human Behavior*, 84:103–113.

Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*. ACM.

Monteiro, R. A., Santos, R. L. S., Pardo, T. A. S., de Almeida, T. A., Ruiz, E. E. S., and Vale, O. A. (2018). Contributions to the study of fake news in portuguese: New corpus and automatic detection results. In *Computational Processing of the Portuguese Language*, pages 324–334. Springer International Publishing.

Modelo Autorregressivo de Integração Adaptativa*

Arthur Ronald[1], **Rebecca Salles**[1], **Kele Belloze**[1], **Dayse Pastore**[1], **Eduardo Ogasawara**[1]

[1]CEFET/RJ - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

{arthur.garcia,rebecca.salles}@eic.cefet-rj.br

{kele.belloze,dayse.pastore}@cefet-rj.br, eogasawara@ieee.org

Abstract. *Several preprocessing techniques combined with time series models have been used to predict non-stationary time series. The study of the mathematical and statistical properties of the data and the preprocessing techniques can help in the adjustment of machine learning models. Such a study, however, may not be easily obtained. Linear models enable the interpretation of such properties. This article introduces and analyzes, based on proof of concept, a new linear model applied to stationary time series that are built by using adaptive normalization. The model allows the use of autoregressive models with sliding windows of data that preserve the properties of the original series, and allow the observation of its inertia. The model was able to present superior prediction performance to other linear models consolidated in the literature, specially in short-term horizons.*

Resumo. *Diversas técnicas de preprocessamento combinadas a modelos de séries temporais vêm sendo utilizadas para previsão de séries temporais não-estacionárias. O estudo das propriedades matemáticas e estatísticas dos dados e das técnicas de preprocessamento pode auxiliar no ajustamento de modelos de aprendizado de máquina. Tal estudo, entretanto, muitas vezes não é facilmente obtido. Modelos lineares, por sua vez, possibilitam a interpretação de tais propriedades. Este artigo introduz e analisa, por meio de prova de conceito, um novo modelo linear aplicado a séries estacionárias construídas com base em normalização adaptativa. O modelo viabiliza o uso de modelos autorregressivos em cenários de janelas deslizantes que preservam as propriedades da série original, e permitem acompanhar a sua inércia. O modelo foi capaz de apresentar desempenho de previsão superior a outros modelos lineares consolidados na literatura, principalmente em horizontes de curto-prazo.*

1. Introdução

No âmbito de previsões de séries temporais (ou, simplificadamente, séries), diversos métodos para modelagem requerem que a série seja estacionária, *i.e.*, possua média, variância e covariância constantes. Entretanto, para a maioria delas, a condição de não-estacionariedade é a regra enquanto a estacionariedade é a exceção. O trabalho seminal para modelagem integrada de séries resilientes à não-estacionariedade veio a partir do processo *Autoregressive Integrated Moving Average* ($ARIMA$), que usa a diferenciação como forma de obter a estacionariedade [Box et al., 2008].

*Os autores agradecem à FAPERJ, à CAPES (código 001) e ao CNPq pelo financiamento do projeto.

Posteriormente, outras abordagens de transformação de séries não-estacionárias em estacionárias surgiram e vêm sendo frequentemente associadas ao estado-da-arte em modelos de aprendizado de máquina [Salles et al., 2019]. No entanto, o uso direto dessas novas abordagens aplicadas a modelos de aprendizado de máquina pode ocultar as propriedades matemáticas e estatísticas dos dados, em virtude do ajustamento por hiperparâmetros, comumente aplicado no processo de construção de tais modelos. Em contraste, essas propriedades são bem estabelecidas e podem ser melhor observadas e interpretadas no contexto de modelos lineares. Este fato favorece o seu uso quando a análise do comportamento das séries temporais é um fator de interesse.

Neste contexto, este artigo introduz um novo modelo linear, denominado $ARAI$ (do inglês, *Autoregressive Adaptive Integration*), que combina os modelos autorregressivos com a normalização adaptativa [Ogasawara et al., 2010]. O modelo inova ao viabilizar o uso do $ARIMA$ em cenários de janelas deslizantes, comumente adotados no contexto de aprendizado de máquina. Cada janela preserva as propriedades da série original, e permite acompanhar o seu aspecto inercial. A análise deste modelo pode também contribuir para o emprego adequado da normalização adaptativa no cenário de aprendizado de máquina. O modelo foi avaliado com base na previsão de séries temporais socieconômicas e foi capaz de apresentar desempenho superior quando comparado a outros modelos lineares consolidados na literatura, como AR, MA e $ARIMA$, principalmente em horizontes de curto-prazo.

2. Referencial Teórico

Uma série pode ser compreendida como um processo estocástico composto por uma coleção y de variáveis randômicas y_t, tal que $|y| = n$ e y_n corresponde à observação mais recente. A série é classificada como estritamente estacionária se, para qualquer $(t_1, t_2, ..., t_k)$ e para qualquer h, a distribuição $(y_{t_1}, y_{t_2}, ..., y_{t_k})$ for idêntica à $(y_{t_{1+h}}, y_{t_{2+h}}, ..., y_{t_{k+h}})$ [Brockwell and Davis, 2016].

Essa condição é rígida e geralmente difícil de ser observada matematicamente, haja vista que, para a maioria das aplicações, as respectivas distribuições são desconhecidas a priori [Woodward et al., 2017]. Por causa disso, uma definição menos restritiva, denominada covariância-estacionária, foi estabelecida, em que uma série é considerada estacionária se possuir média, variância e covariância constantes.

2.1. ARIMA

Modelos $ARIMA(p, d, q)$ são compostos pelos processos de modelagem autorregressivo (AR) e média móvel (MA) (respectivamente representados por p e q). Além disso, tais modelos contam com um processo de diferenciação preliminar (I) (representado por d), desenvolvidos para lidar com a não estacionariedade. Dentro da família $ARIMA$, o processo autorregressivo $AR(p)$ estabelece que o valor corrente de uma série é função de seus p valores precedentes. Tal processo é descrito pela Equação 1, em que δ corresponde ao intercepto, p à ordem do modelo AR e ε_t ao erro, que idealmente deve se comportar como um ruído branco [Brockwell and Davis, 2016].

$$\mathrm{y}_t = \delta + \sum_{j=1}^{p} \phi_j \mathrm{y}_{t-j} + \varepsilon_t \quad (1)$$

Tradicionalmente, a metodologia usada para descrever o processo $ARIMA(p, d, q)$ é a Box-Jenkins, composta pelas fases de identificação, estimação e validação. A ordem d estabelece o nível de diferenciação a ser aplicado a priori na série temporal. Os valores das ordens p e q, quando aplicáveis, baseiam-se na interpretação dos gráficos das funções de autocorrelação e autocorrelação parcial [**?**].

No caso proposto por Brockwell and Davis [2016], o critério sugerido é a combinação p e q que minimize o valor da função AICc [Hurvich and Tsai, 1989]. Essa função é uma versão aprimorada do critério de informação de Akaike (AIC) [Akaike, 1992], que penaliza o resultado da função original a fim de evitar introdução de viés. Idealmente, quanto menor o valor da função AICc, melhor. Dessa maneira, o modelo se ajusta cada vez mais à série à medida que o valor da função se aproxima de $-\infty$ [Enders, 2015]. Assim, para cada combinação p e q, os coeficientes do processo $ARIMA$ são estimados de forma a minimizar o valor da referida função. Diante disso, a fim de auxiliar na automação do processo, Hyndman and Khandakar [2008] desenvolveram um algoritmo que otimiza a escolha dos parâmetros p, d e q do modelo $ARIMA(p, d, q)$.

2.2. Mapeamento do Processo AR como Regressão Múltipla

Uma subsequência é uma amostra contínua de uma série. A i-ésima subsequência de tamanho p em uma série y, representada por $seq_{p,i}(y)$, é uma sequência ordenada de valores $(y_i, y_{i+1}, \ldots, y_{i+p-1})$, onde $|seq_{p,i}(y)| = p$ e $1 \leq i \leq |y| - p$.

As janelas deslizantes de tamanho p consistem em explorar todas as subsequências de tamanho p de uma série. Formalmente, podem ser representadas por $sw_p(y)$, que corresponde a uma matriz A de tamanho $(|y| - p + 1) \times p$. Cada vetor-linha a_i em A é a i-ésima subsequência de tamanho p em y. Dado $A = sw_p(y)$, $\forall a_i \in A$, $a_i = seq_{p,i}(y)$. Vale observar que as janelas deslizantes organizam as colunas da matriz A de forma que a j-ésima coluna corresponda a uma defasagem da série original y por $(p - j)$ valores precedentes.

Considere uma matriz A formada por janelas deslizantes de tamanho $p + 1$ ($A = sw_{p+1}(y)$). Considere o mapeamento da matriz A por meio de uma relação $\mathcal{R}(A)$, na qual, os vetores-colunas a_j de A podem ser mapeados como uma dimensão x_j $(1 \leq j \leq p+1)$ em $\mathcal{R}(A)$. A dimensão x_{p+1} corresponde às observações da série enquanto as dimensões x_j $(\forall j \in [1, p])$ aos seus valores defasados. A Equação 1 pode ser reescrita por meio da Equação 2, de modo que o modelo autorregressivo AR pode ser interpretado como um problema de regressão múltipla, *i.e.*, os coeficientes $\phi_1, \cdots \phi_p$ podem ser aproximados pelos métodos de mínimos quadrados ou verosimilhança adotados no cálculo de regressão múltipla. O δ e ϵ_{p+1} são análogos aos da Equação 1, onde neste caso, ϵ_{p+1} corresponde ao conjunto de erros observados em toda a série em x_{p+1}, ou $\epsilon_{p+1} = (\varepsilon_{p+1}, \cdots, \varepsilon_{|y|})$ [Tsay, 2010].

$$x_{p+1} = \delta + \phi_1 x_1 + \cdots + \phi_p x_p + \epsilon_{p+1} \tag{2}$$

3. ARAI

O processo de construção do modelo ARAI é composto por três fases sequenciais: transformação da série temporal, remoção de *outliers* e modelagem.

Seja a matriz A formada por janelas deslizantes de tamanho $p+1$ sobre uma série y de tamanho n. Seja $a_i = (a_{i_1}, \cdots, a_{i_{p+1}})$ o *i*-ésimo vetor-linha ($1 \leq i \leq (|y|-p+1)$). A primeira fase consiste em transformar a matriz A em uma matriz B normalizada adaptativamente. Para cada $a_i \in A$, define-se a operação básica do $ARAI$ como uma diferenciação a qual se aplica uma função f computada a partir dos termos endógenos $(a_{i_1}, \cdots, a_{i_p})$, como descrito na Equação 3. A função f, descrita na Equação 4, é uma função de centralidade (média) sobre um vetor v. A função traz um aspecto inercial de acompanhamento da série. A função f aplicada sobre todas as linhas a_i produz uma matriz B normalizada adaptativamente [Ogasawara et al., 2010]. Ao final deste processo, a matriz B possui média 0 e variância σ^2. Tais propriedades foram observadas experimentalmente.

$$b_{i_j} = a_{i_j} - f(a_{i_1}, \cdots, a_{i_p}), \forall j \in [1, p+1] \tag{3}$$

$$f(v) = \frac{\sum_{j=1}^{|v|}(v_j)}{|v|} \tag{4}$$

Logo após a primeira fase, os *outliers* são removidos, desconsiderando os vetores-linha da matriz B nos quais um ou mais valores estejam fora do intervalo $[Q_1 - 1.5(IQR), Q3 + 1.5(IQR)]$. Este critério de identificação de *outliers* é comumente utilizado por *boxplots*, em que IQR corresponde ao intervalo interquartil.

A partir deste momento, tem-se a terceira fase (modelagem), onde encontram-se os coeficientes $\phi_1, \cdots, \phi_p$ que ajustam a Equação 2 ao se converter a matriz B em uma relação $\mathcal{R}(B)$. Uma vez previsto o valor $b_{i_{p+1}}$ proveniente das observações $b_{i_1}, \cdots, b_{i_p}$, basta somar o valor de $f(a_{i_1}, \cdots, a_{i_p})$ para se obter o valor $a_{i_{p+1}}$.

4. Avaliação Experimental

Para a avaliação do modelo $ARAI$, foram obtidas as quatro séries mensais mais usadas do IPEAData (http://www.ipeadata.gov.br), conforme descritas pela Tabela 1. A ordem p do $ARAI(p)$ é definida variando-se de 2 a 12 (do menor tamanho de janela possível ao número de meses que compõem um ano), definindo como modelo final aquele que obtiver o menor valor para o $AICc$ para um conjunto de treinamento. Os modelos foram avaliados usando como métrica o erro quadrático médio (MSE) para horizontes de 1 a 12 (um ano de previsão).

Tabela 1. Séries analisadas pelo processo ARAI

Série - periodicidade mensal	Treino	Teste
Salário mínimo real	07/1994 - 03/2018	04/2018 - 03/2019
Índice de Preços ao Consumidor Amplo (IPCA)	07/1994 - 03/2018	04/2018 - 03/2019
Índice de desemprego	12/1984 - 02/2018	03/2018 - 02/2019
Produto Interno Bruto (PIB)	07/1994 - 03/2018	04/2018 - 03/2019

A Figura 1 apresenta o MSE das previsões feitas por ($ARAI$, $ARIMA$, AR e MA) para as quatro séries analisadas por horizonte de previsão. No caso do salário mínimo real (Figura 1.a), o $ARAI$ apresentou acurácia superior ao $ARIMA$ até o horizonte

9. A partir deste ponto, a acurácia degenerou ficando próxima aos modelos AR e MA. De modo análogo, no caso do índice de desemprego (Figura 1.b), o $ARAI$ apresentou melhor acurácia até horizonte 8. A partir do mês 9, a acurácia do $ARAI$ ficou semelhante à dos processos AR, MA e $ARIMA$.

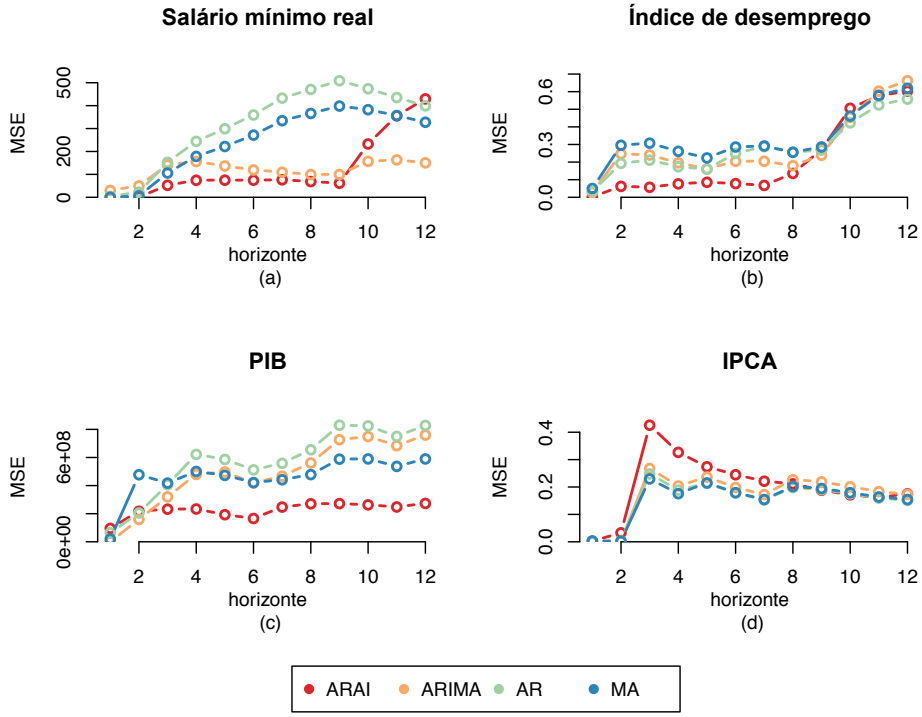


Figura 1. Séries avaliadas com os respectivos MSE

No caso do PIB (Figura 1.c), o $ARAI$ foi dominante, apresentando melhor acurácia para quase todos os horizontes estudados. Em contrapartida, no caso do IPCA (Figura 1.d) para o horizonte de tamanho 3, o $ARAI$ teve uma acurácia inferior. Conforme o horizonte foi aumentando, as perdas provenientes da observação 3 são recuperadas, aproximando-se dos demais modelos.

Em função deste comportamento no IPCA, foi feita uma análise adicional. Observou-se que, no mês de junho de 2018 (observação 3), a inflação foi 1,26%, impactada pela greve dos caminhoneiros. Nos doze meses precedentes, a inflação média foi 0,235% com desvio padrão de 0,17%, sendo a mínima e a máxima de -0,23% e 0,44%, respectivamente. A Figura 2.a apresenta a contribuição individual do erro ao quadrado para cada previsão. Nota-se que para todos os modelos, a acurácia da previsão foi afetada pela observação 3, sendo mais impactante no $ARAI$. Entretanto, o $ARAI$ produziu previsões competitivas para as observações de 4 a 9. Nos demais modelos, as previsões tiveram acurácia inferior para as observações deste intervalo.

Estudando-se mais profundamente a previsão da observação 3, identificou-se que o *outlier* afetou o valor inercial usado para normalização. Por conta disto, avaliou-se em vez da média a mediana como função inercial (Equação 4). Neste caso, o impacto do *outlier* é minimizado, alcançando-se previsões próximas àquelas registradas pelos processos AR, MA e $ARIMA$ (Figura 2.b), sem deteriorar as previsões entre 4 e 9, o que fez esta adaptação no $ARAI$ apresentar melhor desempenho em relação aos demais modelos.

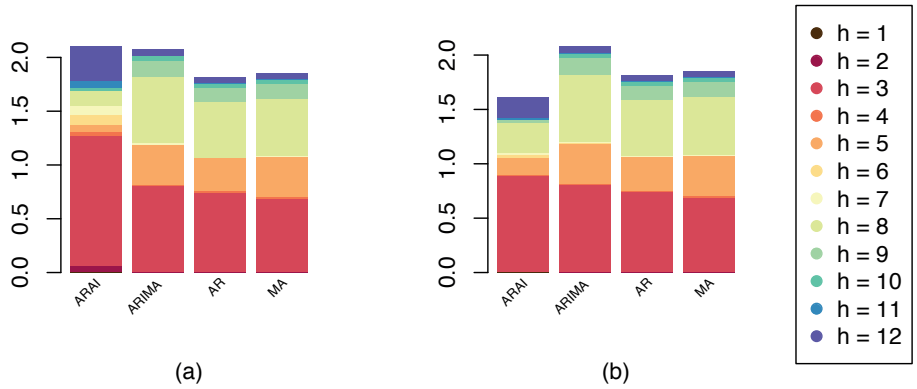


Figura 2. Erro ao quadrado associado ao respectivo horizonte para a série IPCA

5. Conclusão

Neste artigo foi apresentado um modelo linear Autorregressivo de Integração Adaptativa. Uma característica interessante desta abordagem é que, diferentemente dos processos clássicos de diferenciação no qual a série transformada é bem diferente da série original, o formato da série em cada janela preserva semelhanças em relação à série original, uma vez que a série é subtraída em relação à sua função inercial.

O modelo abre espaço para o estudo das propriedades estatísticas da normalização adaptativa no contexto de modelos lineares, o que pode auxiliar no desenvolvimento do seu emprego no cenário de aprendizado de máquina. Como trabalhos futuros, tem-se o estudo da acurácia das previsões do $ARAI$ em horizontes maiores e a prova de que os erros ϵ_{p+1} do modelo $ARAI$ se caracterizam como ruído branco.

Referências

Akaike, H. (1992). Information Theory and an Extension of the Maximum Likelihood Principle. In Kotz, S. and Johnson, N. L., editors, *Breakthroughs in Statistics: Foundations and Basic Theory*, Springer Series in Statistics, pages 610–624. Springer New York, New York, NY.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (2008). *Time Series Analysis: Forecasting and Control*. Wiley, Hoboken, N.J, 4 edition.

Brockwell, P. J. and Davis, R. A. (2016). *Introduction to Time Series and Forecasting*. Springer International Publishing, Cham, 3 edition.

Enders, W. (2015). *Applied Econometric Time Series*. Wiley, 4 edition.

Hurvich, C. and Tsai, C. (1989). Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307.

Hyndman, R. and Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 27(3):1–22.

Ogasawara, E., Martinez, L., De Oliveira, D., Zimbrão, G., Pappa, G., and Mattoso, M. (2010). Adaptive Normalization: A novel data normalization approach for non-stationary time series. In *Proceedings of the International Joint Conference on Neural Networks*.

Salles, R., Belloze, K., Porto, F., Gonzalez, P., and Ogasawara, E. (2019). Nonstationary time series transformation methods: An experimental review. *Knowledge-Based Systems*, 164:274–291.

Tsay, R. S. (2010). *Analysis of Financial Time Series*. Wiley, Cambridge, Mass, 3 edition.

Woodward, W. A., Gray, H. L., and Elliott, A. C. (2017). *Applied Time Series Analysis with R*. CRC Press, Boca Raton, 2 edition.

Industrial Paper: Large-scale Record Linkage of Web-based Place Entities

Vinícius M. R. Cousseau[1,2], **Luciano Barbosa**[2]

[1]In Loco, Recife – PE – Brazil

[2]Centro de Informática, Universidade Federal de Pernambuco, Recife – PE – Brazil

vinicius.cousseau@inloco.com.br, luciano@cin.ufpe.br

Abstract. *Extracting data about entities from the Web has become commonplace in the industry and academia alike. Web-based entities, however, are inherently noisy and, as such, introduce several normalization issues which must be attended to in order to maintain a clean database. Record linkage, which refers to the detection of replicated datum from possibly multiple sources, is one of the most critical of those issues. This paper presents a practical approach for solving the record linkage problem in the places data domain at an industrial scale, displaying both a model which reaches a normalized Gini coefficient of 0.92, and an architecture that supports large-scale processing.*

1. Introduction

The creation and management of databases built upon web-based data is a process which suffers from several issues pertaining to a noisy and unstructured nature. While the amount of information covered by the web grows steadily, only 52% of its websites display their data in some structured manner[1]. On top of that, one may also encounter well-structured but ill-defined *datum*, which not only is harder to treat, but also lessens its informational and commercial value.

One of the most prominent normalization issues that must be addressed in this context is record linkage, which refers to the detection and deduplication of redundant data in databases. This field of research has evolved considerably, since most of the Web is currently open to user input and there is usually a plethora of websites displaying information about the same entities but in different forms.

Our domain consists of data about places around the globe, gathered from a focused web crawler. We define a place in our database as a direct representation of a physical location in the real world which has well-defined boundaries, a purpose, a considerable size, and is a source of context.

The most relevant place fields for the work herein described are **ID** (a unique textual identifier), **Name**, **Geo Location** (expressed in latitude and longitude coordinates), **Address**, **Phone**, **Homepage**, **Labels** (which are the places' categories such as restaurant or store), and **Parentage** (which is an ID of another place which encompasses the current one, if any).

We should note that places are particularly complex entities, since they are tightly coupled with a location and several of its fields are subjective. The places' **Geo Location**

[1]https://w3techs.com/technologies/overview/structured_data/all

field, for instance, presents issues due to a common lack of precision resulting from the geocoding - address to latitude and longitude translation - strategies utilized by websites. Moreover, having redundant places in our database could result in catastrophic results for our company, due to resultant product malfunctions.

One common strategy for approaching the linkage problem, mainly in the industry, is the development of deterministic systems based on rule chaining [Berjawi 2017]. Albeit quick to implement and effective, these systems are domain-dependant and do not scale well, as the number of chained rules grows indefinitely. There are some techniques to reduce temporal constraints, such as record blocking [Christen 2012].

The work by [Dalvi et al. 2014] is the one that most closely resembles ours, since they also investigate the record linkage problem in the places domain, proposing a dynamic programming technique based on the detection of core words for each place's name in order to match pairs. Significant gains should be expected when using machine learning models for record linkage [Wilson 2011], but it is important to note that the record linkage problem has often highly imbalanced class distributions, akin to an anomaly detection task. Hence, any chosen statistical model and metrics must account for class imbalance.

On top of these issues, we also had to take into consideration the need for a resilient and scalable architecture. For reference, as of the writing of this paper, our database was being queried approximately 700 million times on a daily basis, and it held information about more than 28 million places globally. This greatly restricted available technique choices for us, and led to some interesting trade-offs.

This paper then describes our experience in dealing with duplicated places in an industrial scale. Its contribution is twofold: it unravels architecture choices and trade-offs in dealing with on-line and off-line approaches to the record linkage problem in parallel, and presents our algorithms and machine learning models applied to the task, alongside insights learned about this problem and its domain.

2. Proposed Method

In the places domain, the record linkage problem may be defined by Research Problem (`RP`) I, which is a relaxed version of `RPII`:

> **Research Problem I** *Given a pair of place entities, how do we detect if they represent the same real-world place?*
>
> **Research Problem II** *Given a set of place entities, how do we detect if they represent the same real-world place?*

In order to translate a model which works on `RPI` to solve `RPII`, one must simply represent each positive pair output as connected nodes in a graph, and negative outputs as unconnected ones. The connected components of said graph will represent the replicated data sets, thus fulfilling the requirements of `RPII`. The techniques described in this section solve `RPII` by applying this incremental procedure.

Our first approach was deterministic and utilized the concept of a place's name being usually defined by a set of core words and a set of background ones [Dalvi et al. 2014]. We name it *WordRelevanceHeuristics* or *WRH* for short. As we had to take both off-line and on-line contexts into consideration during our development process, the relative naivety of this approach allowed us to quickly deploy a working version to both fronts.

Table 1. *PairwiseRandomForest*'s features and their descriptions

Feature	Description
MostRelevantWordJaro	Jaro Winkler similarity between core words
NameSoftTfIdf	Soft TF-IDF between normalized names
AddressSoftTfIdf	Soft TF-IDF between normalized street names
LabelsJaccard	Jaccard similarity between label lists
Distance	Distance in meters between two geo locations
HomepageMatches	Exact homepage match
ParentageRelationship	Exact match between an id and a parentage
SiblingRelationship	Exact match between parentages

WRH derives the core word concept from the fact that those words are usually the most unique ones in a name, and thus, rarer. We also restrict the detection to a single core word. Translated to code, this means that each place's core word is the one with the least global term frequency value. These words, however, may simply be localized words, such as street names, neighborhoods or local folklore, which are not good candidates for core words. To account for that, our algorithm prunes localized words by applying experimentally defined thresholds to TF-IDF values calculated only inside each place's Geohash[2], instead of globally.

Afterwards, for each place pair, the core word of each of their members is compared via exact string matching. Geographic distance, parentage and label similarity rules are then taken into account to produce a final result, indicating whether or not a given pair represents duplicated places. This approach is still very sensitive to typing errors and acronyms, which are almost always detected as the core word due to their uniqueness. As is the case with deterministic systems, it also suffers from being hard to maintain without a good domain knowledge, which in turn is undesirable in an industrial setting.

During production usage, *WRH*'s recall was deemed too low, and we decided to follow a second approach which leveraged the capabilities of supervised learning. We opted for a Random Forest model, since it tends to excel at handling skewed data and finding a good bias-variance trade-off without feature scaling. We name this model *PairwiseRandomForest* or *PRF*, and are utilizing it in our current production environment.

The *PRF* model features are described at Table 1. These are all pairwise features, i.e. for each possible place pair, each feature is computed by comparing attributes from both of the places. As the reader may notice, we still make use of the place's core word, but this time only as a single feature in the model. We also include two Soft TF-IDF [Moreau et al. 2008] comparisons since the Soft TF-IDF algorithm compares words which surpass a secondary similarity threshold, thus being more resilient to typing errors, different verbal tenses and acronyms. Finally, parentage and homepage comparisons are included as one-hot encoded categorical features.

In conclusion, each place pair is represented as a row vector $v \in \mathbb{R}^{15}$, which is then processed by the model. The random forest's output and an experimentally defined classification threshold α are utilized in tandem to generate the final binary classification for each input *datum* (`RPI`), and the aforementioned connected components analysis is

[2]http://geohash.org/

applied to generate the replicated place clusters (RPII).

2.1. Pre-processing and Trade-offs

Given that our approaches attempt to solve RPI and then adapt the results to RPII, generating possible place pairs is one of the key pre-processing steps in our pipeline. With a database containing upwards of 28 million records, performing quadratic combinations over the whole database would be too computationally and economically expensive, and as such, we utilize distributed computing and opt for the implementation of blocking techniques to reduce the number of potential pairs.

A labeled diagram for our final architecture utilizing the *PRF* model, including its on-line version and translation from RPI to RPII, can be seen at Figure 1.

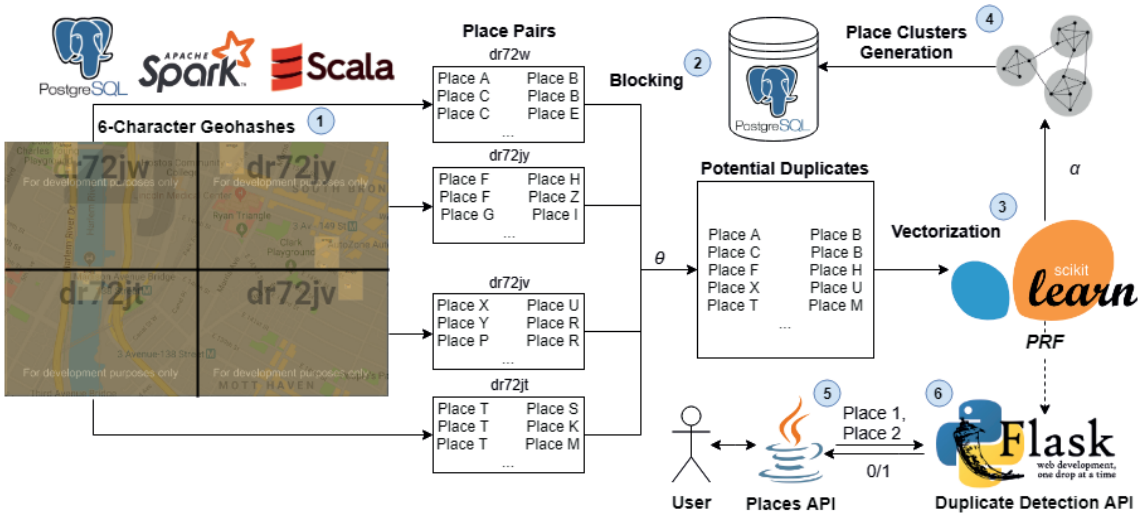


Figure 1. Diagram of our final architecture, including off-line and on-line approaches with labels. Map image made with http://geohash.gofreerange.com/.

Our blocking algorithm firstly subdivides the space into 6-character long Geohashes, each having an area of approximately $744m^2$ and a diagonal distance of approximately $1364m$, as seen in step (1). Then, for each Geohash, all possible place pairs are generated and filtered by comparing both place names with a Jaro Winkler string similarity, using a coefficient that is low enough to preserve the class distribution between duplicates and non-duplicates, and high enough to prune obvious non-duplicate pairs (2). This threshold was experimentally defined as 0.7. Place pairs are then converted into pairwise features in a separate feature engineering step, and categorized using *PRF* (3). The final off-line results are generated by the detection of connected components among the positive pairwise outputs and fed back to our database (4).

Performing quadratic operations when blocking on each Geohash would still not fulfill the needs of a real-time responsive service. Hence, in order to generate possible pairs in an on-line environment for each new place, we first query one of our APIs to find all nearby places in a $250m$ radius, limiting the query result to 150 places (5). The generated pairs are then passed through our name similarity blocking algorithm and sent to a dedicated API to respond to record linkage queries, using the trained model (6).

3. Experiments

The first step in our experiments was the definition of a silver standard over which we could iterate. With manual and crowdsourced approaches to this dataset construction having failed, we decided to utilize the same blocking technique described in the previous section to generate possible pairs for our dataset. For each of these pairs, we then filtered out all the ones without the **Phone** attribute, present in roughly 40% of the dataset entities, and utilized a direct match of the phone digits to indicate whether or not the pair was positive or negative. We then manually analyzed a few troublesome cases and fixed them.

This approach still preserved the estimated class distribution, and provided us with a dataset containing 597452 place pairs, 572560 of those being negative samples: a $24 : 1$ negative to positive ratio. It is important to note that we decided to run both of our experiments for Brazilian places only, since different languages could imply different feature importance values.

The dataset was then divided into training and test datasets using a stratified split of 70% and 30%, respectively. Afterwards, we undersampled the training set with Tomek links and then oversampled the positive samples with SMOTE. The *PRF* model was trained on top of this training dataset, while *WRH* did not require any kind of training. The hyperparameters for *PRF* all follow the default values from `scikit-learn v0.20.3`, apart from the maximum tree depth, which was manually optimized to 23. In these experiments, α is manually set to 0.95.

Table 2 shows the results for the execution of the two strategies on top of the test dataset. The balanced accuracy score metric takes the class imbalance issue into account, preventing the majority of negative samples from biasing the results. We added precision and recall as means of completeness, even though they are not good evaluation metrics for an imbalanced scenario.

Table 2. Results on top of our test dataset, with $\alpha = 0.95$

Metric	*WRH*	*PRF*
Balanced Accuracy Score	0.51	**0.87**
$F_{\beta=0.5}$ (Positive)	0.21	**0.42**
Precision (Positive)	0.56	0.56
Recall (Positive)	0.06	**0.22**

The $F_{\beta=0.5}$ score was chosen in spite of the $F1$ score to better reflect the increased importance of false positives in our context, when compared to false negatives. Regardless, it is still highly affected by the class imbalance, i.e. due to the higher density of negative samples, there will naturally be several more false negatives than false positives as the size of the dataset grows.

We also computed the normalized Gini coefficient for the *PRF* model. This score presents itself as one of the best ways to measure performance of classifiers in a highly skewed dataset, by measuring the degree of inequality among the results. The *PRF* model achieved a normalized Gini coefficient of 0.92, very close to the optimal value of 1.

By comparing both models, we clearly see the predicted effects of a much higher recall, a 366% gain. The absent precision gain for the second model with the chosen α

value is a result of the precision-recall trade-off that we were willing to adhere to. By adjusting α according to the model's learning curve, however, we are able to achieve a precision of 0.72, a 28% gain, with *WRH*'s recall value of 0.06.

Upon manual investigation, we discovered that rejecting address comparisons is detrimental, since there are many seemingly replicated places with the same name near each other that are actually two real separate locations from a chain store. Indeed, a feature importance analysis by permutation shows that that the **MostRelevantWordJaro**, **NameSoftTfIdf**, and **AddressSoftTfIdf** features are the most crucial ones, respectively.

Furthermore, to assess the scalability of our model, we recorded execution times of our distributed algorithms, which run on a recurring basis on top of all our database containing approximately 252Gb of data stored in the parquet format. We utilized 7 AWS m4.xlarge instances, each having 16Gb of memory and 4vCPUs. The pipeline achieved an average run time of 93 minutes to process the whole database, with the generation of over 2 million potential pairs to be processed by our model. The classification and clusterization steps were extremely fast, and always took less than a minute to process all generated pairs in each experiment.

4. Conclusions

In this paper, we presented the evolution of an approach to detect replicated records in an increasingly growing database of web-based place entities, with upwards of 28 million records. These records are inherently noisy, and we had to deal with several issues pertaining to this nature during the development of our methods.

As a result of our work, we were able to both fulfill the current company needs and overcome the technical challenges in the area, making our architecture fully operational in a production environment. We expect that this work should be of use to researchers and engineers in the academia and industry, whether they are dealing directly with places data or not. As future steps, we intend to dig deeper into more recent techniques, such as the development of spatial embeddings for usage in deep neural network architectures.

References

Berjawi, B. (2017). *Integration of Heterogeneous Data from Multiple Location-Based Services Providers: a Use Case on Tourist Points of Interest*. PhD thesis.

Christen, P. (2012). A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 24(9):1537–1555.

Dalvi, N., Olteanu, M., Raghavan, M., and Bohannon, P. (2014). Deduplicating a places database. In *Proceedings of the 23rd international conference on World wide web - WWW 14*. ACM Press.

Moreau, E., Yvon, F., and Cappé, O. (2008). Robust similarity measures for named entities matching. In *Proceedings of the 22nd International Conference on Computational Linguistics - COLING 08*. Association for Computational Linguistics.

Wilson, D. R. (2011). Beyond probabilistic record linkage: Using neural networks and complex features to improve genealogical record linkage. In *The 2011 International Joint Conference on Neural Networks*. IEEE.

Descoberta automática de restrições de negação confiáveis

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. *Restrições de negação (RNs) expressam regras que identificam inconsistências em um banco de dados. Compô-las, no entanto, é uma tarefa onerosa. Nós propomos um método que descobre RNs com base em evidências extraídas das tuplas de um conjunto de dados. Nosso método descobre RNs confiáveis, mesmo que o conjunto de dados contenha erros. Nossos experimentos com dados reais mostram que é possível encontrar RNs que, com alta precisão e revocação, apontam para inconsistências dos dados de entrada.*

Abstract. *Denial constraints (DCs) express rules that identify inconsistencies in a database. Their design, however, is an onerous task. We propose a method that discovers DCs based on evidence of the tuples of a dataset. Our method discovers reliable DCs, even if the dataset has errors. Our experiments with real data show that it is possible to find DCs that, with high precision and recall, point to inconsistencies of the input data.*

1. Introdução

Manter um banco de dados livre de erros é computacionalmente caro. Erros incluem valores atípicos de domínio, duplicatas, violações de padrões (e.g., expressões regulares), e violações de restrições de integridade [Rekatsinas et al. 2017]. Atender essa última classe de erros é uma tarefa particularmente desafiadora. Restrições de negação (RNs) fornecem um formalismo capaz de expressar uma ampla gama de restrições e regras de negócio. Cada RN define relacionamentos entre predicados relacionais que identificam combinações de valores de atributo consideradas inconsistentes. Uma instância de relação viola uma RN φ se nela existir qualquer par de tuplas que simultaneamente satisfaça todos os predicados de φ. Nesse caso, dizemos que a instância está inconsistente (ou suja) em relação à RN φ. Recentemente, ferramentas do estado da arte em limpeza de dados têm sido baseadas no formalismo de RNs [Rekatsinas et al. 2017].

A construção de RNs requer análises da estrutura e conteúdo do banco de dados. Se feita por usuários, tal tarefa requer expertise e tempo; e está propensa a erros, considerando quão complexos e dinâmicos são os bancos de dados modernos. A descoberta automática de RNs é uma alternativa pois existem algoritmos eficientes que encontram todas as RNs mantidas em um conjunto de dados. Porém existem entraves que limitam a adoção da descoberta automática de RNs em cenários reais. Primeiro, as RNs são tão confiáveis quanto os dados de onde foram descobertas. Como obter dados 100% corretos é muitas vezes utópico, a descoberta deve ser ajustada a fim de acomodar possíveis erros. Segundo, o número de RNs descobertas cresce exponencialmente com o número de atributos da relação de entrada. Mesmo que a descoberta seja feita com um conjunto 100% correto, muitas RNs se mantêm apenas ao acaso.

Neste trabalho, mostramos que os resultados da descoberta de RNs em conjunto de dados limpos (corretos) é significativamente diferente dos resultados da descoberta de RNs em conjunto de dados sujos (inconsistentes). Nessa premissa, elaboramos um método para descoberta de RNs que, a partir de dados sujos, aproxima seus resultados daqueles que seriam obtidos caso o conjunto de dados limpos estivesse disponível. Baseado na significância estatística das RNs, nosso método seleciona um subconjunto de RNs capaz de detectar com alta precisão e revocação inconsistências nos conjuntos de dados.

2. Trabalhos relacionados

A descoberta de meta-informações a partir dos dados disponíveis tem ganhado notoriedade na área de banco de dados [Abedjan et al. 2015]. Estamos particularmente interessados em descobrir meta-informações na forma de RNs: dois algoritmos podem nos ajudar com tal objetivo. O algoritmo FASTDC compara pares de tuplas para calcular um conjunto de evidências que direciona a busca por RNs [Chu et al. 2013]. O algoritmo BFASTDC é computacionalmente mais eficiente que o algoritmo FASTDC pois aprimora substancialmente o cálculo de evidências [Pena and de Almeida 2018]. FASTDC e BFASTDC podem adicionalmente ser adaptados para descobrir RNs que se mantêm parcialmente no conjunto de dados. A saída desses algoritmos deve ser filtrada e validada por um especialista porque nem todas as RNs que se mantêm em uma instância são igualmente úteis. Além disso, o parâmetro de parcialidade das RNs deve ser configurado manualmente, o que pode tornar a descoberta incorreta. RNs corretas servem de entrada para ferramentas de limpeza de dados, como [Rekatsinas et al. 2017], pois violações de RNs geralmente apontam para dados inconsistentes.

3. Preliminares e descrição do problema

Seja r uma instância de relação com esquema $\mathsf{R}(\mathsf{A}_1, ..., \mathsf{A}_n)$, t uma tupla de r, e $\mathsf{O} = \{=, \neq, <, \leq, >, \geq\}$ um conjunto de operadores (com negação fechada). Um predicado p expressa uma comparação da forma $\mathsf{t}_x.\mathsf{A}_i$ o $\mathsf{t}_y.\mathsf{A}_j$: $\mathsf{A}_i, \mathsf{A}_j \in \mathsf{R}$; $\mathsf{t}_x, \mathsf{t}_y \in \mathsf{r}$; e $\mathsf{o} \in \mathsf{O}$. O espaço de predicados P é o conjunto de predicados que podem formar RNs sobre R.

Definição 1 (Restrição de negação (RN))**.** *Uma restrição de negação φ sobre a instância r é uma expressão da forma φ: $\forall \mathsf{t}_x, \mathsf{t}_y \in \mathsf{r}, \neg(\mathsf{p}_1 \wedge \ldots \wedge \mathsf{p}_m)$ onde φ* se mantêm *em r se e somente se para qualquer par de tupla $\mathsf{t}_x, \mathsf{t}_y \in \mathsf{r}$ pelo menos um dos predicados $\mathsf{p}_1, \ldots, \mathsf{p}_m$ é falso. Uma RN φ_1 é* minima *se não existe uma RN φ_2 tal que ambas φ_1 e φ_2 se mantenham em r, e que os predicados de φ_2 sejam um sub-conjunto de φ_1.*

Consideremos duas versões da instância r. A instância r_{limpa} é completa e correta; e a instância r_{suja} é qualquer versão incompleta e incorreta de r_{limpa}. Em r_{suja} podem haver diversos erros e inconsistências que não estão presentes em r_{limpa}. Denotamos por $\Sigma_{\mathsf{r}_{limpa}}$ o conjunto de todas as RNs mínimas que se mantêm em r_{limpa}. Ao impormos $\Sigma_{\mathsf{r}_{limpa}}$ sobre r_{suja} encontramos todas as potenciais inconsistências de r_{suja} (detectáveis pelo formalismo RN). Tal abordagem não é viável por dois motivos. Primeiro, a obtenção de r_{limpa} é uma tarefa cara, ou até mesmo irrealista. Se não temos a instância r_{limpa}, não temos o conjunto de RNs $\Sigma_{\mathsf{r}_{limpa}}$. Segundo, mesmo que seja possível obter uma instância correta r_{limpa}, muitas RNs do conjunto $\Sigma_{\mathsf{r}_{limpa}}$ se mantêm ao acaso e não expressam uma regra do mundo real. Faz se necessário uma seleção de RNs de $\Sigma_{\mathsf{r}_{limpa}}$. Nossa hipótese é

que é possível descobrir um conjunto de RNs Σ_{conf} que se aproxime de $\Sigma_{\mathsf{r}_{limpa}}$, a partir de instâncias r_{suja}. Em particular, as RNs em Σ_{conf} devem ser *confiáveis*, ou seja, encontrar as reais inconsistências de r_{suja}.

4. RN parcial, sua significância estatística, e confiabilidade

Conforme a definição 1, uma RN φ é válida em r se não existir um único par de tuplas em r que viole φ. Como utilizamos dados potencialmente sujos para descoberta de RNs, precisamos relaxar o critério de satisfação de uma RN. Mesmo que uma RN seja violada por alguns pares de tuplas de uma instância, ela ainda pode ser considerada válida. Em outras palavras, uma RN é *parcialmente* válida em r. Utilizamos a proporção entre o número de pares de tuplas que violam uma RN φ e o número total de pares de tuplas de uma instância r para quantificar o *grau de parcialidade* (gp) de uma RN φ em r.

Definição 2 (RN parcial)**.** *Dado um limite de erro ε, $0 \leq \varepsilon < 1$, uma RN φ é ε-parcial em r se e somente se seu grau de parcialidade $gp(\varphi, \mathsf{r})$ for menor que ε.*

É possível descobrir RNs e RNs parciais a partir de evidências geradas pelos pares de tuplas de um conjunto de dados. Uma *evidência* $\lambda_{\mathsf{t}_x,\mathsf{t}_y}$ é o conjunto de predicados que o par de tuplas $\mathsf{t}_x, \mathsf{t}_y$ satisfaz, i.e., $\lambda_{\mathsf{t}_x,\mathsf{t}_y} = \{\mathsf{p} \mid \mathsf{p} \in \mathsf{P}, \mathsf{t}_x, \mathsf{t}_y \models \mathsf{p}\}$. Pares de tuplas diferentes podem gerar a mesma evidência. Na prática, a quantidade de evidências distintas é apenas uma fração da quantidade total de pares de tuplas dos conjunto de dados. O conjunto de evidências Λ_{r} é composto por toda evidência de r. Utilizamos a função $\mathsf{multi}(\lambda)$ do tipo $\Lambda \rightarrow \mathbb{N}$ para retornar a multiplicidade da evidência λ no conjunto Λ. A multiplicidade de um conjunto de evidências é dada por $\|\Lambda\| = \sum_{\lambda \in \Lambda} \mathsf{multi}(\lambda)$. Cada evidência representa um relacionamento entre os predicados de P e o conjunto de pares de tuplas com a mesma assinatura sobre P. A partir da definição 1 percebemos que se uma evidência λ satisfaz os predicados $\{\mathsf{p}_1, \ldots, \mathsf{p}_m\}$, qualquer RN contendo pelo menos um predicado de $\{\bar{\mathsf{p}}_1, \ldots, \bar{\mathsf{p}}_m\}$ não pode ser violada pelos pares de tuplas que produziram a evidência λ. Os algoritmos de descoberta de RNs calculam o conjunto de todas as evidências do conjunto de dados, e então buscam por conjuntos de cobertura para o conjunto de evidências. A negação destes conjuntos são RNs mínimas [Chu et al. 2013]. Uma RN parcial φ deriva de uma cobertura parcial, para qual existem evidências que violam φ. Nos algoritmos FastDC e BFastDC, o limite de evidências que podem violar RNs parciais são definidas pelo usuário por meio do parâmetro limite de erro ε.

A evidência gerada por um par de tuplas errôneo tem um traço diferente de sua equivalente gerada pelo par de tuplas correto. Assim, erros degradam o conjunto de evidências correto, e a multiplicidade de seus elementos. Para ilustrar esse comportamento, obtemos o conjunto de dados *Hospital* (mais detalhes na Secção 5) em versões limpa e suja, e calculamos o conjunto de suas evidências. Na Figura 1, para cada evidência λ de $\Lambda_{\mathsf{Hospital}_{limpa}}$ plotamos no eixo Y a multiplicidade de λ, e a multiplicidade de λ em $\Lambda_{\mathsf{Hospital}_{suja}}$, caso λ também esteja em $\Lambda_{\mathsf{Hospital}_{suja}}$. Observamos que a maioria das evidências de $\Lambda_{\mathsf{Hospital}_{limpa}}$ também está em $\Lambda_{\mathsf{Hospital}_{suja}}$, com menores valores de multiplicidade. A variação é maior em direção à calda da distribuição de $\Lambda_{\mathsf{Hospital}_{suja}}$, onde algumas evidências desaparecem e algumas evidências ganham maior multiplicidade. Além disso, em $\Lambda_{\mathsf{Hospital}_{suja}}$ há milhares de evidências espúrias não presentes em $\Lambda_{\mathsf{Hospital}_{limpa}}$. Por exemplo, cerca de um terço das evidências de $\Lambda_{\mathsf{Hospital}_{suja}}$ têm multiplicidade unitária. Ainda assim, a tendência dos dois conjuntos de evidência é similar.

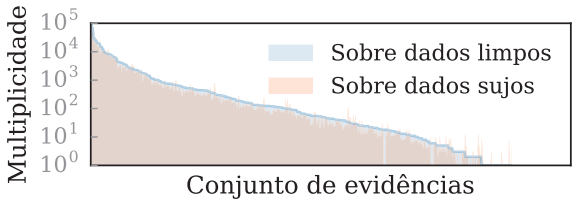


Figura 1. Multiplicidades das evidências de $\Lambda_{\mathsf{Hospital}_{limpa}}$, e correspondentes em $\Lambda_{\mathsf{Hospital}_{suja}}$. Eixo X em função das evidências de $\Lambda_{\mathsf{Hospital}_{limpa}}$.

A degradação do conjunto de evidências impacta diretamente na qualidade e quantidade das RNs descobertas. Com base na definição 2 observamos o seguinte. Considere uma RN mínima $\varphi_1 : \forall \mathsf{t}_x, \mathsf{t}_y \in \mathsf{r}, \neg(\mathsf{p}_1 \wedge \mathsf{p}_2)$ de $\Sigma_{\mathsf{r}_{limpa}}$. Sem perda de generalidade, existem dois cenários possíveis ao impormos φ_1 sobre uma instância r_{suja}. Primeiro, não há violações de φ_1, logo, φ_1 é uma RN exata em r_{suja}. Segundo, há violações de φ_1, logo, φ_1 é uma RN parcial em r_{suja}. Nesse caso, se estivermos descobrindo RNs exatas em r_{suja}, encontraremos uma especialização de φ_1, digamos $\varphi_1' : \forall \mathsf{t}_x, \mathsf{t}_y \in \mathsf{r}, \neg(\mathsf{p}_1 \wedge \mathsf{p}_2 \wedge \ldots)$. Quando a busca encontra a RN candidata φ_1, ainda existirão evidências para serem cobertas. O algoritmo então adiciona predicados à φ_1, até que todas as evidências sejam cobertas. Adicionar predicados à RN φ_1 camufla os pares de tuplas que violam φ_1 já que a especialização φ_1' não encontrará as violações de φ_1. Além disso, a busca atinge caminhos mais longos na árvore de busca, o que aumenta o número de RNs candidatas.

Em cenários reais, o número de restrições de integridade que um banco de dados deve manter é relativamente pequeno, e.g., unidades ou dezenas. No entanto, o número de RNs descobertas em conjunto de dados reais alcança facilmente a casa dos milhares, mesmo em dados limpos. Esse número é fruto do espaço de busca de RNs que cresce exponencialmente em função do número de predicados $|\mathsf{P}|$. Neste trabalho, medimos a utilidade de uma RN com base na medida *cobertura*, proposta em [Chu et al. 2013]. Tal medida expressa a *significância estatística* de uma RN com base na soma ponderada do número de predicados que cada par de tuplas satisfaz. Quanto maior o número de pares de tuplas que satisfazem números de predicados próximos à $|\varphi| - 1$, maior a cobertura de uma RN. Em nossos experimentos, as RNs que apontavam para inconsistências reais normalmente tinham as maiores coberturas dentre as RNs descobertas.

Um importante resultado da degradação de evidências pode ser visto na quantidade de RNs descobertas, e na distribuição dos valores de cobertura das RNs. A Figura 2 mostra, em ordem decrescente, os valores de coberturas das RNs em $\Sigma_{\mathsf{Hospital}_{limpa}}$ e $\Sigma_{\mathsf{Hospital}_{suja}}$. A quantidade de RNs em $\Sigma_{\mathsf{Hospital}_{suja}}$ é ordem de magnitude maior que a quantidade de RNs em $\Sigma_{\mathsf{Hospital}_{limpa}}$. Uma única RN em $\Sigma_{\mathsf{Hospital}_{limpa}}$ pode ter várias especializações em $\Sigma_{\mathsf{Hospital}_{suja}}$. Os valores de cobertura dessas especializações permeiam diferentes intervalos porque o cálculo da cobertura dessas especializações é baseado em evidências espúrias e incorretas. Há ainda milhares de novas RNs; geralmente com vários predicados, cobertura próxima a zero, e sem significado semântico aparente. Numericamente, percebemos que a distribuição das coberturas é composta por partes estacionárias. As áreas sombreadas na Figura 2 mostram onde os valores de cobertura mudam abruptamente (com base na mediana). O número de mudanças abruptas é menor e mais suave para $\Sigma_{\mathsf{Hospital}_{limpa}}$. Em $\Sigma_{\mathsf{Hospital}_{suja}}$, no entanto, há um maior número de mudanças abruptas, e consequentemente, um maior número de partes estacionárias. Dessa forma, a classifica-

ção de $\Sigma_{\mathsf{Hospital}_{limpa}}$ é numericamente melhor já que a suavidade da cobertura mostra RNs com coberturas igualmente distribuídas e com uma evidente amplitude de separação.

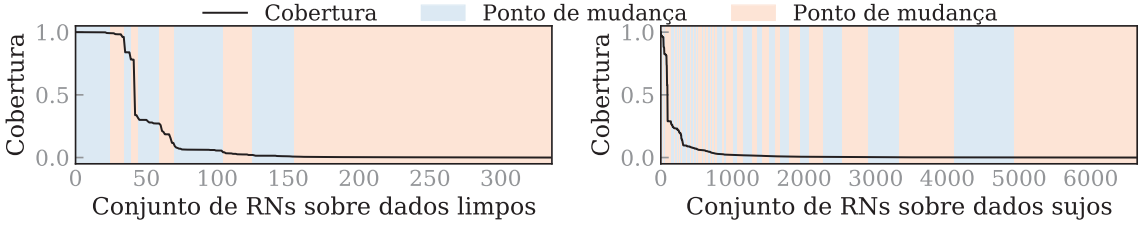


Figura 2. Cobertura das RNs em $\Sigma_{\mathsf{Hospital}_{limpa}}$ (à esquerda) e $\Sigma_{\mathsf{Hospital}_{suja}}$ (à direita).

Queremos descobrir um conjunto de RNs Σ_{conf} que se aproxime do subconjunto de $\Sigma_{\mathsf{r}_{limpa}}$ com alta cobertura. Primeiro, calculamos o conjunto de todas as evidências do conjunto de dados (sujo). Erros frequentemente constituem uma pequena porcentagem de dados [Abedjan et al. 2015]. Nesse caso, mesmo que evidências corretas sejam dissolvidas por erros, a tendência central dessas evidências é mantida. Como a variabilidade na multiplicidade das evidências é significativa, usamos a mediana como medida de tendência central. Seja md a mediana das multiplicidades dos elementos de Λ_{r}. Calculamos um conjunto de evidências Λ_{md} tal que $\Lambda_{md} = \{\lambda \mid \lambda \in \Lambda_{\mathsf{r}} \wedge \mathsf{multi}(\lambda) > md\}$. Se considerarmos apenas as evidências Λ_{md} para descoberta de RNs, desprezamos muitas evidências que podem ser consistentes com relação a predicados não envolvidos em erros (e.g., tuplas que contêm erros em um atributo A_i mas não em um atributo A_j). Ao invés disso, calculamos uma expectativa do erro de r e atribuímos ao parâmetro ε com a seguinte fórmula $\varepsilon = 1 - \frac{\|\Lambda_{md}\|}{|\mathsf{r}|\cdot(|\mathsf{r}|-1)}$. Usamos o método de descoberta de coberturas mínimas com a entrada Λ_{r} e limite de erro ε. Assim garantimos que cada RN descoberta é violada por no máximo $\varepsilon \cdot |\mathsf{r}| \cdot (|\mathsf{r}| - 1)$ pares de tuplas. Isso reflete uma esperança de erro calculada a partir da tendência central das evidências. A negação das coberturas mínimas são as RNs ε-parciais de interesse. Após ordenar o resultado de forma decrescente pela medida de cobertura, aplicamos o método de detecção de mudança abrupta descrito em [Killick et al. 2012] e inserimos em Σ_{conf} toda RN que aparecer antes da primeira mudança abrupta.

5. Avaliação Preliminar

Nós adaptamos o método descrito na Seção 4 ao algoritmo BFASTDC para medir a precisão e revocação em que as RNs de Σ_{conf} encontram inconsistências reais nos conjuntos de dados. Nosso protótipo é um cliente escrito na linguagem Java (versão 1.8) conectado a uma instância de banco de dados PostgreSQL (versão 9.5.19). Utilizamos dois conjunto de dados reais: *Hospital* e *Flights*. Ambos têm sido extensivamente utilizados na literatura de limpeza de dados, maiores detalhes em [Rekatsinas et al. 2017]. Os conjuntos de dados possuem uma versão correta (r_{limpa}); e uma versão incorreta (r_{suja}), onde todas as inconsistências são conhecidas. *Hospital* tem 1000 registros, 20 atributos e taxa de erro de 0.03; *Flights* tem 2376 registros, 6 atributos e taxa de erro de 0.30. Como *baselines*, utilizamos a saída do algoritmo BFASTDC original configurado com valores de erro utilizados na literatura: $\varepsilon = 0.01$, e $\varepsilon = 0.05$.

A Tabela 1 mostra os índices de precisão e revocação obtidos pelos diferentes métodos. O método proposto supera consistentemente os demais, e atinge bons índices de precisão e revocação mesmo quando a taxa de erros do conjunto de dados é alta (i.e.,

Flights). Em particular, o método encontra todas as inconsistências nos dois cenários. Nos *baselines*, muitos pares de tuplas que não caracterizam inconsistências são identificados, o que reduz drasticamente a precisão. Além disso, a revocação nos *baselines* é sensível ao parâmetro ε o que aponta que a medida deve ser estimada com cautela. Nosso método leva em consideração as características dos dados e atinge bons resultados sem a necessidade de intervenção humana. Sobre os recursos computacionais do método proposto comparado aos *baselines*, não houve aumento significativo no tempo de execução ou memória requerida. Isso é esperado uma vez que o método proposto apenas inclui cálculos simples, e a implementação da detecção de mudanças abruptas tem custo linear.

Tabela 1. Comparação em termos de detecção de pares de tuplas inconsistentes.

Método	Hospital		Flights	
	Prec.	Rev.	Prec.	Rev.
BFASTDC adaptado com o método proposto	**0.93**	**1.0**	**0.70**	**1.0**
BFASTDC com $\varepsilon = 0.01$	0.08	1.0	0.06	0.52
BFASTDC com $\varepsilon = 0.05$	0.03	1.0	0.06	0.99

6. Discussão final e direções futuras

Os resultados da Secção 5 são promissores e mostram que é possível identificar RNs confiáveis mesmo à partir de dados com erros. No entanto, o método precisa ser avaliado em mais cenários: mais registros, mais atributos, e variados níveis de sujeira. Obter dados 100% corretos é um desafio. Por isso, pretendemos adicionar dados sintéticos aos experimentos para avaliar os limites do nosso método. Também pretendemos investigar o impacto que outros tipos de erros (e.g., duplicatas) causam na descoberta de RNs confiáveis, e como seus efeitos podem ser contornados. Um problema ortogonal a nossa pesquisa é a detecção de pares de tuplas inconsistentes. A detecção tem caráter quadrático no número de registros e requer estruturas e algoritmos adequados para uma execução eficiente.

Referências

Abedjan, Z., Golab, L., and Naumann, F. (2015). Profiling relational data: A survey. *The VLDB Journal*, 24(4):557–581.

Chu, X., Ilyas, I. F., and Papotti, P. (2013). Discovering denial constraints. *Proc. VLDB Endow.*, 6(13):1498–1509.

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598.

Pena, E. H. M. and de Almeida, E. C. (2018). BFastDC: A bitwise algorithm for mining denial constraints. In *DEXA 2018*, pages 53–68.

Rekatsinas, T., Chu, X., Ilyas, I. F., and Ré, C. (2017). Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201.

Visualização dos dados abertos da Polícia Rodoviária Federal sobre acidentes nas rodovias brasileiras

Victor G. P. de Mattos, Pedro H. V. Vasconcelos, Yussef Parcianello, Nádia P. Kozievitch, Rita Berardi

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Avenida Sete de Setembro, 3165
Departamento Acadêmico de Informática – DAINF – Curitiba – Brasil

{victormattos,pedrovasconcelos}@alunos.utfpr.edu.br,

yussef.parcianello@ifsc.edu.br, {nadiap,ritaberardi}@utfpr.edu.br

Abstract. *Traffic accidents are one of the main causes of death in Brazil and, in the last 10 years, more than 1.6 million accidents along with 83 thousand deaths were registered. This paper proposes an open-source prototype for the visualization of open data provided by Polícia Rodoviária Federal (PRF), mapping historically the most dangerous sections of the highways, the most frequent causes of accidents, and helps to identify patterns. The purpose of this paper is to promote the awareness of users, aiming at preventing and reducing the number of accidents on Brazilian highways, which today, in addition to human loss, generate an average loss of R$10 billion annually.*

Resumo. *Os acidentes de trânsito são uma das principais causas de óbito no Brasil e, somente nos últimos 10 anos, foram registrados mais de 1.6 milhões de acidentes e mais de 83 mil mortes. Este artigo propõe um protótipo de código aberto para a visualização dos dados abertos da Polícia Rodoviária Federal (PRF), mapeando os trechos historicamente mais perigosos nas rodovias, auxiliando na identificação de padrões. O intuito deste artigo é promover a conscientização dos usuários, visando a prevenção e diminuição no número de acidentes nas rodovias brasileiras que, hoje, além da perda humana, geram um prejuízo médio de R$ 10 bilhões anualmente.*

1. Introdução

Cidade Inteligente é o termo dado à cidade que utiliza Tecnologias da Informação e Comunicação (TIC) de forma criativa, inclusiva e eficiente na gestão pública, auxiliando nas tomadas de decisão, bem como na criação de inovações que façam uso destas informações para gerar soluções que facilitem a vida dos cidadãos [Weiss et al. 2017].

Uma das peças-chave da gestão pública para a disseminação da informação e a promoção da transparência é a abertura de dados públicos, os chamados dados abertos. O aumento de dados públicos cria um cenário favorável à participação social, permitindo o cidadão ajudar a produzir informações que auxiliem na discussão e mapeamento de problemas, bem como o poder público na identificação dos mesmos [Lemos 2013].

Com o intuito de compreender melhor as características dos acidentes registrados pela PRF, o uso de tecnologias geoespaciais como os Sistemas de Informação Geográfica

(SIG) proporciona, além de facilidade na análise, a criação de modelos espaciais que auxiliem no processo de tomada de decisões [Tao 2013].

A pesquisa de [Vila et al. 2016] propôs uma interface de visualização de dados geográficos com técnicas de clusterização utilizando a base de dados de mobilidade urbana utilizando técnicas de clusterização, permitindo o agrupamento de um grande conjunto de dados georreferenciados, tornando a visualização espacial mais intuitiva, como mostra a Figura 1 (esquerda).

Já o Instituto de Pesquisa e Planejamento Urbano de Curitiba (IPPUC)[1] disponibiliza uma interface[2] de visualização dos dados geográficos de acidentes de trânsito com vítimas fatais no município de Curitiba, como mostra a Figura 1 (direita). Além da visualização, o IPPUC também disponibiliza, em forma de gráficos, os dados mais relevantes destes acidentes, de modo que o usuário possa filtrar e definir quais dados devem aparecer nos gráficos.

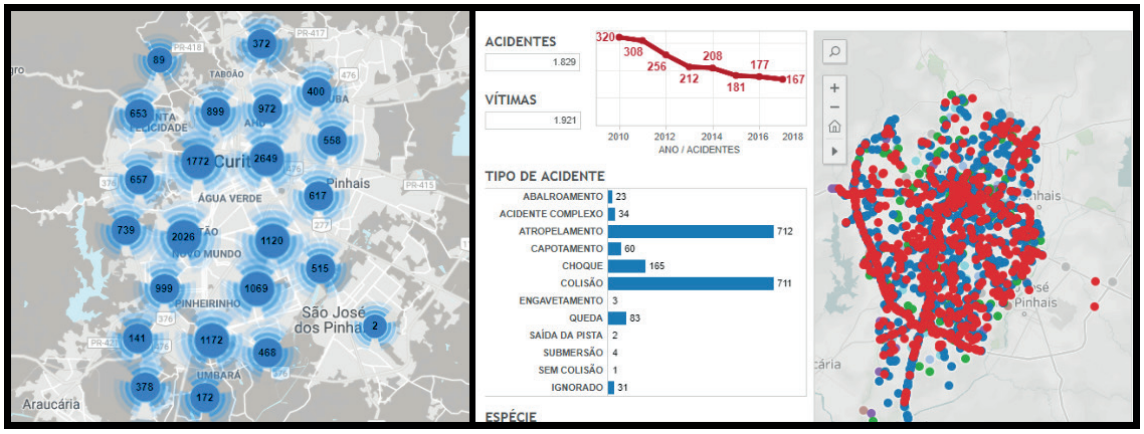


Figura 1. Agrupamento dos pontos de ônibus de Curitiba (esquerda) [Vila et al. 2016] e interface de visualização de acidentes de trânsito com vítimas fatais em Curitiba (direita) - IPPUC.

Em particular, dados de acidentes possuem não só componentes geográficos e temporais, como também dinâmicas incluindo trecho de BR, classificação diferenciada de acidentes, veículos e passageiros envolvidos em agregações diferenciadas (país, estado, cidade). Neste sentido, este trabalho em andamento utiliza como base os dois trabalhos acima para uma ferramenta visual de dados de acidentes, com o objetivo de suprir o problema de sobreposição de acidentes em áreas e identificação de padrões. Para isso, foram usados dados de uma década da PRF[3].

O restante do trabalho está organizado da seguinte maneira: a seção 2 apresenta trabalhos relacionados. A seção 3 apresenta o protótipo da aplicação, e a seção 4 lista as conclusões e trabalhos futuros.

[1]www.ippuc.org.br - Acessado em: 09 de outubro de 2018.

[2]www.ippuc.org.br/mapasinterativos/acidentesDeTransito/dashboard.html - Acessado em: 17 de abril de 2019.

[3]https://www.prf.gov.br/portal/dados-abertos - Acessado em: 26 de junho de 2019.

2. Trabalhos Relacionados

À medida que o mundo se torna cada vez mais instrumentado, interconectado e inteligente, as cidades têm a chance fazer uso de novas soluções e práticas de gerenciamento "inteligentes" [Dirks et al. 2009]. Com as cidades ficando mais inteligentes, uma grande quantidade de dados pode ser colocada à disposição da população e empresas, impulsionando o desenvolvimento de aplicações comerciais, pesquisas e apoio à tomada de decisões complexas [Lopes et al. 2016], contribuindo para o desenvolvimento de novos serviços de utilidade para cidadãos e gestores municipais [Cunha et al. 2016].

Neste contexto, os SIGs possuem como funcionalidades principais a captura, armazenamento, consulta, análise e exibição de dados geoespaciais [Chang 2015]. Longley [Longley et al. 2005] definem os SIG como uma extensão de sistemas de informação, que armazenam não só eventos e atividades, mas também onde estes acontecem. Para um acidente de trânsito em uma rodovia brasileira, como mostra [Kageyama et al. 2019], por exemplo, esses dados geoespaciais descrevem, além da localização, informações relacionadas como: em que rodovia ocorreu, em que quilômetro ocorreu, qual era o sentido da pista, o horário, a data, etc.

Com o aumento da quantidade de dados e informações disponíveis, torna-se cada vez mais necessário o uso de ferramentas e técnicas que ajudem a fazer uso efetivo desse excesso de informação, como por exemplo, a visualização de dados. Dentro do contexto deste projeto, das aplicações de visualização de dados de acidentes de trânsito pesquisadas, destacam-se: o *New York City Open Data*[4] (Figura 2 - canto superior esquerdo), o *Mapping 10 Years of Fatal Traffic Accidents*[5] (Figura 2 - canto superior direito), o *The Boston Crash Model*[6] (Figura 2 - canto inferior esquerdo) e o *Seattle Collisions*[7] (Figura 2 - canto inferior direito).

Todas as aplicações citadas utilizam o *OpenStreetMap* e, com exceção do *Seattle Collisions*, que faz uso de bibliotecas do *Javascript*, são utilizadas ferramentas pagas especializadas em visualização de mapas. Além disso, a maioria delas não faz uso de ferramentas de clusterização ou visualização (o que gera sobreposição dos dados e dificulta o entendimento dos mesmos), além de não possuirem código aberto. Este trabalho propõe o uso de ferramentas de clusterização para a visualização no intuito de facilitar a identificação de padrões.

3. Protótipo

Aproximadamente 10 anos de dados de acidentes de todo o Brasil foram inseridos em um servidor *PostGIS* [Obe and Hsu 2011], com aproximadamente 1 milhão de tuplas, entre os anos de 2007 e 2018. Detalhes dos dados (como o menor número de acidentes ao longo dos últimos 5 anos, a concentração de mortes em fins de semana, e a preferência dos acidentes pelo horário das 18 horas e pelo mês de dezembro, entre outros) podem ser verificados em [Kageyama et al. 2019].

O protótipo de visualização foi desenvolvido para a *web* e utiliza a arquitetura apresentada na Figura 3: *PostgreSQL (11.0)*, *PostGIS* (6.5.2) (camada de dados), *Node.js*

[4]www.opendata.cityofnewyork.us - Acessado em: 18 de setembro de 2018.
[5]www.metrocosm.com/10-years-of-traffic-accidents-mapped.html - Acessado em: 10 de abril de 2018.
[6]www.apps.boston.gov/vision-zero - Acessado em: 15 de outubro de 2018.
[7]www.seattlecollisions.timganter.io/collisions - Acessado em: 15 de outubro de 2018.

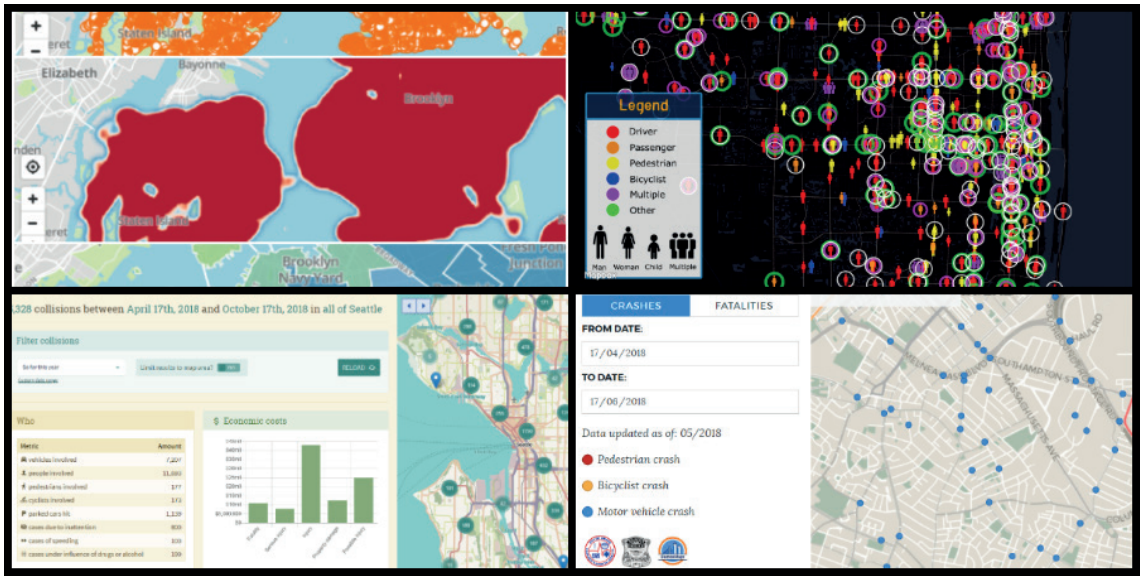


Figura 2. Aplicações de visualização de acidentes de trânsito.

(8.12.0) e *Express* (4.17.0) (camada de processamento) e *HTML*, *CSS*, *JavaScript*, *D3* (3.5.17) e *Leaflet* (0.7.2) (camada de apresentação).

Figura 3. Arquitetura da aplicação.

A interface de resultado foi inspirada em interfaces já utilizadas pelo governo local (IPPUC), e está dividida entre o mapa que contém os acidentes (mostrado na Figura 4) e em outra metade com as estatísticas gerais dos acidentes (mostrado na Figura 5). Os gráficos podem ser reproduzidos através de filtros como: dia da semana, ano, horário, região e tipo de acidente.

Na Figura 4, os acidentes estão agrupados em clusters que representam a quantidade de acidentes em determinada rodovia. Além disso, os clusters foram classificados, inicialmente, em acidentes com vítimas fatais, vítimas feridas e sem vítimas, cada um com uma cor específica descrita no índice ao lado do mapa (parte superior da Figura 4). A clusterização no mapa é ajustada dinamicamente de acordo com a posição e *zoom*,

como na parte inferior da Figura 4, que mostra o *zoom* na cidade de Curitiba e sua região metropolitana, evidenciando as rodovias em que ocorreram os acidentes e a reorganização dos clusters (opções não disponíveis em ferramentas locais do governo como IPPUC, por exemplo).

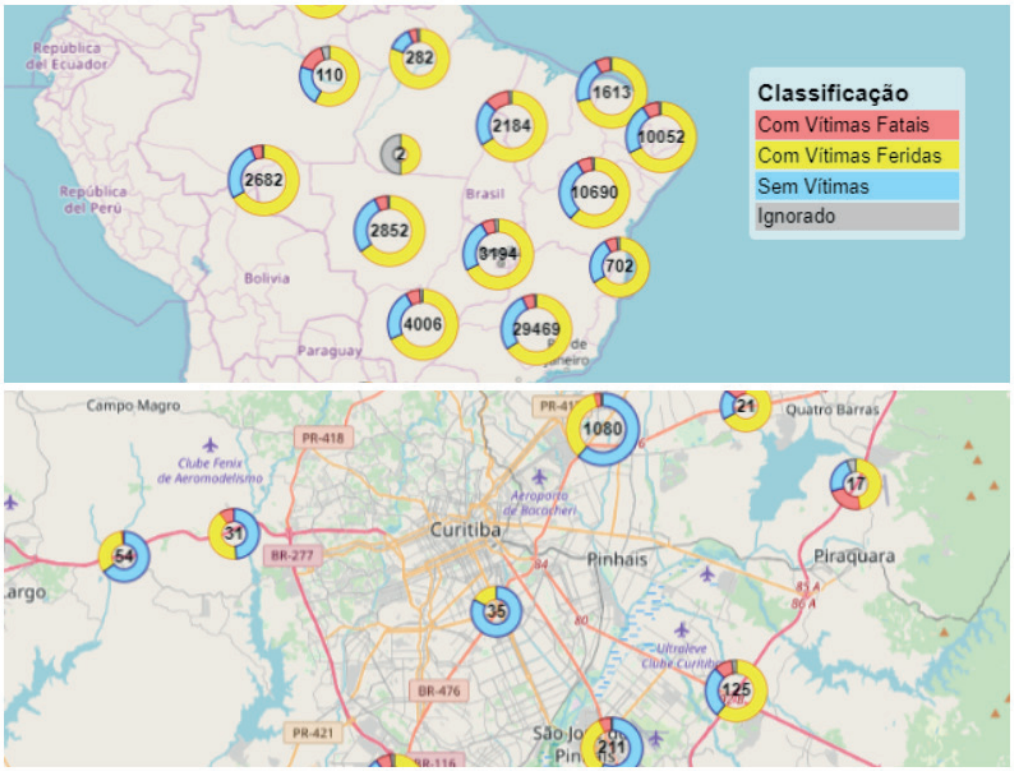


Figura 4. Protótipo da aplicação.

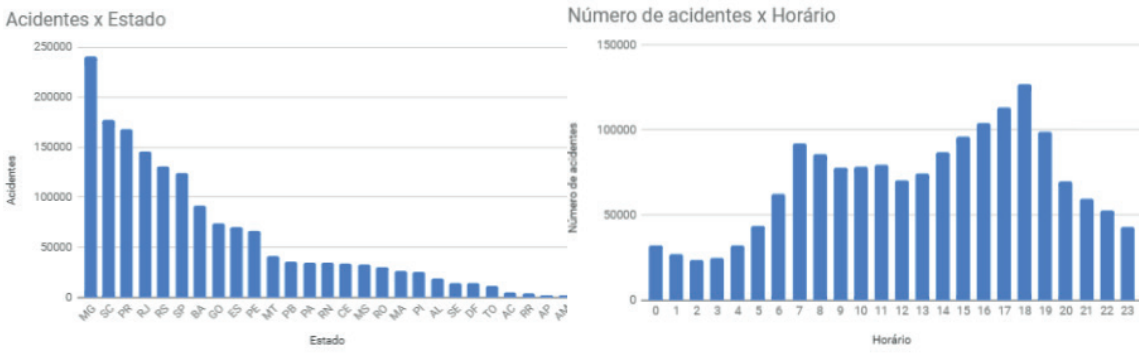


Figura 5. Análise de acidentes por Estado (esquerda) e análise de número de acidentes por horário (direita).

Além do protótipo, um questionário para validar requisitos[8] foi desenvolvido para ser submetido a PRF e ao público em geral. O questionário possui perguntas básicas como: ”Você faz uso de alguma rodovia federal?”, ”Qual a frequência com que você utiliza as rodovias federais?”, ”Você, algum conhecido ou familiar, já se envolveu em

[8]https://forms.gle/7fhcWQF57aDrq24u9 - Acessado em: 09 de julho de 2019.

algum acidente de trânsito nas rodovias brasileiras?", etc. Após a finalização do projeto, outro questionário será realizado para validar a ferramenta.

4. Conclusão

Este artigo apresentou um trabalho em andamento de uma ferramenta visual de dados de acidentes, com o objetivo de suprir o problema de sobreposição de dados em áreas e identificação de padrões. Através da utilização do protótipo, é possível tomar ciência da quantidade de acidentes de trânsito nas rodovias brasileiras, assim como é possível também identificar onde tais acidentes ocorrem, o que permite identificar as regiões mais violentas em termos de acidentes com vítimas fatais.

Como trabalho futuro, a este trabalho poderia ser adicionado os dados de redutores de velocidade, veículos e pessoas envolvidos em acidentes, além de outras estatísticas por período, como meses com maior número de acidentes, localização onde os mesmos ocorrem, as principais causas, e períodos com festas.

Referências

Chang, K.-T. (2015). *Introduction to geographic information systems*. McGraw-Hill Higher Education Boston, Boston.

Cunha, M. A., Przeybilovicz, E., Macaya, J. F. M., and Santos, F. B. P. d. (2016). Smart cities: Transformação digital de cidades.

Dirks, S., Keeling, M., and Dencik, J. (2009). How smart is your city?: Helping cities measure progress. *IBM Institute for Business Value, IBM Global Business Services, New York*.

Kageyama, M., Kozievitch, N. P., and Berardi, R. (2019). Acidentes nas rodovias brasileiras nos ultimos 10 anos: uma análise com dados abertos. In *XV Escola Regional de Banco de Dados*, pages 79–87.

Lemos, A. (2013). Cidades inteligentes. *GV-executivo*, 12(2):46–49.

Longley, P. A., Goodchild, M. F., Maguire, D. J., and Rhind, D. W. (2005). *Geographic information systems and science*. John Wiley & Sons, England.

Lopes, G., Vidal, V. M. P., and Oliveira, M. (2016). Construção de linked data mashup para integração de dados da saúde pública. In *SBBD 2016*, pages 145–150.

Obe, R. and Hsu, L. (2011). Postgis in action. *GEOInformatics*, 14(8):30.

Tao, W. (2013). Interdisciplinary urban gis for smart cities: advancements and opportunities. *Geo-spatial Information Science*, 16(1):25–34.

Vila, J., Kozievitch, N. P., Gadda, T. M. C., Fonseca, K., Rosa, M. O., Gomes, L. C., and Akbar, M. (2016). Urban mobility challenges - an exploratory analysis of public transportation data in curitiba. *Revista de Informática Aplicada*, 12(1).

Weiss, M. C., Bernardes, R. C., and Consoni, F. L. (2017). Cidades inteligentes: casos e perspectivas para as cidades brasileiras. *Revista Tecnológica da Fatec Americana*, 5(1):01–13.

Conformity Analysis of GTFS Routes and Bus Trajectories

Andreza Raquel M. Queiroz[1], **Veruska B. Santos**[1],
Dimas C. Nascimento[1,2], **Carlos Eduardo S. Pires**[1]

[1]Departamento de Sistemas e Computação
Universidade Federal de Campina Grande (UFCG)

[2]Unidade Acadêmica de Garanhuns
Universidade Federal Rural de Pernambuco (UFRPE)

{andrezaraquel,veruska}@copin.ufcg.edu.br

dimascnf@gmail.com, cesp@dsc.ufcg.edu.br

Abstract. *General Transit Feed Specification (GTFS) is a standard data format generated by transportation agencies of most of the cities worldwide to provide scheduled data of their services. Despite being the standard in the public transportation field, applications that consume GTFS data may face two problems: outdated versions, since some transportation agencies do not provide GTFS in the same frequency that transit changes; and discrepancy with positioning data sent by the buses. This paper provides a conformity analysis of GTFS routes and bus positioning data from multiple cities. We have found inconsistencies related to GPS route labels and GTFS routes. We also classify the conformity of bus trajectories and enumerate the main inconsistencies found in data analysis.*

1. Introduction

Several bus monitoring applications[1] use Global Positioning System (GPS) and General Transit Feed Specification (GTFS[2]) data as a source of supposedly accurate and official information about transit services. GTFS is a standard data composed of predefined bus routes represented by sequences of geographical points as well as other public transportation data. In turn, GPS bus data basically consists of records containing timestamp, vehicle number, bus route label, latitude and longitude. A set of GPS records with the same vehicle number ordered by timestamp is called a bus trajectory.

Bus monitoring applications usually face two crucial problems. The first one consists of outdated versions of GTFS data while the second problem is related to inconsistencies in the GPS data sent by buses. To address the first problem, the authors of [Wessel et al. 2017] developed a method to improve the accuracy of GTFS data using real-time GPS data sent by vehicles. Briefly, the method updates the GTFS data each time a significant change in a bus trajectory is detected. However, the authors do not deal with GPS inconsistencies which can introduce errors in the updated GTFS.

Ideally, the GPS route label should indicates the GTFS route that the bus is following. However, as mentioned before, there are inconsistencies in GPS data sent by

[1]Examples of bus monitoring applications: https://www.google.com/maps/, https://moovitapp.com/, https://www.ciomcg.com.br/

[2]https://developers.google.com/transit/gtfs/reference/

buses. These inconsistencies usually refer to a) missing GPS route label, probably due to device failures [Raymond and Imamichi 2016] and b) route label inconsistency, e.g., the route label indicated in the GPS data is different from the GTFS route that the bus is actually following. The problem of missing GPS route label was previously addressed in [Raymond and Imamichi 2016]. The authors have shown that the cosine similarity is an effective method to determine the routes followed by buses. In their experiments, using data from Rio de Janeiro, they compared the results generated using the cosine similarity with the route label sent by the buses. However, the authors neither considered the problem of outdated GTFS nor the second inconsistency in GPS route label, although their results were clearly affected by them.

In this work, we analyze the conformity of GTFS routes and bus trajectories. The analysis addresses both the outdated GTFS problem and GPS inconsistencies. For doing so, we use the cosine similarity method proposed in [Raymond and Imamichi 2016] to perform a more detailed study regarding the results generated by the method. Our main contributions are: a) we classify the conformity of bus trajectories with GTFS routes; and b) we enumerate the most frequent data inconsistencies found in our analysis. In addition, we demonstrate that, in some cases, the route determined by the cosine method is more likely to be the route that the bus is following than the route labeled in the GPS data.

2. Methodology

In order to analyze the conformity of GTFS routes and bus positioning data, we employ datasets of three Brazilian cities: City A (name omitted due to privacy requirements related to the usage of its bus GPS data), Curitiba and Rio de Janeiro. These datasets are summarized in Table 1. We downloaded Curitiba GTFS and GPS historical data from URBS (Urbanização de Curitiba S/A) web page. URBS is the agency that manages public transportation in Curitiba. Both datasets from Curitiba are publicly available[3]. City A data was made available by its public transportation agency. Rio de Janeiro GTFS and GPS data were provided by the authors of [Raymond and Imamichi 2016].

Table 1. Summary of datasets.

	GPS interval	# Bus trajectories	# GTFS routes
City A	Dec 03, 2018 to Dec 07, 2018	247	250
Curitiba	Aug 27, 2017 to Aug 31, 2017	697	235
Rio de Janeiro	Feb 15, 2016 to Feb 17, 2016.	5,376	375

To process the datasets and perform the conformity analysis, we followed the same steps of the approach proposed in [Raymond and Imamichi 2016] as follows.

2.1. Map matching of bus trajectories and GTFS routes

The first step is to apply map matching (MM) in bus trajectories and GTFS routes. MM is a process that integrates noise positioning data with the road network to obtain enhanced positions [Quddus et al. 2007]. In this work, we applied the GraphHopper MM algorithm that basically follows the approach described in [Newson and Krumm 2009]. GraphHopper implements the Hidden Markov Model (HMM) which is based on states, such that

[3]http://dadosabertos.c3sl.ufpr.br/curitibaurbs/, http://transporteservico.urbs.curitiba.pr.gov.br/index.php

the probability of the next state depends only on the current state. In the MM problem, the states are the road segments (i.e., a portion of a road between two intersections) of the road network. HMM map matching methods are known to be robust when dealing with GPS data which may contain measurement errors, as well as long and irregular intervals between measurements [Kubicka et al. 2018]. The code is publicly available[4]. The latest version of Open Street Map (OSM[5]) was used as the road network. Regarding Rio de Janeiro data, we used the MM output made available by [Raymond and Imamichi 2016].

2.2. Bag-of-roads transformation

The second step is to turn the MM outputs (i.e., the enhanced positions) generated in the previous step into vectors of roads. This process generates bag-of-roads (BoR) vectors to represent bus trajectories and GTFS routes. The BoR vectors maintain the same dimension in order to allow comparisons between them using classical similarity metrics, such as the cosine similarity. BoR vectors work analogously to the bag-of-words vectors for document classification. However, instead of counting words, each cell of a BoR vector represents a road segment and stores the frequency of a bus or route intersecting that segment.

We performed a modification in BoR vectors of bus trajectories in the sense that each cell simply indicates whether (or not) the bus intersected that segment. This modification brought better similarity results than the original BoR representation. This improvement occurs because bus trajectories are composed of various trips, all of them intersecting the same road segments. In turn, a trip is a sub-sequence of the trajectory that covers the route only once. In the original version proposed in [Raymond and Imamichi 2016], the authors assigned low similarity values to trajectories that are very similar to the route only because these trajectories are composed of several trips.

2.3. Route Identification

The last step is to identify the GTFS route of each bus. To this end, we compute the cosine similarity between BoR vectors representing bus trajectories and BoR vectors representing GTFS routes. The result of this step is a list of bus trajectories, each of them associated with its most similar route. Bus trajectories composed by GPS data whose route label does not correspond to an existing route in the GTFS were discarded. We consider more than one trajectory for the same bus if it follows more than one route and the label in its GPS data indicates this.

3. Conformity Analysis

We grouped the bus trajectories according to the similarity value between them and the corresponding GTFS route, determined using the cosine method. As a result, six groups were generated as depicted in Figure 1. Each group (represented by a box) indicates the conformity level of the bus trajectory. The horizontal axis refers to the similarity degree ranging from 0 to 1. The boxes above the horizontal axis include the cases in which the bus route label is in conformity with the found route. On the other hand, the boxes below the horizontal axis are related to cases where the bus route label is not in conformity with the found route.

[4]https://github.com/graphhopper/map-matching

[5]http://download.geofabrik.de/

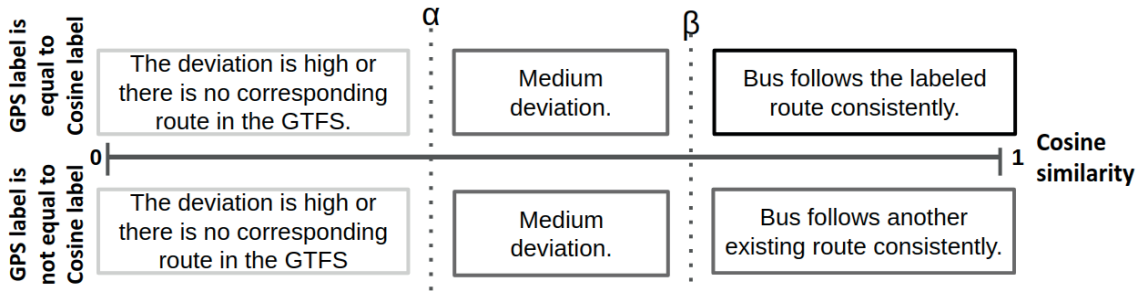


Figure 1. Conformity levels of bus trajectories based on their similarity with GTFS routes.

One level of conformity is assigned to each color of the boxes: a) black represents an ideal conformity level, i.e., the similarity of the trajectory with the GPS labeled route is high (similarity $\geq \beta$) and the GPS route label is equal to the found route; b) dark grey indicates that there is a certain discrepancy between bus trajectories and GTFS routes, i.e., the cosine similarity is not high ($\alpha <$ similarity $< \beta$) and/or the GPS route label is different from the found route; and c) light grey represents the worst cases of discrepancy with GTFS routes, i.e., the similarity between the bus trajectory and GTFS route is low (similarity $\leq \alpha$). α and β are configurable similarity thresholds used to classify the bus trajectories in levels. The threshold values should be chosen by a domain specialist. For the purpose of our analysis, we considered $\alpha = 0.4$ and $\beta = 0.7$, since these values partition the results consistently, generating cohesive groups.

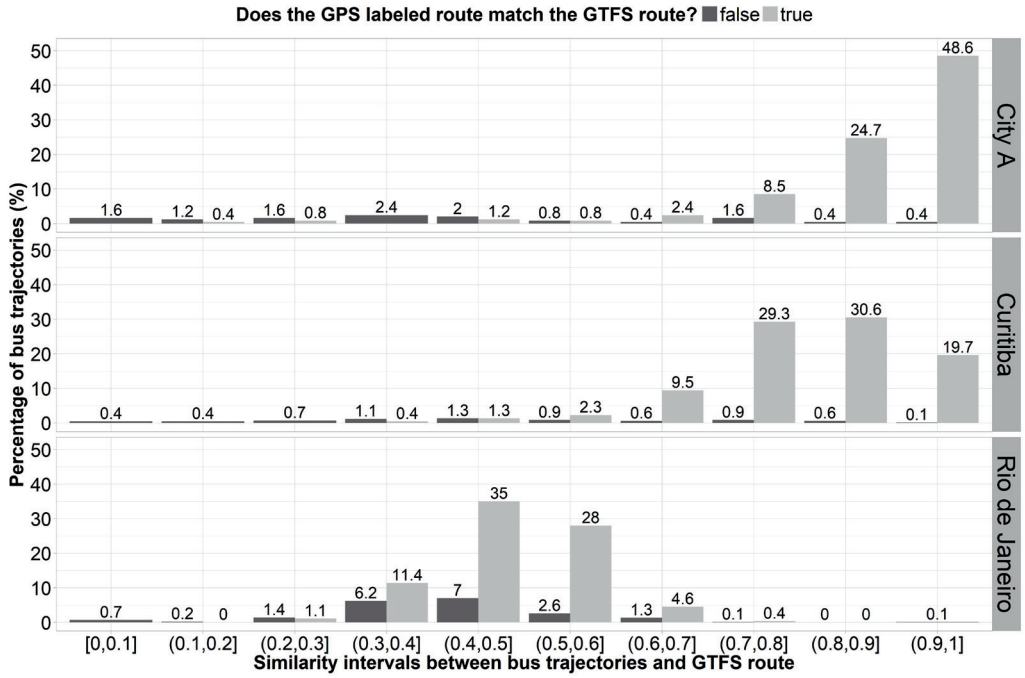


Figure 2. Percentage of bus trajectories that match (or not) the labeled GPS route, per similarity interval.

Figure 2 presents the percentage of bus trajectories that match (or not) the labeled GPS route, per similarity interval for each city. The horizontal axis represents the intervals of similarity whilst the vertical axis represents the percentage of bus trajectories.

The dark grey bars refer to the bus trajectories whose GPS route label is different from the identified route. In turn, the light grey bars refer to the bus trajectories whose GPS route label matches the identified route. Most buses from City A follow the labeled route consistently. However, 18.2% of its buses present a level of discrepancy with the GTFS. The same observation is verified for bus trajectories from Curitiba. On the other hand, Rio de Janeiro presents a different scenario. The bus trajectories are concentrated exactly in the intermediary region of similarity level, i.e., most of the trajectories show a medium deviation from the GTFS routes.

The low similarity levels associated with the bus trajectories can be explained by a variety of problems: GPS monitoring failure[6], outdated GTFS and/or bus route deviations. In the following, we enumerate the data inconsistencies generated by these problems and present examples of inconsistencies found in the analyzed datasets:

- **Inconsistency 1: the GPS route label is different from the route that the bus is following.** Figure 3 shows the trajectory of bus JC013. As it can be seen, the bus is actually following route 778 even though the GPS is labeled with route 776;
- **Inconsistency 2: the bus deviates partially from the route that it is actually following.** In Figure 3, bus LA002 never follows the highlighted portion of the route labeled on GPS;
- **Inconsistency 3: the bus follows more than one route, even though the GPS label indicates only one route.** In Figure 4, bus A29023 seems to be following both route 473 and route 441, but the GPS label indicates only route 473;
- **Inconsistency 4: the bus does not follow any of the GTFS routes.** In Figure 4, the trajectory of bus A48071 does not match any of the routes defined in the GTFS.

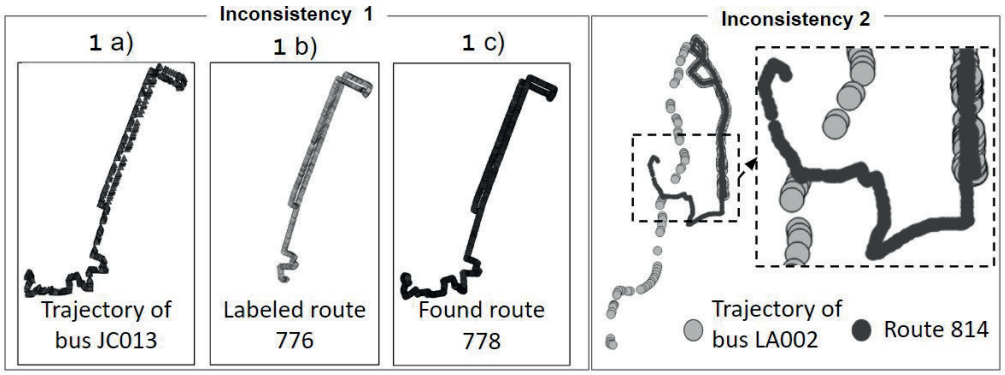


Figure 3. Examples of inconsistencies found in the Curitiba dataset. 1a: trajectory of bus JC013; 1b: route labeled in the GPS data; 1c: route found using the cosine method. The route determined by the cosine method is visually more similar to the bus trajectory than the GPS labeled route. Inconsistency 2: the bus never follows the highlighted part of the route.

4. Conclusion

GTFS is a standard that facilitates the exchange of public transportation data and should be considered the gold standard in the field. However, we demonstrated that buses oper-

[6]https://g1.globo.com/rj/rio-de-janeiro/noticia/fora-do-ponto-mais-da-metade-dos-onibus-do-rio-tem-falha-no-monitoramento-por-gps.ghtml

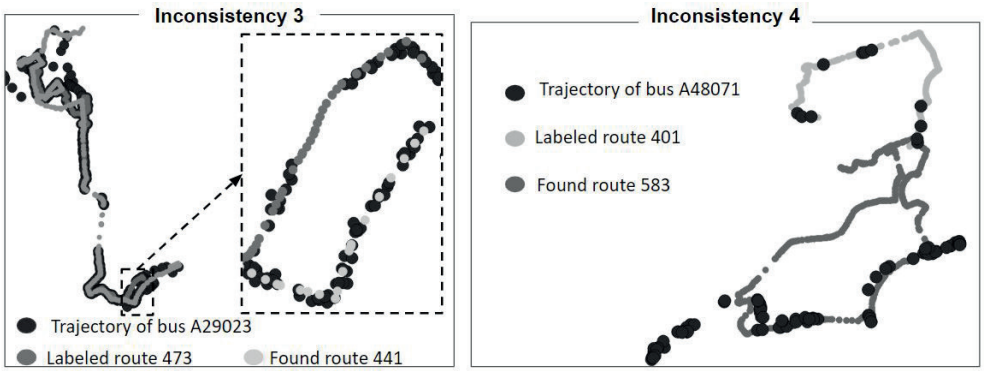


Figure 4. Examples of inconsistencies found in the Rio de Janeiro dataset. Inconsistency 3: the highlighted part indicates that bus A29023 is following both routes 473 and 441. Inconsistency 4: bus A48071 is neither following the found route (583) nor the labeled route (401).

ating in diverse cities do not always follow the programmed route, showing a certain inconsistency with the GTFS. Some inconsistencies are more serious and should be treated immediately, such as the incorrect GPS route label. Other inconsistencies, such as the bus following more than one route or deviating from its predefined route could be avoided with a more effective strategic plan in the GTFS creation and real-time supervision of bus GPS data. The categorization of conformity presented in this work can be extended by a specialist in public transportation, according to its interests. Future work will focus on generate GTFS routes based on bus trajectory data. Thus, we can ensure that GTFS data is updated and also it is in conformity with GPS data.

Acknowledgment

This research was partially funded by INES 2.0, FACEPE grant APQ-0399-1.03/17, CAPES grant 88887.136410/2017-00 and CNPq grant 465614/2014-0.

References

Kubicka, M., Cela, A., Mounier, H., and Niculescu, S.-I. (2018). Comparative study and application-oriented classification of vehicular map-matching methods. *IEEE Intelligent Transportation Systems Magazine*, 10(2):150–166.

Newson, P. and Krumm, J. (2009). Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343. ACM.

Quddus, M. A., Ochieng, W. Y., and Noland, R. B. (2007). Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation research part c: Emerging technologies*, 15(5):312–328.

Raymond, R. and Imamichi, T. (2016). Bus trajectory identification by map-matching. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1618–1623. IEEE.

Wessel, N., Allen, J., and Farber, S. (2017). Constructing a routable retrospective transit timetable from a real-time vehicle location feed and gtfs. *Journal of Transport Geography*, 62:92–97.

Comunicação em bloco na exploração de grafos em bases RDF distribuídas

Eduardo Villas Boas Santos[1], **Carmem S. Hara**[2], **Raqueline R. M. Penteado**[1]

[1] Departamento de Informática – Universidade Estadual de Maringá
Avenida Colombo 5.790 – 87.020-900 – Maringá, PR – Brasil

[2]Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19.081 – 81.531-990 – Curitiba, PR – Brasil
ra62973@uem.br, carmem@inf.ufpr.br, raque@din.uem.br

Abstract. *Distributed SPARQL query processing involves the exchange of intermediate results among RDF storage servers. This paper analyzes the impact of grouping these results into blocks in order to reduce the number of transmissions. The proposed communication strategy has been implemented on a SPARQL query processor based on a graph exploration algorithm. Experimental results showed that block-based communication can improve the performance of distributed query processing. Future works include extension of the SPARQL processor by considering the cost of block communication in query planning and optimization.*

Resumo. *Sistemas RDF distribuídos adotam estratégias de comunicação que trocam resultados intermediários entre servidores durante a execução distribuída de consultas. Este artigo analisa o impacto do agrupamento de resultados intermediários em blocos, com a finalidade de reduzir a quantidade de transmissões. A comunicação em blocos foi implementada em um processador de consultas SPARQL baseado em um algoritmo de exploração de grafos. Resultados experimentais mostram que esta estratégia de comunicação pode melhorar o desempenho de consultas distribuídas. Trabalhos futuros envolvem alterações no processador analisado a fim de considerar o agrupamento no cálculo do custo de comunicação do otimizador de consultas.*

1. Introdução

A flexibilidade e a simplicidade do modelo de dados RDF (*Resource Description Framework*) tem motivado a proliferação de bases de dados que o adotam. Uma base RDF é composta por um conjunto de triplas *(sujeito,predicado,objeto)* e pode ser vista como um grafo, no qual sujeitos e objetos são representados por nodos ligados por arestas que representam os predicados, conforme mostra a Figura 1(a). De acordo com o consórcio *W3C*[1], bases de dados RDF comerciais têm alcançado o tamanho de 1 trilhão de triplas[2]. Logo, o processamento eficiente de consultas SPARQL (*SPARQL Protocol and RDF Query Language*) neste tipo de base tornou-se um grande desafio para os sistemas gerenciadores de dados RDF.

A adoção de abordagens centralizadas para o armazenamento de dados penaliza a escalabilidade em sistemas que processam grandes volumes de dados. Sendo assim,

[1]http://www.w3.org
[2]http://www.w3.org/wiki / LargeTripleStores

sistemas com armazenamento distribuído têm sido propostos. Nestes sistemas, tanto os dados quanto as consultas são distribuídas entre servidores a fim de promover o processamento distribuído e paralelo de consultas. Porém, a troca de dados entre servidores durante o processamento atinge diretamente no custo de comunicação no processamento de consultas [Ozsu and Valduriez 2011].

Em geral, sistemas RDF distribuídos adotam estratégias de comunicação que trocam resultados intermediários entre servidores durante o processamento distribuído de consultas SPARQL. [Penteado et al. 2016] denomina este tipo de estratégia como *send-result*. A unidade de comunicação utilizada nestes sistemas pode considerar um ou mais resultados intermediários. Segundo [Galárraga et al. 2012], o envio em blocos (grupos de resultados intermediários) pode minimizar o custo de comunicação nestes sistemas.

Este artigo implementa o conceito de blocos na estratégia *send-result* do sistema proposto em [Penteado et al. 2016] que processa consultas SPARQL por meio de um algoritmo de exploração distribuída de grafos. Estudos experimentais mostram que a agrupamento de resultados intermediários pode minimizar o tempo de processamento de consultas distribuídas em alguns cenários específicos.

O artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 apresenta os conceitos envolvidos no artigo; a Seção 4 apresenta a abordagem de agrupamento de resultados intermediários implementada no sistema analisado; a Seção 5 mostra experimentos baseados na abordagem de agrupamento, bem como os resultados alcançados; e, a Seção 6 conclui o artigo apresentando trabalhos futuros.

2. Trabalhos relacionados

A unidade de comunicação adotada no envio de resultados intermediários entre servidores pode variar entre os sistemas RDF distribuídos. Uma unidade de comunicação pode ser um único resultado intermediário ou um grupo de resultados intermediários.

[Galárraga et al. 2012] propõe um sistema que adota como unidade de comunicação um lote de 1024 resultados intermediários. Os autores justificam que o uso de blocos como unidade minimiza o custo de comunicação em sistemas distribuídos. Em contrapartida, sistemas tais como os propostos em [Rohloff:2010 2010] e [Goasdoué et al. 2013] adotam um único resultado intermediário como unidade de comunicação na comunicação *send-result*. Os dois sistemas usam o *framework MapReduce Hadoop*[3] no processamento distribuído de consultas. Neste *framework*, um plano de execução é composto por tarefas de mapeamento e redução. Uma tarefa de mapeamento gera um conjunto de pares `<chave,valor>` em um servidor. Cada par deste servidor é enviado e processado por uma tarefa de redução executada em um servidor remoto.

O sistema proposto em [Penteado et al. 2016] propõe um processador de consultas baseado em um algoritmo de exploração distribuída de grafos que viabiliza a escolha entre duas estratégias de comunicação durante a execução de consultas: o envio de resultados intermediários para outros servidores (*send-result*) ou a requisição de fragmentos necessários de outros servidores (*get-frag*). Entende-se por fragmento um conjunto de triplas RDF que segue uma estrutura pré-definida no momento da distribuição de dados nos servidores do sistema. A estratégia *send-result* adota um (único) resultado intermediário

[3] https://hadoop.apache.org/docs/r1.2.1/index.html

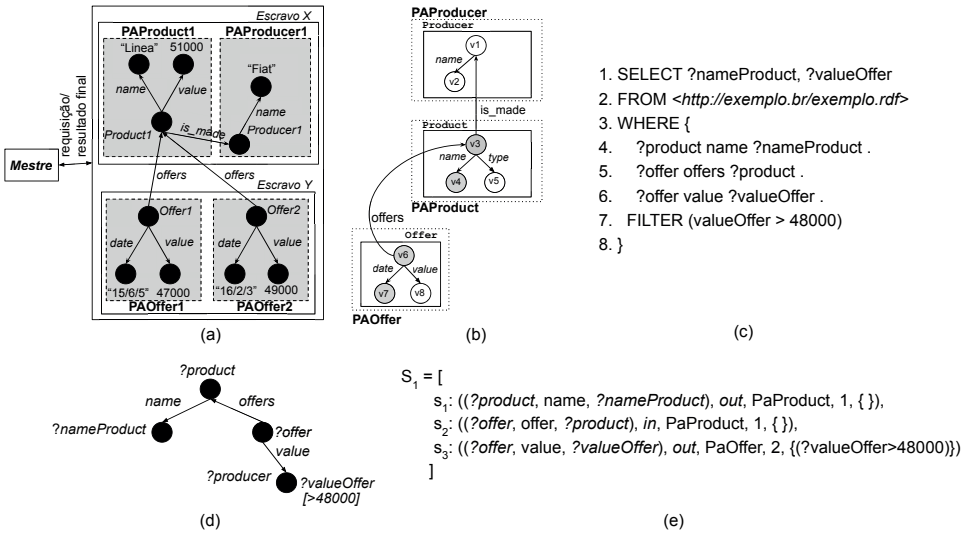


Figura 1. Base RDF (a); Grafo de sumarização da base RDF (b); Consulta SPARQL (c); Padrão de grafo básico da consulta (d); Plano de execução da consulta (e)

como unidade de comunicação e *get-frag*, um fragmento de dados. Este artigo estende o processador implementando o conceito de agrupamento de resultados intermediários em sua estratégia *send-result*. Assim, a estratégia pode utilizar blocos como unidade de comunicação, que resulta a redução do custo de comunicação no sistema.

3. Definições preliminares

Uma base RDF é composta por um conjunto de triplas *(sujeito,predicado,objeto)* representando que um sujeito tem uma propriedade com um objeto. Bases RDF podem ser definidas como um conjunto de nodos com suas listas de adjacência. Por exemplo, na base RDF da Figura 1(a), o conjunto de arestas incidentes em *Producer1* é *in*:{(is_made, {Product1})} e o conjunto de arestas que partem, *out*:{(name, *"Fiat"*)}.

O núcleo da sintaxe da linguagem SPARQL é baseado em um conjunto de padrões de triplas semelhantes às triplas RDF, que admitem variáveis em seus termos. A consulta da Figura 1(c) recupera, para cada produto, o seu nome e os valores de suas ofertas maiores que *48000*. O conjunto de padrões de triplas de uma consulta SPARQL constitui um padrão de grafo denominado de padrão de grafo básico (PGB) da consulta. A Figura 1(d) representa o PGB da consulta da Figura 1(c), onde os sujeitos e os objetos dos padrões de triplas são representados por nodos e os predicados por arestas. O filtro da consulta está representado entre colchetes abaixo do nodo apropriado. O processamento de consultas SPARQL pode ser transformado para um problema de casamento de grafos onde subgrafos da base RDF que são homomórficos ao PGB de uma consulta são recuperados.

O processador de consultas SPARQL proposto em [Penteado et al. 2016] tem como base um algoritmo de exploração distribuída de grafos que adota um método de comunicação híbrido. Ele adota uma arquitetura mestre-escravo, na qual fragmentos de dados são distribuídos nos escravos de um *cluster* por meio de padrões de alocação (PAs) previamente definidos a partir do grafo de sumarização da base, que representa seu esquema. Fragmentos de dados são instâncias de PAs que seguem sua estrutura. Os fragmentos garantem co-alocação no armazenamento, além de serem usados como unidades de comunicação no sistema. A Figura 1(b) apresenta o grafo de sumarização G_S da base da

Figura 1(a). G_S possui três PAs que estão representados por figuras tracejadas e nomeados de acordo com a classe que os compõem. Os PAs adotam o padrão estrutural do tipo estrela, garantindo que cada sujeito que representa um recurso seja armazenado com os seus respectivos objetos literais. Fragmentos estão representados por meio de figuras cinzas na Figura 1(a). Por exemplo, os fragmentos *PaOffer1* e *PaOffer2* da Figura 1(a) são instâncias do padrão de alocação *PAOffer* da Figura 1(b).

O planejamento de consultas do processador baseia-se nos PAs de G_S a fim de minimizar a comunicação entre servidores durante o processamento de consultas. O subgrafo destacado na Figura 1(b) representa o subgrafo homomórfico ao grafo PGB da Figura 1(d). A Figura 1(e) mostra o plano de consulta gerado. Cada passo s_i do plano é uma tupla (a, dir, pat, id, f), onde: (**1**) a é um padrão de tripla da consulta, (**2**) $dir \in \{in, out\}$; se $dir = out$ a exploração é do sujeito para o objeto. Caso contrário, a exploração é do objeto para o sujeito; (**3**) pat é o PA que contém a; (**4**) id é um identificador único associado ao PA. Todos os passos com um mesmo id compõem um bloco de exploração dentro de um mesmo fragmento. (**5**) f é um conjunto de filtros definidos em a.

Uma vez gerado o plano da consulta, o mestre inicia a execução enviando o plano para todos os escravos do *cluster*. Cada escravo inicia a execução em paralelo recuperando os pontos iniciais de exploração de acordo com o plano, que correspondem a nodos de fragmentos do primeiro passo do plano (s_1). Ou seja, nodos de fragmentos de $s_1.pat$ alocados no escravo são recuperados para obter as triplas que casam com o padrão $s_1.a$. Caso a direção de exploração $s_1.dir$ seja out, os pontos iniciais de exploração são os sujeitos das triplas; caso contrário, os pontos iniciais são os objetos das triplas. O resultado da execução de um passo é a associação de variáveis a nodos da base. Na execução do passo s_1 do exemplo, o escravo X recupera *Product1* gerando o mapeamento $[\![s_1]\!]_X$: $\{(?product \mapsto Product1, ?nameProduct \mapsto \text{“}Linea\text{”})\}$. Na sequência, cada passo s_i do plano adiciona novos mapeamentos. Se $s_i.a$ não é mapeado para a base local ou se o novo mapeamento não satisfaz $s_1.f$, o mapeamento correspondente é removido do conjunto resultante. No exemplo, a execução dos passos s_1 e s_2 é feita no escravo X uma vez eles possuem o mesmo id. Após o passo s_2, o resultado intermediário gerado por X é $[\![s_2]\!]_X$: $[\![s_1]\!]_X \cup \{(?offer \mapsto \{Offer1, Offer2\})\}$. s_3 pode ser executado ou não em X, uma vez que $s_2.id \neq s_3.id$. No exemplo, as ofertas estão armazenadas no servidor Y. Logo, X se comunica com Y para continuar a execução. Neste momento, ou X requisita os fragmentos das ofertas para Y e continua o processamento localmente (usando a estratégia *get-frag*), ou o resultado intermediário gerado é enviado para Y continuar a execução do plano (usando a estratégia *send-result*). O otimizador de consultas escolhe uma das estratégia por meio de funções de custo. A escolha depende do número de mensagens e do volume de dados a ser transmitido entre os escravos. Continuando a execução, o resultado intermediário gerado para *Offer1* é descartado na execução de s_3, uma vez que o valor da oferta não satisfaz o filtro $s_3.f$. Por fim, quando um escravo explora o último passo do plano, o seu conjunto de mapeamentos é projetado usando as variáveis da cláusula *SELECT* da consulta e o resultado final é enviado para o mestre.

4. Agrupamento de resultados intermediários

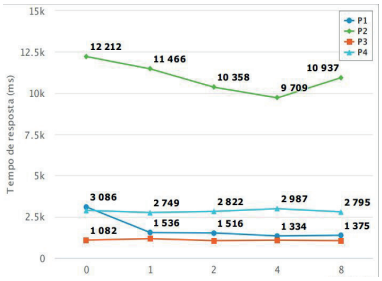
A abordagem proposta neste artigo viabiliza a escolha do tamanho do bloco de resultados intermediários usado na estratégia *send-result*. A abordagem envolve duas operações, uma de agrupamento e outra de desagrupamento de resultados intermediários.

```
Algoritmo: agrupamento de resultados intermediários
Entrada: [L, fGroup]
1. accSize := 0;
2. groupedResults := { };
3. For each l in L
4. {
5.      currentSize := size(l);
6.      accSize := accSize + currentSize;
7.      groupedResults := groupedResults ∪ {l} ;
8.      If (index(l)=size(L)) or (accSize>=fGroup)
9.      {
10.             send(header,groupedResults);
11.             accSize := 0;
12.             groupedResults := { };
13.     }
14.}
```

(a)

P	#fGroup	#Ag	#Msg	#Vol	#Rtime
$P1$	0	1	1222	296	3086
$P1$	1	4	306	1184	1536
$P1$	2	7	175	2072	1516
$P1$	4	14	88	4144	1334
$P1$	8	28	44	8288	1375
$P2$	0	1	1222	1284	12212
$P2$	1	1	1222	1284	11466
$P2$	2	2	611	2444	10358
$P2$	4	4	306	4888	9709
$P2$	8	7	175	8554	10937
$P3$	0	1	57	902	1082
$P3$	1	2	29	1804	1167
$P3$	2	3	19	2706	1049
$P3$	4	5	15	4510	1078
$P3$	8	10	6	7216	1047
$P4$	0	1	57	1233	2884
$P4$	1	1	57	1233	2749
$P4$	2	2	29	2466	2822
$P4$	4	4	15	4932	2987
$P4$	8	7	9	8631	2795

(b)



(c)

Figura 2. Algoritmo de agrupamento (a); Dados sobre agrupamento variando *fGroup* de 0–8 nos planos $P1$–$P4$ (b); Relação entre *fGroup* e tempo de execução dos planos (c)

O algoritmo da Figura 2(a) descreve a operação de agrupamento usada por um escravo que usa um fator de agrupamento (*fGroup*) no momento do envio de uma lista de resultados intermediários (L). Este fator define um limiar para o volume da mensagem em *bytes* que será enviada ao servidor remoto implicando diretamente no número de resultados intermediários agrupado em cada mensagem. A variável *accSize* monitora o volume da mensagem (linha 6) e *groupedResults* o agrupamento dos resultados intermediários (linha 7). Caso o último resultado intermediário da lista L seja considerado no agrupamento ou o tamanho da mensagem alcance ou ultrapasse o limiar definido para a mensagem (linha 8), o grupo é enviado para o escravo remoto (linha 10). Cada envio tem um cabeçalho a fim de auxiliar a operação de desagrupamento. O cabeçalho indica dados como o número de resultados intermediários do bloco e a finalização do envio do bloco.

A operação de desagrupamento é iniciada no servidor remoto com o recebimento de uma mensagem. O desagrupamento ocorre em paralelo com a operação de recebimento de mensagens. Cada mensagem recebida é desagrupada e os seus resultados intermediários são armazenados no servidor remoto. A exploração de grafos continua no servidor remoto assim que todos os resultados intermediários são recebidos do escravo emissor.

5. Estudo experimental

O objetivo dos experimentos foi a análise do impacto do fator de agrupamento no tempo de execução de consultas. A abordagem de agrupamento foi implementada no sistema proposto em [Penteado et al. 2016] usando Java e o protocolo de comunicação TCP/IP. A biblioteca GSON[4] foi usada: no envio de mensagens, mapeando listas de resultados intermediários para o formato JSON[5]; e, no desagrupamento, convertendo um fluxo de *bytes* em uma lista de resultados intermediários. *fGroup* variou entre 0, 1, 2, 4 e 8 *kilobytes*.

O experimento foi realizado em um *cluster* com três servidores, sendo um mestre e dois escravos. Uma base com 1.875.815 triplas foi gerada a partir do benchmark Berlin[6] e distribuída entre os dois escravos. a Figura 1(b) mostra o grafo de sumarização da base. Quatro planos de execução foram usados no experimento envolvendo *Product* e *Producer*.

Os escravos comportaram-se de maneira similar no experimento. A Figura 2(b) refere-se às mensagens enviadas por um escravo durante a execução dos planos. Para cada

[4]https://google.github.io/gson/apidocs/com/google/gson/Gson.html
[5]https://www.oracle.com/technetwork/articles/java/json-1973242.html
[6]http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

execução, a figura mostra: o plano executado (P), o fator de agrupamento (*#fGroup*), o número de resultados intermediários agrupados (*#Ag*), o número de mensagens enviadas (*#Msg*), o volume médio de cada mensagem (*#Vol*) e o tempo médio de execução do plano em milisegundos (*#Rtime*). Por exemplo, quando *#fGroup*=1 em $P1$, o servidor enviou 306 mensagens, cada uma com 4 resultados intermediários e volume médio de 1.184 *bytes*. O tempo médio da execução de $P1$ foi 1.536 ms. Observe que os planos variaram o *número* e o *volume* de resultados intermediários enviados do escravo, conforme mostra a Figura 2(b) quando *#fGroup*=0. Em $P1$ e $P2$ foram enviados 1.222 resultados intermediários de volume 296 e 1.284 *bytes*, respectivamente. Em $P3$ e $P4$ foram enviados 57 resultados intermediários de volumes 902 e 1.233 *bytes*, respectivamente. Vale destacar que o fator de agrupamento 0 corresponde ao *send-result* original do sistema.

A Figura 2(c) mostra o comportamento do tempo de resposta das execuções com a variação de *fGroup* nos quatro planos. É possível notar que $P1$ e $P2$ obtiveram vantagem com o agrupamento. Em contrapartida, o mesmo não ocorreu com $P3$ e $P4$. O desempenho de $P1$ e $P2$ melhorou conforme *fGroup* aumentou de 0 até 4. Porém, o mesmo não ocorreu quando *fGroup* foi de 4 para 8. Apesar da redução do número de mensagens (*#Msg*) conforme pode ser constatado na Figura 2(b), notou-se o custo da operação de desagrupamento quando o número de resultados intermediários agrupados (*#Ag*) aumentou. Em relação à $P1$, o volume (*#Vol*) penalizou $P2$ aumentando o número mensagens trocado entre os escravos e o tempo de execução. Por fim, pode ser observado que $P2$ e $P4$ possuem o volume médio de cada resultado intermediário similar, 1.284 e 1.233 *bytes*, respectivamente. $P2$ enviou 1.222 resultados intermediários para o escravo remoto e $P4$, 57. O tempo de execução de $P4$ não melhorou com o aumento de *fGroup*, como ocorreu com $P2$. Logo, notou-se que o agrupamento melhorou o desempenho de planos que trocaram um número significativo de resultados intermediários entre pares de escravos.

6. Conclusão

Este artigo apresenta uma análise sobre o agrupamento de resultados intermediários pela estratégia de comunicação *send-result* de um processador de consultas SPARQL baseado na exploração distribuída de grafos. A abordagem de agrupamento analisada viabiliza o uso de diferentes tamanhos de bloco na estratégia. Resultados experimentais mostram que a abordagem pode otimizar o processamento de consultas em algumas situações. Trabalhos futuros envolvem o uso de agrupamento no cálculo do custo de comunicação da estratégia *send-result* do sistema analisado. Em cada comunicação, o otimizador do processador poderá optar ou não pelo agrupamento, escolhendo o volume de bloco adequado.

Referências

Galárraga, L., Hose, K., and Schenkel, R. (2012). Partout: A Distributed Engine for Efficient RDF Processing. volume abs/1212.5636.

Goasdoué, F., Kaoudi, Z., Manolescu, I., Quiané-Ruiz, J., and Zampetakis, S. (2013). CliqueSquare: efficient Hadoop-based RDF query processing. In *BDA'13 - Journées de Bases de Données Avancées*.

Ozsu, M. T. and Valduriez, P. (2011). *Principles of Distributed Database Systems, 3rd Ed.*

Penteado, R. R. M., Schroeder, R., and Hara, C. S. (2016). Exploring controlled RDF distribution. In *IEEE CloudCom 2016, Luxembourg, December 12-15, 2016*, pages 160–167.

Rohloff:2010 (2010). High-performance, Massively Scalable Distributed Systems Using the MapReduce Software Framework: The SHARD Triple-store. In *Programming Support Innovations for Emerging Distributed Applications*, pages 4:1–4:5, New York, NY, USA. ACM.

Achieving Differential Privacy in Smart Home Scenarios

Israel C. Vidal[1], **Franck Rousseau**[2], **Javam C. Machado**[1]

[1]Universidade Federal do Ceará (UFC), Fortaleza - CE - Brazil

[2]Université Grenoble Alpes, CNRS, Grenoble INP, LIG, Grenoble - France

{israel.vidal, javam.machado}@lsbd.ufc.br, Franck.Rousseau@imag.fr

Abstract. *With the growth of the Internet of Things (IoT) and Smart Homes, there is an ever-growing amount of data coming from within people's houses. These data are intrinsically private and should be treated carefully, despite their high value for analysis. In this work, we propose a differentially private strategy to estimate frequencies of values in the context of Smart Home data.*

1. Introduction

With the popularization of the Internet of Things (IoT) and the greater availability of various kinds of sensors in the market, there is an increasing amount of data being generated. We expect that by the year 2021, the amount of data generated by IoT devices, people and machines will reach the magnitude of zettabytes. These data can be beneficial for improving services, for example, by using Smart Meters data to gain a better understanding of the energy use in a city. However, careful attention to the privacy of these data is becoming more urgent. The lack of care with privacy can lead to severe problems, as shown in [Molina-Markham et al. 2010], where, through relatively simple statistical methods, one is able to identify crucial private information, such as if any household member watched the game on a given night or if the household members were late for work.

Although many works in the literature implement privacy through a trusted entity who has access to the raw data of a set of users, in real-world scenarios it is often not reasonable to depend on such an entity. More recent works focus on the local perspective of privacy, where the privacy process is done closer to the user without depending on a trusted third party entity. Besides that, IoT data often appears as streaming data, which brings some challenges due to its intrinsic characteristics, i.e., the data is potentially unbounded and happens in a non-predictable order.

In Differential Privacy (DP) [Dwork and Roth 2014], a mechanism $\mathcal{M}$ is said to be differentially private if the probability of any output of $\mathcal{M}$ does not vary significantly, by a threshold of ε, independent of the input. DP was initially proposed to work as an interactive model [Dwork et al. 2006], responding privately to statistical queries in a database. In this interactive scenario, a trusted entity that has access to the raw data is necessary. However, in more recent work, such as [Erlingsson et al. 2014], there has been significant interest in the local version of this model, called Local Differential Privacy (LDP), where a randomization process is done locally to ensure the definition of DP.

In this paper, we present a strategy that guarantees Local Differential Privacy for estimating frequencies of values in the context of Smart Homes. To evaluate our work, we have used real sensor data from Smart Meters [UK Power Networks 2015].

2. Related Works

The authors of [Molina-Markham et al. 2010] tackle the problem of privately charging energy consumption. In addition to proposing a statistical procedure capable of identifying house activities in fine-grained measurements, showing that there must be a meticulous privacy procedure to make use of smart meter data, they describe a protocol that allows smart meters to report a bill without revealing how the energy was used. The procedure uses cryptography and zero-knowledge proof to guarantee that the company will not have access to the information from which house comes a given data, even though the company will be able to charge for the energy used. A downside of this work is that they still have access to what they call *blinded data*, which consists of the data from all houses with the identification removed, and this is not enough for guaranteeing privacy.

The work [Ács and Castelluccia 2011] uses differential privacy to deal with the problem of using consumption data to learn privately about users. The approach is based on the Laplace Mechanism, which adds a noise sampled from a Laplace distribution to the result of a numerical query. The authors propose a Distributed Laplace Mechanism (DLM). The information they want to learn privately in this work is the consumption summation of N houses in a given time. To do this, each house adds a small noise n to its consumption c (which is not enough to guarantee DP), and encrypts the report $r = c + n$ in a way that the server is not able to decrypt a report alone, but it is able to decrypt the summation of the reports. The server, then, has $S = \sum_{h=1}^{N} c_h + n_h = \sum_{h=1}^{N} c_h + N$, where N is a noise that follows the Laplace distribution as previously presented, i.e. that is enough to guarantee DP. This strategy is useful for learning the summation of consumption, but cannot be used to learn more information than that.

IoT data often appear as data streams. Works that deals with the problem of guaranteeing privacy in the context of streaming data deal with additional complexity because streaming data is potentially unbounded and continuously generated at rapid rates. The work [Leal et al. 2018] proposes a strategy to estimate the sensitivity and also presents a microaggregation algorithm that is capable of enhancing the utility for publishing differentially private data using the Laplace Mechanism in the context of streaming data. This work depends on a trusted third party entity to achieve its privacy, which may not be acceptable in the context of IoT data and smart homes.

The work [Cao and Yoshikawa 2015] uses differential privacy to publish statistics about streaming of trajectories. The objective is to publish, for a defined set of possible locations, how many people are in each location at a given time. The authors use the concept of a l-trajectory, i.e., a trajectory of size l. They show that it is possible to guarantee that a l-trajectory is DP. The concept of l-trajectory proposed is relevant and gives us an insight that to solve the privacy problem in streaming scenarios it may be useful to simplify the problem in order to be able to achieve a solution. On the other hand, this work, besides depending on a third party trusted entity, need to know beforehand the set of locations, which may not be reasonable in real-world scenarios.

3. Proposal

Our proposal aims to provide LDP for users, in the context of Smart Homes, that agree to provide their data to entities, so those entities can learn privately from data produced inside peoples' houses and, for example, provide better services. From now on, we will

consider that this entity is the Service Provider (SP), but in real-world scenarios, it would be possible that they were two separated entities. We do not tackle the problem of charging for consumption, as described in [Molina-Markham et al. 2010].

It is necessary to collect Smart Home data privately because those data are intrinsically sensitive and the lack of proper care in their management could lead to harmful inferences, e.g., the energy producer entity could learn, using fine-grained energy measurements, when a given house is empty.

The solution was thought to work over an edgeOS, i.e., a specialized operating system that runs in an edge gateway, from now on called edgeBox, and manages smart things. In this paper, we have omitted the full architecture of an edgeOS, but [Shi et al. 2016] can be checked for more details. For our proposal, what is important about an edgeOS is that there is a data abstraction layer in it that gathers data produced by things inside a house. Our solution works between the data abstraction layer and all external communication to guarantee that all data that goes outside the house is private. A possible exception for this is that for the SP to be able to charge for the consumption, it may need to have access to coarse-grained measurements. As shown in section 2, there are possible strategies to charge privately.

Figure 1 illustrates the scenario where the SP has access to the total consumption in order to charge for it (ideally in a private way) and uses the Privacy Gateway proposed in this paper to learn from the data generated by things inside houses. The process executed by the Privacy Gateway will be detailed next.

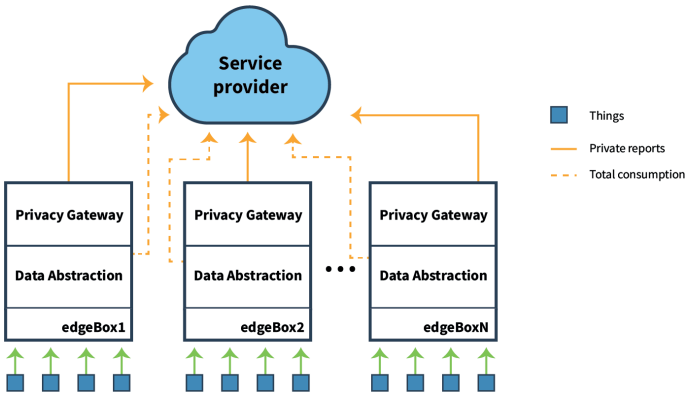


Figure 1. Communication between Service Provider and the N selected houses

The solution works as follows: the SP contacts a number N of people (the greater, the better) in different houses, and offers them something, e.g., a discount on the energy bill, in exchange for a defined privacy budget ε to be consumed in a number k of reports. Notice that a smaller budget means a more private output. The SP sends, then, a set H of parameters to each participant p. Every participant must use these parameters in H, so the SP can later decode the reports and get utility from them. The parameters are: the number of bins, the ranges of each bin, the number k of reports and the time δ after which each report will be sent. Given that the value to be sent is likely to be a real number in the context of IoT data, we will transform this value into a discrete form, in order to be able to make it private so that we can later get information from it. To do this, we will use a histogram representation of the values.

Each p will report, after some time δ, its private value v, as defined at the beginning of the process. The method of privately reporting consists of encoding the value v in a bit-array B of length equal to the number of bins, where only the bit corresponding to the range that contains v is set to 1, and the other positions are set to 0. This procedure is called Unary Encoding (UE) [Wang et al. 2017].

After encoding v into B, the next step is to bitwise perturb B in a differentially private manner. Let B' be the differentially private version of B. The process of obtaining B' consists in keeping the bit value of B with a probability p and changing it with a probability $q = 1-p$. In order for this process to achieve differential privacy it is necessary that $\frac{P[B|v1]}{P[B|v2]} \leq e^{\varepsilon_i}$. As shown in [Wang et al. 2017], $p = \frac{e^{\frac{\varepsilon_i}{2}}}{e^{\frac{\varepsilon_i}{2}}+1}$ and $q = \frac{1}{e^{\frac{\varepsilon_i}{2}}+1}$ are sufficient for this property to happen.

Notice that, as we want to send a number k of reports privately, we cannot consume all the privacy budget ε in a single report. Therefore, for each single private report, we will use $\varepsilon_i = \frac{\varepsilon}{k}$. This strategy will guarantee that, if continuously reported, any window of k consecutive reports is ε-differentially private. The guarantee comes from the sequential composition property of differential privacy [McSherry 2009].

With B' adequately generated, the Privacy Gateway can finally send the differentially private version of v to the SP, which will then gather it with the reports from the other N houses to obtain information from it. The process of obtaining information from the differentially private reports consists in constructing the histogram $Hist$. Each bin i of $Hist$ is obtained with the summation of the i^{th} element from all reports, $Hist[i] = \sum_{j=1}^{N} B_j[i]$. It is important to remember that each $Hist[i]$ contains not only the reports that were truly reported for the i^{th} bin, but also some noise added by the differentially private mechanism.

Therefore, the next step is to get an unbiased estimation for $Hist$, which we will call Unb_Hist. To calculate Unb_Hist, we need to get rid of the noise added for each bin, which can be done in the following way: $Unb\_Hist[i] = \frac{Hist[i]-Nq}{p-q}$, where p is the probability of keeping the bit value and q is the probability of inverting a bit used in the process of creating the private reports. N is the number of reports. For the sake of space, we let the reader refers to [Wang et al. 2017] for the proof that this yields an unbiased estimation. Notice that if we have an unpopular bin, i.e., the number of reports ($Hist[i]$) is small, our unbiased estimation can be negative. When this happens, the considered value for $Unb\_Hist[i]$ will be zero.

As the SP have selected N different participants to send k reports spaced by a δ time, the process of calculating $Unb\_Hist[i]$ could be performed in one of two ways: (i) using each set of size N separately, which keeps the notion of time and (ii) using the union of all data with size $k * N$. This strategy may yield more accurate frequencies for the generated histogram since there is a more significant number of reports, but the notion of time is lost. In our evaluation, we have opted to use strategy (i), since we believe the time attribute is essential in the context of IoT data and Smart Cities.

4. Evaluation

For the experimental evaluation, we have used real sensor data that consists of energy consumption readings from 5,567 London households generated between 2011 and 2014

as part of the Low Carbon London project. There are 167 million rows. We have used the attribute "KWH/hh (per half hour)."

In order to better evaluate our proposal, we have sampled rows from the data set to simulate a fixed number N of houses. The number of samples k to be sent from each house was fixed in 10, thus, the privacy budget ε_i used for a single report is equal to $\frac{\varepsilon}{k} = \frac{\varepsilon}{10}$. We have tested the following values for ε: 1, 2, 3, 4, 10. Therefore, the values used for ε_i were: 0.1, 0.2, 0.3, 0.4, 1. Notice that the choice of using a single value for k does not have a significant impact on the results since it has a direct influence on ε, which we have tested for different values.

The varying values for ε and N give us a better understanding of how these two variables impact the utility, as can be seen in Figure 2. To measure the utility we have evaluated the histogram intersection, given by $\frac{\sum_{i=1}^{nb} min(Ori\_Hist[i], Unb\_Hist[i])}{\sum_{i=1}^{nb} Unb\_Hist[i]}$, where nb is the number of bins, Unb_Hist is the histogram generated by our strategy and Ori_Hist is the original histogram. The histogram intersection measures how similar does our proposal generate the histogram compared with the histogram of the original data. The number of bins used in the experiments was 100. The maximum and minimum values in the data set are 0.0 and 10.76, respectively, and each bin used has equal width.

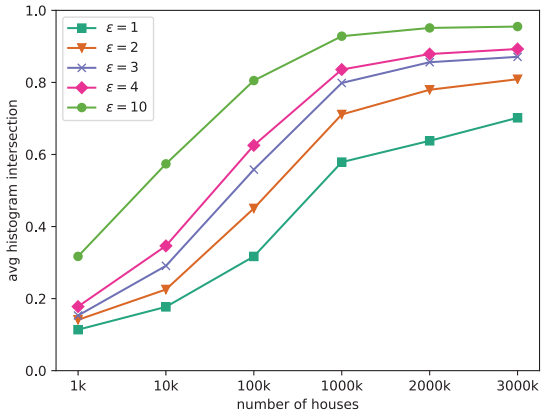


Figure 2. Average histogram intersection by number of houses. Varying ε.

Figure 2 shows, for each value of ε, the average histogram intersection of 10 different execution of our proposal which simulates the process of each house reporting $k = 10$ times and, after each set of reports (from N houses), the Service Provider calculates the unbiased histogram. It is possible to observe that for a small number of houses, the solution outputs a low histogram intersection, but as N grows, we get more accurate results. The reason why this happens is that when there are few reports, we cannot cancel out the noise added by the differential privacy mechanism. Remember that, the higher the ε, the weaker the privacy guarantee.

5. Conclusion

In this paper, we have proposed a practical solution for estimating the frequency of values issued from houses' IoT devices in a differentially private manner. It allows an energy

provider to collect house consumption for analytics and still provides privacy for individuals living in the house. Data utility depends on the number of houses and the available privacy budget. For one million houses, our preliminary results have reached around 80% of data utility with a privacy budget of 3 to 4, which is very reasonable. For the next steps, it might be valuable to adapt the strategy to work for consecutive windows. Notice that this is not a trivial problem and could demand sophisticated adaptations in order to be solved.

Acknowledgments

This research was supported by FUNCAP and LSBD/UFC.

References

Ács, G. and Castelluccia, C. (2011). I have a dream!(differentially private smart metering). In *International Workshop on Information Hiding*, pages 118–132. Springer.

Cao, Y. and Yoshikawa, M. (2015). Differentially private real-time data release over infinite trajectory streams. In *2015 16th IEEE International Conference on Mobile Data Management*, volume 2, pages 68–73. IEEE.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

Dwork, C. and Roth, A. (2014). The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407.

Erlingsson, Ú., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM.

Leal, B. C., Vidal, I. C., Brito, F. T., Nobre, J. S., and Machado, J. C. (2018). δ-doca: Achieving privacy in data streams. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 279–295. Springer.

McSherry, F. D. (2009). Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30. ACM.

Molina-Markham, A., Shenoy, P., Fu, K., Cecchet, E., and Irwin, D. (2010). Private memoirs of a smart meter. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 61–66. ACM.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646.

UK Power Networks (2015). SmartMeter Energy Consumption Data in London Households. `https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households`. Accessed: 2019-06-28.

Wang, T., Blocki, J., Li, N., and Jha, S. (2017). Locally differentially private protocols for frequency estimation. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 729–745.

A Science Gateway to Support Research in Spectral Graph Theory*

Daniel Oliveira[1], **Carlos Magno Abreu**[1], **Eduardo Ogasawara**[1], **Eduardo Bezerra**[1], **Leonardo de Lima**[2]

[1]Federal Center of Technological Education of Rio de janeiro - CEFET/RJ, Brazil

[2]Paraná Federal University - UFPR, Curitiba, PR, Brazil

{daniel.oliveira, ebezerra, eogasawara}@cefet-rj.br

leonardo.delima@ufpr.br, magno.mabreu@gmail.com

Abstract. *Describing classes of graphs that optimize a function of the eigenvalues subject to some constraints is one of the topics addressed by Spectral Graph Theory (SGT). In this paper, we propose RioGraphX, a science gateway developed on top of Apache Spark, which aims to obtain all graphs that optimize a given mathematical function of the eigenvalues of a graph. Initial experiments involving small graphs have pointed out optimal graphs in a reasonable computational time, and also have shown that leveraging parallel processing is a promising approach to handle larger graphs.*

1. Introduction

Spectral Graph Theory (SGT) is a research area which aims to obtain structural properties of a graph from eigenvalues and eigenvectors of matrices related to its graph. Given a graph $G(V, E)$ with vertex set V of cardinality n and edge set E of cardinality m, a set of matrices can be associated with G. The most used representations for graphs in SGT are the adjacency, the Laplacian, and the signless Laplacian matrices. The largest and the smallest non-zero eigenvalues of those matrices are strongly related to structural properties of the graph. For instance, one can cite the second smallest Laplacian eigenvalue of a graph, called the algebraic connectivity of the graph. This parameter is associated with the connectivity of the graph such that G is connected if and only if the algebraic connectivity of G is positive [Mohar et al., 1991]. For other connections on the eigenvalues of the adjacency, Laplacian and signless Laplacian to invariants of graphs, we refer the reader to [Cvetkovic et al., 2007], [Wilf, 1967], and [Bomze et al., 1999]. The SGT community has increased over the last years after the very first work of Dragos Cvetkovic thesis [Cvetković, 1971]. Since then, applications in many areas have been reported, and in particular in Computer Science [Cvetković and Simić, 2011]. Computational tools aiming to help in either proposing or refuting conjectures and describing families of graphs satisfying some properties have been developed in the last few years. For instance, the AutoGraphiX [Caporossi and Hansen, 2000] is a heuristic tool which has been used to solve many open problems in the literature and has led to the publication of over 20 papers, all of them related to problems of SGT. This fact shows how computer systems that aid theoretical researchers can be useful to propose the right mathematical conjectures or to refute

*Os autores agradecem à FAPERJ, à CAPES (código 001) e ao CNPq pelo financiamento do projeto.

them. Other relevant computational tools in SGT are NewGraph [Brankov et al., 2006], MathChem [Vasilyev and Stevanović, 2014], and Graph6Java [Ghebleh et al., 2019]. It is worth mentioning that all these computational tools for SGT run monolithically on desktop computers and none of them are available online. Besides, most of them require some level of coding in a specific programming language. In order to verify a conjecture, one may code routines generating all graphs for a given range of n vertices and look for counterexamples, which is a very time-consuming process. The limitations of these tools can be seen in Table 1. In fact, due to the lack of scalability, evaluating conjectures for graphs with ten or more vertices becomes a computational barrier while using these tools.

In this paper, we propose RioGraphX, a science gateway published as a Web application to aid SGT researchers either in the investigation of counterexamples for an existing conjecture or in describing classes of graphs that optimize a given function under some possible constraints. RioGraphX does an exhaustive search to find graphs in a given range of orders that optimize some user-provided combinatorial function (also known as extremal graphs). A workflow defined in the Apache Spark environment [Zaharia et al., 2016] searches for extremal graphs using a parallel and distributed computational infrastructure, by calculating their invariants and producing a ranked list of graphs that optimize the given function. This ranked list is finally presented to the researcher for further analysis. RioGraphX implements a workflow which integrates well-known packages, such as NetworkX [Hagberg et al., 2008], GraphX [Xin et al., 2013], and GraphFrames [Dave et al., 2016] and is innovative in the sense that it makes use of a scientific workflow approach with distributed and parallel processing. It is worth mentioning that Paralell BGL [Gregor and Lumsdaine, 2005] is another alternative of parallel and distributed graph packages, which we intend to explore in future works. Interesting features of RioGraphX include: (i) no usage of processing resources in the user's machine; (ii) provides an interactive online user interface, where the users can formulate mathematical conjectures and test them without coding computational routines; (iii) users can monitor the execution of their submissions. Also, a single user can submit several jobs simultaneously and all results are displayed in the report folder area; (iv) provides a PDF report containing the k top extremal graphs that optimize a mathematical function given as an input by the user; (v) makes use of parallel and distribution execution (through Spark) to generate all graphs that optimize a given function with some predefined constraints. We conducted initial computational experiments using a conjecture proposed in the literature for graphs with different orders and obtained reasonable speedup results.

2. The RioGraphX Science Gateway

The RioGraphX is a science gateway built on top of Spark environment and makes usage of Python, Nauty package, NetworkX, and GraphFrames. Table 1 presents the RioGraphX functionalities compared to five similar tools. It can be seen that it has the main features of other tools and also presents ease of use, accessibility (via a Web interface) and parallel distributed execution (integrated with Spark). The workflow of activities executed by RioGraphX is summarized in Figure 1. The RioGraphX deals with a huge computational challenge of a potentially large number of graphs to be processed, even for small graphs for each job submission. In Figure 1, we describe the activities of the RioGraphX workflow, and how it uses Spark parallelization to distribute the processes. All invariants of a graph implemented in the system are described in Table 2.

Table 1. Functional characteristics of SGT computational tools

Functionalities	NewGraph	MathChem	AutoGraphiX	Graph6Java	RioGraphX
Manipulation of g6 files	No	Yes	Yes	Yes	Yes
Interactive environment	Yes	No	Yes	No	Yes
Programming skills needed	No	Yes	No	Yes	No
Parallel and distributed execution	No	No	No	No	Yes
Accessible from anywhere	No	No	No	No	Yes
Search for extremal graphs	No	No	Yes	Yes	Yes

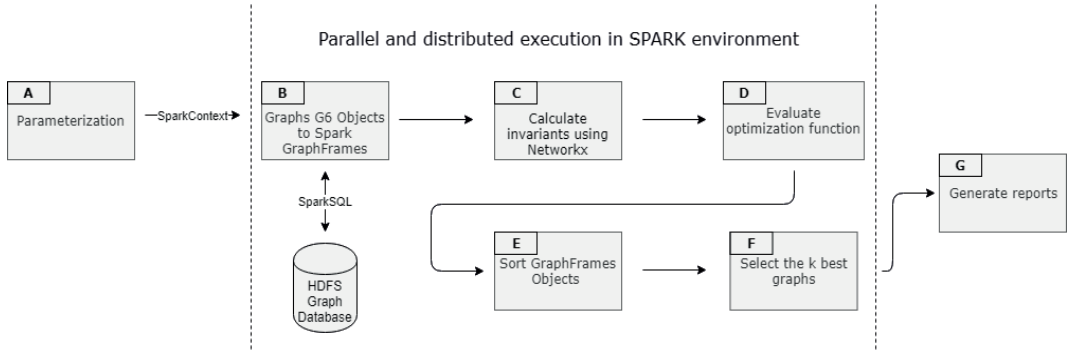


Figure 1. The workflow implemented in RioGraphX

The front-end of RioGraphX consists of a Java WEB application where users can create a RioGraphX account. Once authenticated, the user has access to the submission form (see Figure 2) where configuration parameters are filled to submit the job. In step **A** of the workflow, the user configures the experiment (job) related to the conjecture to be assessed (see Figure 2). First, the analytical form of a function is typed in Latex format. More specifically, the function has the type $f(x_1, x_2, \ldots, x_t)$, where each x_i, for $i = 1, 2, \ldots, t$ can be one of the following graph invariants of the Table 2. After defining the function f, the optimization type (either maximizing or minimizing f) is defined and also the number k of graphs that should either minimize or maximize f. Table 3 shows the available constraints to be added to the problem. Such constraints decrease the search space of graphs to be selected.

Through an account, the user can monitor the submission status (processing or terminated) of all requests already made. Other screens of the system are not presented here due to lack of space. Once a job submission is placed into the system, RioGraphX

Table 2. List of parameters available for the definition of the optimization function

Graph invariants	Description
$[n_{min}, n_{max}]$	minimum and maximum order of a graph
d_i	i-th largest degree of a graph
λ_i	i-th largest eigenvalue of a graph
μ_i	i-th largest Laplacian eigenvalue of a graph
q_i	i-th largest signless Laplacian eigenvalue of a graph
χ_G	chromatic number of a graph
ω_G	clique number of a graph

Table 3. Available constraints in RioGraphX system

Constraints	Description
Graphs free of triangles	generation of triangle-free graphs only
Connected Graphs	generation of connected graphs only
Bipartite Graphs	generation of bipartite graphs only
k	number of graphs to be included in the report after execution

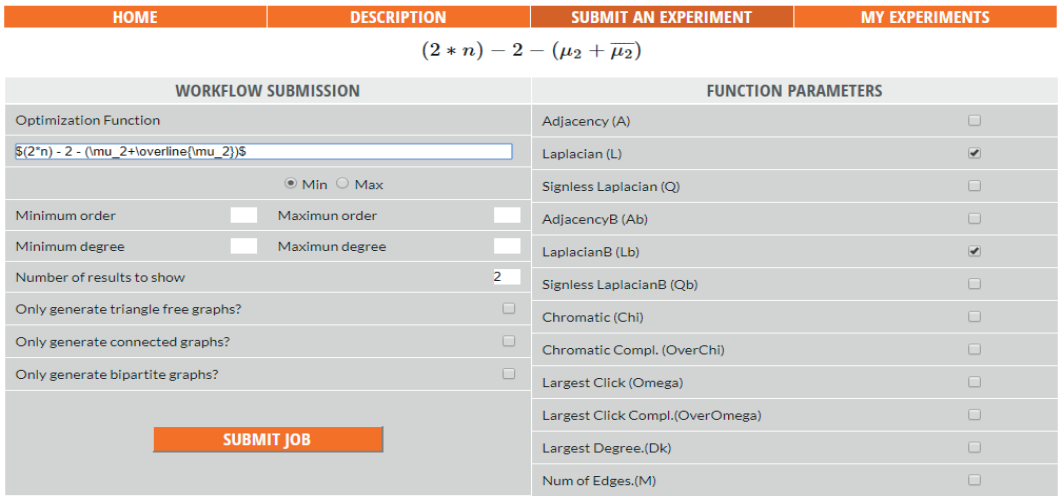


Figure 2. Dynamic form for job submission

runs a remote procedure to generate the corresponding graphs and, for each one of them, the optimization function is evaluated according to the steps of Figure 1 represented in the boxes between the vertical dashed lines. In step **B**, all simple graphs of order n in the range $n_{min} \leq n \leq n_{max}$ are generated. It is the object creation phase and one of the costliest steps, in which RioGraphX make use of distributed and parallel processing on Spark. In order to save the obtained results, the system maintains a cache of all simple connected graphs of orders ranging from 4 to 10 in the HDFS database. Table 4 presents the number of graphs according to its order. In the worst case, all of those graphs should be stored in this cache. For each graph, the following information is stored: (i) a string in g6 format

Table 4. Number of connected graphs and graphs with orders from 4 to 10

Order	4	5	6	7	8	9	10
All graphs	11	34	156	1,044	12,346	274,668	12,005,168
Connected Graphs	6	21	112	853	11,117	261,080	11,716,571
% of connected graphs	54,5	61,8	71,8	81,7	90,0	95,0	97,6

representing the graph; (ii) the number of vertices; (iii) minimum degree; (iv) maximum degree; (v) an attribute indicating whether the graph is triangle free or not; (vi) an attribute indicating whether the graph is connected or not; (vii) an attribute indicating whether the graph is bipartite or not. This data is loaded into the Spark processing stream through the SparkSQL module [Armbrust et al., 2015]. In step **C**, also in parallel and distributed mode between the Spark nodes, the computation of each invariant of the optimization function is

done by using the NetworkX library. In step **D**, the optimization function is calculated and evaluated for each generated graph. Step **E** sorts the set of generated graphs, according to the corresponding optimization function value obtained in the previous step and, in step **F**, the k best graphs are selected. Finally, step **G** produces a summary report in a PDF with information about each graph obtained during the search process, with their bitmap images, which correspond to the parameters informed in step **A**.

3. Experimental Evaluation

The proposed architecture for testing is composed of the main node, responsible for storing the Tomcat service (which provides the WEB interface), a PostgreSQL database, and the Spark Master environment (responsible for administering the worker nodes which perform the processing in a distributed and parallel fashion). All nodes are deployed using dockers with the Alpine Linux distribution as the operating system. The master node has 20GB of RAM, and each Worker has 20GB of RAM and six processing cores. A Python algorithm implementing the workflow until step F was done, and speedup tests were performed in Spark. The speedup tests measured the average time of the system to give the optimal solution for graphs ranging from 5 to 10 vertices. Our example considered the inequality $\mu_2(G) + \mu_2(\overline{G}) \leq 2n - 2$, where $\overline{G}$ is the graph complement of G. In order to test whether this inequality is true, we have used the function $f(\mu_2, \overline{\mu_2}, n) = n - 2 - (\mu_2(G) + \mu_2(\overline{G}))$ and required only connected graphs. After minimization, if RioGraphX returns a graph such that $f(\mu_2, \overline{\mu_2}, n) \leq 0$, we have a counterexample, and so the conjecture will be disproved. On the other hand, if $f(\mu_2, \overline{\mu_2}, n) \geq 0$ for all graphs we have a stronger indication that conjecture might be true. Also, we can take the graphs where $f(\mu_2, \overline{\mu_2}, n) = 0$ and extend the conjecture presenting the extremal graphs. After running RioGraphX for all graphs ranging from 5 to 10 vertices, no counterexamples were found and the obtained extremal graphs motivated the statement of Conjecture 5 in [Grijó et al., 2019]. Table 5 shows the results of the tests with different numbers of workers (nodes) where we can see that the proposed conjecture was confirmed in times considered optimal given the size of the dataset used (see Table 4), a large number of calculations performed and the ordering of the results. We verified that the time gain is linear between the results of one and two nodes. Linearity is lost as nodes were added but execution times remain significant.

Table 5. Speedup tests for graphs with 5 to 10 vertices

Nodes	Average execution time (seconds)	Speedup
1 Node (6 cores)	1817,178 ± 0.316	1,00
2 Nodes (12 cores)	943,937 ± 5,867	1,92
4 Nodes (24 cores)	663,982 ± 41,966	2,73
8 Nodes (48 cores)	472,471 ± 16,541	3,84

4. Conclusion

We proposed RioGraphX, a science gateway to aid SGT researchers in the investigation of accurate and detailed results of graphs and their properties that meet a given function. The corresponding workflow comprises seven well-defined steps that are executed in parallel and distributed inside Spark with the integration of other important tools.

Based on speedup tests, the power of the Spark applied to the proposed workflow already demonstrates that the RioGraphX science gateway is in the path to achieve its goal in becoming an important tool for SGT studies. There are several alternatives for future work. First, we plan to define a more efficient search algorithm, with the possible use of some metaheuristic, in order to find extremal graphs that minimize/maximize the user-provided combinatorial optimization function. We also plan to investigate how to maximize parallel and distributed processing with the GraphFrames libraries and evaluate their performance. Another goal is to evaluate performance and quality in the calculation of invariants in larger graphs (with more than ten vertices). Moreover, we intend to investigate speedup behavior with the use of more than eight nodes to optimize the execution time.

References

Armbrust et al. (2015). Spark sql: Relational data processing in spark. ACM.

Bomze, I. M., Budinich, M., Pardalos, P. M., and Pelillo, M. (1999). The maximum clique problem. In Handbook of combinatorial optimization, pages 1–74. Springer.

Brankov, V., Cvetković, D., Simić, S., and Stevanović, D. (2006). Simultaneous editing and multilabelling of graphs in system newgraph.

Caporossi, G. and Hansen, P. (2000). Variable neighborhood search for extremal graphs: 1 the autographix system. Discrete Mathematics, 212(1-2):29–44.

Cvetkovic, D., Rowlinson, P., and Simic, S. K. (2007). Eigenvalue bounds for the signless laplacian. Publications de l'Institut Mathématique, 81(95):11–27.

Cvetković, D. and Simić, S. (2011). Graph spectra in computer science. Linear Algebra and its Applications, 434(6):1545–1562.

Cvetković, D. M. (1971). Graphs and their spectra. Publikacije Elektrotehničkog fakulteta. Serija Matematika i fizika, (354/356):1–50.

Dave et al. (2016). Graphframes: an integrated api for mixing graph and relational queries. In Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, page 2. ACM.

Ghebleh, M., Kanso, A., and Stevanović, D. (2019). Graph6java: A researcher–friendly java framework for testing conjectures in chemical graph theory. MATCH.

Gregor, D. and Lumsdaine, A. (2005). The parallel bgl: A generic library for distributed graph computations. Parallel Object-Oriented Scientific Computing (POOSC), 2:1–18.

Grijó, R. et al. (2019). Nordhaus–gaddum type inequalities for the two largest laplacian eigenvalues. Discrete Applied Mathematics, 267:176–183.

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. Technical report, LANL.

Mohar, B., Alavi, Y., Chartrand, G., and Oellermann, O. (1991). The laplacian spectrum of graphs. Graph theory, combinatorics, and applications, 2(871-898):12.

Vasilyev, A. and Stevanović, D. (2014). Mathchem: a python package for calculating topological indices. MATCH Commun. Math. Comput. Chem, 71:657–680.

Wilf, H. S. (1967). The eigenvalues of a graph and its chromatic number. Journal of the London mathematical Society, 1(1):330–332.

Xin, R. S., Gonzalez, J. E., Franklin, M. J., and Stoica, I. (2013). Graphx: A resilient distributed graph system on spark. ACM.

Zaharia et al. (2016). Apache spark: A unified engine for big data processing. Commun. ACM, 59(11):56–65.

Análise de Hiperparâmetros em Aplicações de Aprendizado Profundo por meio de Dados de Proveniência *

Débora B. Pina[1] **, Liliane Neves**[1]**, Aline Paes** [2]**, Daniel de Oliveira** [2]**, Marta Mattoso**[1]

[1]COPPE – Universidade Federal do Rio de Janeiro (UFRJ)

[2]Instituto de Computação - Universidade Federal Fluminense (IC/UFF)

{dbpina, lneves, marta}@cos.ufrj.br, {alinepaes, danielcmo}@ic.uff.br

Resumo. *O treinamento das Redes Neurais Convolucionais (CNN) requer o ajuste de hiperparâmetros. As soluções existentes para auxiliar a escolha das melhores combinações de hiperparâmetros definem uma representação própria para modelar os relacionamentos de derivação dos dados. Essa representação proprietária dificulta a análise de dados e a interoperabilidade. Este artigo propõe a CNNProv, que adota o padrão W3C PROV para representar relacionamentos de derivação de dados para facilitar a análise das combinações de hiperparâmetros, contribuindo assim para a fase de treinamento das CNNs. A CNNProv captura dados de proveniência e permite a análise de valores de hiperparâmetros durante a execução. Os experimentos mostram a adequação do W3C PROV para a análise de hiperparâmetros e contribui para a qualidade e confiabilidade dos resultados da CNN, com overhead desprezível de até, no máximo, 4%.*

Abstract. *Convolutional Neural Networks (CNN) training requires adjusting hyperparameters. Current solutions to help choosing the best hyperparameter configuration define their own representation to model the data derivation relationships. This proprietary representation makes data analysis and interoperability difficult. This paper proposes CNNProv, which adopts the W3C PROV standard to represent data derivation relationships to facilitate the analysis of hyperparameter configurations, thus contributing to the CNNs training phase. CNNProv captures provenance data and allows hyperparameter analysis at runtime. The experiments show the suitability of the W3C PROV for hyperparameter analysis, while contributing to the quality and reliability of CNN results, with negligible overhead of up to 4%.*

1. Introdução

Na última década, a comunidade científica presenciou um aumento explosivo no número de aplicações de Aprendizado de Máquina (AM) que se baseiam em técnicas de Aprendizado Profundo (AP - *Deep Learning*) [Goodfellow et al. 2016]. Esse aumento se deve principalmente aos avanços em *hardware*, em especial às unidades de processamento gráfico (GPUs), que se mostraram adequadas para resolver de forma eficiente as operações matriciais necessárias ao AP. Uma das principais abordagens utilizadas em AP são as Redes Neurais Convolucionais (CNN) [Lecun et al. 1998]. Uma CNN é uma classe de Rede Neural do tipo *feed-forward* contendo uma ou mais unidades de convolução que utilizam filtros para produzir um mapa de atributos. Os neurônios em uma mesma camada de uma CNN podem ser agrupados em mapas e compartilham pesos, reduzindo a quantidade de parâmetros a serem treinados. Embora as CNNs representem um grande

*Agradecemos ao CNPq, CAPES e FAPERJ por financiarem parcialmente a pesquisa.

avanço na área de AM, para alcançar uma configuração de parâmetros que produza bons resultados, é necessário selecionar valores de certas variáveis, chamadas de hiperparâmetros [Goodfellow et al. 2016], o que pode demandar tempo e custo computacional. Exemplos de hiperparâmetros são a quantidade de iterações de treinamento (*épocas*) e a quantidade de exemplos apresentados em conjunto para o treinamento (*mini-batch*). O desempenho da CNN treinada depende diretamente dos valores de hiperparâmetros definidos.

De forma a definir os hiperparâmetros, o especialista em AM deve ser capaz de explorar diversas configurações para escolher uma combinação que resulte em um bom desempenho da CNN. Entretanto, esse processo de exploração requer que o treinamento da CNN seja realizado para cada uma das combinações, e, além disso, que as combinações de parâmetros utilizadas sejam registradas. Esse processo exploratório não é simples, uma vez que o espaço de busca da melhor solução é grande e a avaliação de cada configuração de hiperparâmetros pode ser computacionalmente intensiva. Abordagens existentes para AM, como o AutoML [He et al. 2018] e o GridSearch [Bergstra and Bengio 2012], já experimentam diferentes combinações de hiperparâmetros. Porém, tais métodos utilizam uma representação própria do histórico de derivação dos dados, ou seja, quais valores de hiperparâmetros levaram a quais resultados. Essas representações próprias dos dados de derivação podem dificultar a análise e, principalmente, a interoperabilidade. Assim, os resultados de experimentos em ferramentas diferentes vão requerer análises e implementações adicionais para serem comparados, já que não há uma maneira uniforme para determinar se dois modelos foram treinados com os mesmos dados de entrada.

Uma alternativa para esse problema é utilizar recomendações como o W3C PROV [Moreau and Groth 2013], que contém informações básicas para apoiar a interoperabilidade de dados de proveniência em diversos domínios. As consultas para avaliação de hiperparâmetros consumidos durante o treinamento de uma CNN se assemelham às típicas consultas de proveniência encontradas em diversos sistemas [Freire et al. 2008]. Ao definir o caminho de derivação, os dados de proveniência podem desempenhar um papel fundamental na análise e escolha dos hiperparâmetros para o treinamento. Existem diversas abordagens de captura de dados de proveniência [Stamatogiannakis et al. 2016]. Abordagens de captura automática de dados possuem granularidade muito fina, gerando um *overhead* significativo na execução de scripts, principalmente os de larga escala. A abordagem de captura de dados de proveniência por instrumentação permite a pré-seleção dos dados relevantes para análise, tendo menos impacto no tempo de execução. A DfAnalyzer [Silva et al. 2018] foi utilizada em diversos scripts de aplicações científicas com sucesso na análise de dados durante a execução do script, facilitando inclusive adaptações na configuração de parâmetros e por isso foi escolhida para esse artigo. Ademais, a captura de dados da DfAnalyzer é executada de forma assíncrona, ao longo do treinamento da CNN, e não interfere no desempenho do treinamento. Desta forma, ela é capaz de registrar informações complementares quanto ao desempenho do treinamento, além de associá-las a configurações de hiperparâmetros. A DfAnalyzer não é dependente de linguagens de programação e pode ser invocada por meio de uma biblioteca, da mesma forma que os especialistas já invocam outros componentes no código (p. ex., bibliotecas de visualização). Assim, a DfAnalyzer pode ser conectada às CNNs independente da ferramenta escolhida. Além disso, a DfAnalyzer adota o W3C PROV para representação de proveniência de dados, o que facilita a análise de dados e sua reprodutibilidade. A geração de dados de proveniência possui diversas vantagens [Freire et al. 2008], em particular a confiabilidade da CNN treinada.

Este artigo apresenta a abordagem CNNProv, que acopla a DfAnalyzer com CNNs para capturar e analisar valores de hiperparâmetros durante as fases de treinamento e execução

do modelo de forma automática Para avaliar a abordagem utilizamos a CNN AlexNet [Krizhevsky et al. 2012] como estudo de caso e o TensorFlow [Abadi et al. 2016] para o treinamento da CNN. Os experimentos realizados mostram a adequação da representação genérica da W3C PROV para a análise de hiperparâmetros, ao mesmo tempo em que contribui para a qualidade e confiabilidade dos resultados com *overhead* desprezível. Este artigo está organizado como a seguir. A Seção 2 discute os trabalhos relacionados a Seção 3 apresenta a abordagem CNNProv, a Seção 4 os experimentos, e a Seção 5 conclui.

2. Trabalhos Relacionados

A captura de dados de proveniência em experimentos de AM já foi proposta anteriormente na literatura. [Schelter et al. 2017] propõem uma abordagem para rastrear automaticamente os parâmetros de experimentos de AM, incluindo sua extração, armazenamento e gerência. De forma similar a DfAnalyzer [Silva et al. 2018], a abordagem proposta por [Schelter et al. 2017] captura os dados de forma automática com a premissa de que o usuário tenha definido quais dados carregar *a priori*. Além disso, a abordagem proposta atua de forma síncrona ao treinamento do modelo, o que pode introduzir um *overhead* não desprezível. Ademais, tal abordagem é uma representação própria da derivação dos dados, não seguindo recomendações padrão como o W3C PROV, o que complica a interoperabilidade.

DL-Steer [Souza et al. 2018] é uma biblioteca voltada para AP que coleta valores de entrada dos hiperparâmetros do modelo e relaciona-os aos resultados dos modelo treinados, permitindo que usuários alterem os valores dos hiperparâmetros durante o treinamento. Porém, a DL-Steer é limitada apenas para *scripts* Python com chamadas de funções para o TensorFlow, além de não considerar os dados de domínio da aplicação durante o processo de extração. Já o OpenML [Vanschoren et al. 2014] oferece serviços de compartilhamento de conjuntos de dados, implementações e resultados, proporcionando a reprodutibilidade de experimentos de AP. No entanto, o OpenML não representa o fluxo de transformações e não oferece apoio a consultas sobre os metadados.

3. Abordagem Proposta: CNNProv

A CNNProv visa a capturar e analisar dados de proveniência de CNNs. Esta é uma abordagem agnóstica à linguagem de programação ou bibliotecas como TensorFlow e Theano. A CNNProv pode ser descrita como uma extensão à DfAnalyzer com o objetivo de simplificar a adaptação do código da CNN para que o usuário defina quais hiperparâmetros deseja monitorar e consultar. Assim, a estratégia escolhida assume que os programas são caixas cinzas, ou seja, parte do seu código fonte é passível de ser adaptado, enquanto que a outra parte pode invocar um código fonte privado (caixa preta), para, p. ex., realizar invocações a outros programas ou bibliotecas. A abordagem é composta de cinco componentes, de acordo com a Figura 1: (i) Extrator de Proveniência (EP); (ii) Extrator de Dados Brutos (EDB); (iii) Visualizador do Dataflow (DVis); (iv) Interface de Consulta (IC); e (v) Banco de Dados estendido para modelos de CNN (MonetDB). Os dois primeiros componentes são invocados ao adaptarmos o código da CNN, enquanto os outros três têm interfaces independentes para o usuário consultar e analisar dados em tempo de execução.

A primeira etapa do uso da CNNProv é a etapa de adaptação do código que modela o treinamento. Na CNNProv, o código da CNN já vem previamente adaptado, porém os usuários podem considerar a inclusão de novos hiperparâmetros. A adaptação é dividida em duas subetapas. Na primeira, o usuário define as transformações de dados a serem rastreadas e quais hiperparâmetros serão considerados. Na segunda subetapa, o usuário indica no código onde obter os valores dos hiperparâmetros, para serem capturados em tempo de execução.

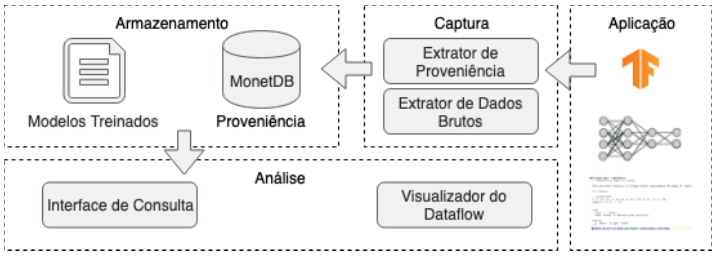


Figura 1. Arquitetura da CNNProv

Uma vez adaptado o código da CNN e seu treinamento tenha sido iniciado, o EP e o EDB são invocados assincronamente. Para cada chamada da CNNProv no código, dados das transformações, dos hiperparâmetros e dos modelos treinados são armazenados no banco de dados de proveniência usando o sistema colunar MonetDB e seguindo a representação Prov-Df [Silva et al. 2018]. Esses dados incluem o histórico de derivação dos dados, erros que tenham ocorrido, e a associação entre os valores de hiperparâmetros e os arquivos de modelo treinados. Finalmente, a interface de consulta e o visualizador do *dataflow* permitem ao usuário analisar os dados de proveniência coletados seja por meio da submissão de uma consulta ao CNNProv ou por meio de visualização.

4. Avaliação Experimental

Utilizamos a CNN AlexNet [Krizhevsky et al. 2012] como estudo de caso da CNNProv. A AlexNet é uma CNN que tem como objetivo o reconhecimento de imagens. Ela é composta de cinco camadas de convolução e três camadas completamente conectadas (Figura 3(a)). A redução na quantidade de parâmetros e no tempo de treinamento é alcançada ao conectar os filtros da 2^a, 4^a e 5^a camadas convolucionais apenas aos mapas de filtro das camadas anteriores que estão na mesma GPU – os filtros da terceira camada são conectados a todos os mapas de filtros da 2^a camada. Para reduzir o problema da dissipação do gradiente, após cada camada convolucional aplica-se a função de ativação *Relu*. Para reduzir *overfitting*, antes da 1^a e 2^a camadas completamente conectadas é incluída uma operação de *dropout* [Krizhevsky et al. 2012].

Durante a fase de adaptação do código da AlexNet para capturar dados de proveniência, foram definidas três transformações: *training*, *adaptation* e *testing*. A transformação *training* consome o nome do *dataset* de imagens de entrada (OxfordFlower17) e a proporção utilizada de dados para treino e teste, e produz um conjunto de dados que define os hiperparâmetros gerados durante a etapa de treinamento, p. ex., o valor da época, da acurácia, da função de custo (*loss function*), o tempo decorrido e a data e hora do fim da execução da época. A adaptação que é realizada pela AlexNet gera uma nova taxa de aprendizado (α') ao fim da época utilizando a função *Scheduler Step Decay* que diminui significativamente o valor da taxa de aprendizado (α) a cada n épocas em um fator m. Dessa forma, a transformação *adaptation* recebe como dados de entrada o conjunto produzido pela transformação anterior: *training* e um conjunto de dados com informações como o fator m, o valor de n e a taxa de aprendizado inicial. O conjunto de dados produzido por essa transformação contém a nova taxa de aprendizado, o valor da época e a data e hora em que a adaptação ocorreu, além de uma identificação para a adaptação, conforme apresentado na Figura 2, onde $t2$ representa a tarefa de uma transformação que terá seus dados extraídos e armazenados e *oAdaptation* representa o conjunto de dados de saída da transformação *adaptation*. Por último, a transformação *testing* provê uma avaliação do modelo de acordo com o conjunto de dados de treinamento e que tem como saída os valores de acurácia e da função de custo da CNN treinada.

Tabela 1. Tempo de treinamento

	Taxa de aprendizado		
# épocas	0,0005	0,001	0,002
20	1.367,51	1.345,42	1.365,18
50	3.387,08	3.354,28	3.359,43
100	6.745,14	6.769,15	6.757,80

Tabela 2. *Overhead* da CNNProv

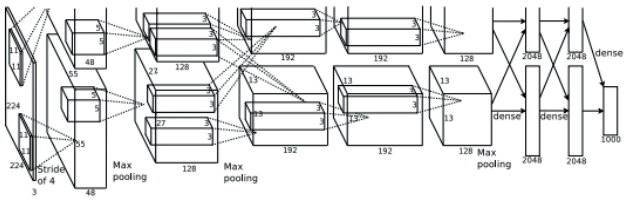
	Taxa de aprendizado		
# épocas	0,0005	0,001	0,002
20	3,57%	1,03%	3,23%
50	2,96%	2,20%	2,69%
100	3,16%	3,74%	3,33%

```
def on_epoch_begin(self, epoch, logs=None):
   old_lr = float(K.get_value(self.model.optimizer.lr))
   new_lr = self.schedule(epoch, lr)
   if (old_lr != new_lr):
      self.adaptation_id += 1
      K.set_value(self.model.optimizer.lr, new_lr)
      t2_output = DataSet("oAdaptation", [Element([new_lr,
      datetime.now().strftime('%Y-%m-%d %H:%M:%S'), epoch,
      str(self.adaptation_id)])])
      t2.add_dataset(t2_output)
      t2.save()
```

Figura 2. Trecho Adaptado do código do Tensorflow que implementa a AlexNet.

Com o código adaptado, a AlexNet foi treinada variando-se o número de épocas e a taxa de aprendizado. Cada combinação de hiperparâmetros foi executada cinco vezes e a média de tempo de execução foi considerada. A Tabela 1 apresenta o tempo de treinamento (em segundos) para cada combinação de parâmetros, enquanto que a Tabela 2 apresenta o *overhead* de tempo introduzido pela abordagem CNNProv para captura de proveniência. O objetivo desta medição foi avaliar o impacto das chamadas à CNNProv durante o treinamento da AlexNet com dados reais. Podemos observar que o *overhead* da solução proposta corresponde a um aumento de menos de 4% no pior caso sobre o tempo total de treinamento da CNN. Esse *overhead* pode ser ser considerado desprezível, principalmente em treinamentos mais demorados, considerando que o usuário terá o benefício das consultas e visualizações aos dados de proveniência capturados.

Foram levantadas e submetidas à CNNProv uma série de consultas desejadas para análise de dados de treinamento, como por exemplo: "Qual a perda, o tempo decorrido de treinamento para cada época?". A Figura 3(b) apresenta o resultado do processamento dessa consulta. A relação descrita na Figura 3(b) apresenta o tempo decorrido (*elapsed time*) em segundos, o número da época (*epoch*) e o valor da perda (*loss*). Mesmo nesse exemplo simples, podemos observar que com as adaptações no código da AlexNet foi plenamente viável consultar quanto tempo levou o treinamento em cada época e qual o valor de *loss function* associado para o treinamento.



elapsed_time	epoch	loss
106.55	20	0.50
110.92	19	0.53
113.86	18	0.66
100.06	17	0.71
109.21	15	0.73

Figura 3. (a) Arquitetura da AlexNet (b) Exemplo de consulta que mostra o tempo de corrido em cada época por ordem descrescente de perda

5. Conclusões

Este artigo apoia a fase de treinamento das CNNs ao registrar informações relevantes à análise de combinações de hiperparâmetros e posteriores reconfigurações. Foi apresentada a CNNProv, que adota uma arquitetura distribuída, com *overhead* desprezível e análise de dados via grafos de proveniência. Experimentos evidenciam a adequação do uso de proveniência nas atividades de análise e monitoramento, contribuindo para um padrão de consultas que pode consultar de forma integrada os dados de proveniência eventualmente associados aos dados usados no treinamento.

Referências

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in Science and Engineering*, 10(3):11–21.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the ECCV*, pages 784–800.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Adv. in neural inf. proc. sys.*, pages 1097–1105.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Moreau, L. and Groth, P. T. (2013). *Provenance: An Introduction to PROV*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers.

Schelter, S., Böse, J.-H., Kirschnick, J., Klein, T., and Seufert, S. (2017). Automatically tracking metadata and provenance of machine learning experiments. In *MLS workshop @ NIPS*.

Silva, V., de Oliveira, D., Mattoso, M., and Valduriez, P. (2018). Dfanalyzer: Runtime dataflow analysis of scientific applications using provenance. *PVLDB*, 11(12):2082–2085.

Souza, R., Neves, L., Azeredo, L., Luiz, R., Tady, E., Cavalin, P. R., and Mattoso, M. (2018). Towards a human-in-the-loop library for tracking hyperparameter tuning in deep learning development. In *LADaS@VLDB*.

Stamatogiannakis, M., Kazmi, H., Sharif, H., Vermeulen, R., Gehani, A., Bos, H., and Groth, P. (2016). Trade-offs in automatic provenance capture. In *IPAW*, pages 29–41. Springer.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2014). Openml: Networked science in machine learning. *SIGKDD Explor. Newsl.*, 15(2):49–60.

Processamento Eficiente de Consultas Analíticas Estendidas com Predicado de Similaridade em Spark

Guilherme Muzzi da Rocha[1], **Cristina Dutra de Aguiar Ciferri**[1]

[1]Departamento de Ciências de Computação – Universidade de São Paulo (USP)
Caixa Postal 668 – 13.560-970 – São Carlos – SP – Brasil

guilherme.muzzi.rocha@usp.br, cdac@icmc.usp.br

Abstract. *An image data warehousing extends a conventional data warehousing to also manipulate images represented by feature vectors and attributes for similarity search. A challenge that arises is the efficient processing of analytical queries extended with a similarity search predicate since these queries have a high computational cost. In this article we propose the BrOmnImg method, which efficiently solves this challenge in Spark. Compared to its closest method, BrOmnImg improved query processing up to 65.49%.*

Resumo. *Um data warehousing de imagens estende um data warehousing convencional para também manipular imagens representadas por vetores de características e atributos para pesquisa por similaridade. Um desafio que surge é o processamento de consultas analíticas estendidas com predicado de similaridade, desde que essas consultas possuem alto custo computacional. Neste artigo é proposto o método BrOmnImg, o qual soluciona eficientemente esse desafio usando o framework Spark. Comparado com o método mais próximo, BrOmnImg proveu ganhos de desempenho de até 65,49%.*

1. Introdução

Um *data warehousing* de imagens estende um *data warehousing* convencional para também manipular imagens [Teixeira et al. 2015]. O processo ETL (*extract*, *transform*, *load*) é estendido para realizar a extração das características intrínsecas das imagens, as quais são representadas por vetores de características e atributos para pesquisa por similaridade. Esses novos tipos de dados são armazenados em um banco de dados especialmente projetado para esse fim, o *data warehouse* (DW) de imagens. O processo OLAP (*on-line analytical processing*) também é expandido, de forma que ofereça suporte ao processamento de consultas analíticas estendidas com predicado de similaridade entre as imagens.

Assim, uma nova gama de consultas analíticas pode ser realizada, enriquecendo a tomada de decisão estratégica. Por exemplo, em uma organização médica voltada à análise de imagens de exames, pode-se determinar quantas imagens de câncer pulmonar são similares a uma dada imagem, considerando pacientes maiores do que 40 anos no estado de São Paulo nos últimos 3 anos [Teixeira et al. 2015]. No contexto da agricultura, medidas agrícolas referentes ao solo, fertilidade e umidade ao longo dos anos representam tipos de dados convencionais e os vetores de características de imagens armazenadas nesse contexto representam tipos de dados estendidos [Nguyen et al. 2017]. Na tomada de decisão, pode-se investigar quantas imagens são similaridades a uma determinada imagem do solo, considerando valores específicos para as medidas agrícolas.

Um desafio que surge é o processamento de consultas analíticas estendidas com predicado de similaridade. Essas consultas têm alto custo porque requerem o processamento conjunto de caras operações de junção-estrela, comuns em DWs armazenados segundo o esquema-estrela, com onerosas operações de cálculos de distância entre as imagens. Outro fato que acentua esse desafio é o gigantesco volume de dados manipulados. DWs convencionais são usualmente muito volumosos. Ademais, DWs de imagens armazenam um volume cada vez maior de imagens. Por exemplo, na área médica, diferentes hospitais, clínicas de saúde e laboratórios produzem várias imagens de exames por dia, as quais são compartilhadas visando a tomada de decisão mais robusta [Sebaa et al. 2018]. Na área agrícola, o uso de sensores possibilita a coleta massiva de dados, inclusive imagens, gerando informações valiosas no campo do agronegócio [Nguyen et al. 2017].

Neste artigo, é proposto o método *BrOmnImg*, o qual introduz uma solução eficiente para o desafio descrito. Para processar consultas analíticas estendidas com predicado de similaridade, *BrOmnImg* introduz as seguintes características. Ele integra as técnicas *broadcast join* [Brito et al. 2016] e *Omni* [Traina et al. 2007], as quais otimizam, separadamente, operações de junção-estrela sobre DWs convencionais e cálculo de distância em pesquisas por similaridade. Adicionalmente, para lidar com o gigantesco volume de dados, *BrOmnImg* é desenvolvido considerando o *framework* de processamento paralelo e distribuído Spark [Zaharia et al. 2010].

O artigo está estruturado da seguinte forma. Trabalhos correlatos são descritos na seção 2. Conceitos são resumidos na seção 3. *BrOmnImg* e os testes de desempenho são descritos nas seções 4 e 5, respectivamente. Conclusões são feitas na seção 6.

2. Trabalhos Correlatos

Em [Teixeira et al. 2015] é proposto o processamento de consultas analíticas estendidas com predicado de similaridade. Esse trabalho usa a técnica Omni, porém não considera o processamento dessas consultas em ambientes computacionais paralelos e distribuídos. Em [Brito et al. 2016] é proposto o método SBJ que usa a técnica de *broadcast join* para processar a junção-estrela em Spark. SBJ, descrito na seção 3, provê melhor desempenho do que seus concorrentes, sendo considerado o método mais próximo ao *BrOmnImg*.

Operações de cálculos de distância em Spark são otimizadas em [Li et al. 2017] usando-se funções *hash* projetadas para diminuir a colisão de imagens similares. Em [Nguyen and Huh 2017] é usado o método de acesso métrico VP-tree para esse fim, considerando o *framework* MapReduce [Dean and Ghemawat 2008]. Diferentemente de *BrOmnImg*, esses trabalhos não consideram operações de junção-estrela. Outras limitações referem-se à complexidade de definição de funções *hash* apropriadas e ao fato de que a Omni provê melhor desempenho do que a VP-tree [Traina et al. 2007].

No melhor do conhecimento dos autores deste artigo, nenhum dos trabalhos correlatos considera conjuntamente todos os aspectos que o método proposto considera, ou seja, o processamento da junção-estrela e similaridade sobre DWs de imagens em Spark.

3. Fundamentação Teórica

Em DWs relacionais, os dados são armazenados segundo um esquema-estrela composto de uma tabela de fatos que se relaciona com várias tabelas de dimensão. Isso requer a

realização de caras operações de junção-estrela no processamento das consultas OLAP, nas quais são realizadas junções entre a tabela de fatos e cada uma das tabelas de dimensão envolvidas na consulta, bem como resolvidas condições de seleção e agrupamento.

Além das tabelas de dimensão convencionais, um DW de imagens também contém outras que armazenam as características intrínsecas das imagens: uma tabela *Vetor Características* e várias tabelas *Camada Perceptual* [Teixeira et al. 2015]. A tabela *Vetor Características* contém, para cada camada perceptual de cada imagem, um vetor de características gerado por um extrator de características. Por exemplo, podem ser extraídas características referentes às camadas perceptuais de cor, textura ou forma. Os vetores de características contêm representações numéricas das imagens, e são usualmente representados no espaço métrico. Na pesquisa por similaridade, uma função de distância é usada para medir a dissimilaridade entre duas imagens por meio de seus vetores de características, de forma que a função de distância torna-se menor à medida que as imagens são mais similares. Nesse sentido, cada tabela *Camada Perceptual i* contém as distâncias entre cada imagem armazenada e cada elemento representativo da camada perceptual *i*. Esses elementos representativos são imagens estrategicamente posicionadas no espaço métrico visando a redução na quantidade das onerosas operações de cálculo de distância realizadas na pesquisa por similaridade, e são gerados pela técnica Omni [Traina et al. 2007].

Em ambientes computacionais paralelos e distribuídos, a técnica de *broadcast join* chamada SBJ [Brito et al. 2016] resolve a junção-estrela sobre DWs convencionais da seguinte forma. Ela assume que as tabelas de dimensão são suficientemente pequenas para serem enviadas para todos os nós do *cluster* durante o processamento da consulta OLAP, e realiza todas as junções, em paralelo, localmente em cada nó.

4. O Método *BrOmnImg*

Nesta seção é proposto o método *BrOmnImg* (acrônimo para ***Br**oadcast **Omni** for processing analytical **Imag**e queries*) para processar, em Spark, consultas analíticas estendidas com predicado de similaridade sobre um DW de imagens conforme definido na seção 3.

As consultas possuem dois tipos de predicado: convencional e de similaridade. O predicado convencional é composto por condições de seleção, sendo cada condição definida sobre um atributo armazenado em uma tabela de dimensão convencional. O predicado de similaridade é composto por uma operação de similaridade e pelas camadas perceptuais consideradas. Por exemplo, na consulta "Liste a quantidade de imagens similares a uma dada imagem, para pacientes do sexo feminino diagnosticadas com câncer de mama e as camadas perceptuais de cor e textura", tem-se: (i) pacientes do sexo feminino e câncer de mama como condições de seleção; (ii) o uso da operação de *range query* para resolver a similaridade; e (iii) as camadas perceptuais de cor e textura.

BrOmnImg integra as técnicas *broadcast join* e *Omni* da seguinte forma (Figura 1). As condições de seleção são aplicadas sobre as tabelas de dimensão convencionais relacionadas (Figura 1a). Para cada tabela k, os dados filtrados são armazenados em uma estrutura $HashMapConvencional_k$. Com relação ao predicado de similaridade, primeiramente a operação de similaridade é aplicada sobre todas as tabelas $CamadaPerceptual_i$ consideradas (Figura 1b). Para cada camada perceptual i, seus dados são filtrados utilizando a técnica Omni, gerando resultados candidatos armazenados nas estruturas $HashMapFiltragem_i$, as quais são transmitidas para todos os nós. Na

sequência, são realizadas as operações de cálculo de distância usando os resultados candidatos e a tabela *Vetor Características* para eliminar falsos positivos (Figura 1c). Os resultados gerados são aqueles que atendem ao predicado de similaridade, e são armazenados na estrutura $HashMapRefinamento$. Por fim, as estruturas $HashMapConvencional_k$ e a estrutura $HashMapRefinamento$ são transmitidas para todos os nós do *cluster* para a realização da junção-estrela estendida sobre a tabela de fatos (Figura 1d).

Spark é baseado em computação em memória e na abstração de RDDs (*resilient distributed dataset*) [Zaharia et al. 2010]. *BrOmnImg* armazena cada tabela do esquema-estrela como um RDD. Ele aplica os filtros convencionais e de imagem sobre os RDDs correspondentes usando a operação *filter*, e armazena os dados nas estruturas *hash map* por meio das operações *mapToPair* e *collect*. Adicionalmente, *BrOmnImg* realiza a junção-estrela estendida aplicando as operações *filter*, *mapToPair* e *reduceByKey*.

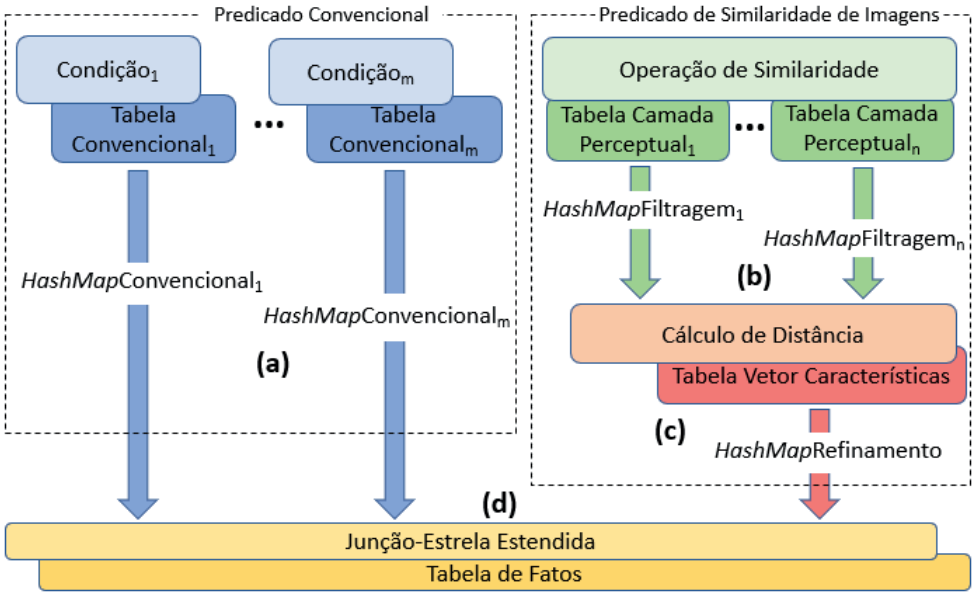


Figura 1. Visão geral do método proposto.

5. Testes de Desempenho

Dados. O DW de imagens da área médica foi povoado com dados gerados pela ferramenta *ImgDW Generator* [Rocha and Ciferri 2018]. Foram projetadas 2 tabelas de dimensão convencional: *Pacientes*, com 2 milhões de tuplas, e *Descrição Exame*, com 20 milhões de tuplas. Foram projetadas 2 tabelas *Camada Perceptual*: *Histograma Cores* e *Haralick Variância*, cada qual contendo as distâncias entre cada imagem e 3 elementos representativos gerados pela Omni. Foram inseridas 20 milhões de tuplas na tabela de fatos *Exame*, em *Histograma Cores*, em *Haralick Variância* e na tabela *Vetor Características*.

Consultas. As consultas analíticas estendidas com predicado de similaridade foram definidas considerando até 3 condições de seleção do predicado convencional (sexo *feminino*, diagnóstico de *câncer* e parte do corpo sendo *mama*) e 1 predicado de similaridade especificado pela operação *range query* [Traina et al. 2007] com um raio de abrangência de 20% do diâmetro do conjunto de dados, considerando cada camada perceptual envolvida.

Configurações. As configurações consideraram a seletividade do predicado convencional e a dimensionalidade das camadas perceptuais. Para o predicado convencional, foram definidas consultas: (i) sem predicado - *SemConv*; (ii) com predicado de baixa seletividade (33,33%) envolvendo os pacientes do sexo feminino - *BaixaSel*; e (iii) com predicado de alta seletividade (0,08%) envolvendo os pacientes do sexo feminino com câncer de mama - *AltaSel*. Para o predicado de similaridade, foram definidas consultas sobre: (i) a camada perceptual de alta dimensionalidade *Histograma Cores* (256 dimensões) - *His*; a camada perceptual de baixa dimensionalidade *Haralick Variância* (4 dimensões) - *HV*; e (iii) essas essas duas camadas conjuntamente - *His/HV*. No total, foram geradas 9 configurações, conforme mostrado na Tabela 1. Para cada configuração, o número de imagens que atenderam aos predicados convencional e de similaridade foram: *C1*: 224.060; *C2*: 125.460; *C3*: 5.620; *C4*: 74.800; *C5*: 42.000; *C6*: 1.800; *C7*: 160; *C8*: 180; *C9*: 20.

Tabela 1. Configurações definidas para os testes de desempenho

	His	**HV**	**His/HV**
SemConv	(*C1*) SemConvHis	(*C2*) SemConvHV	(*C3*) SemConvHis/HV
BaixaSel	(*C4*) BaixaSelHis	(*C5*) BaixaSelHV	(*C6*) BaixaSelHis/HV
AltaSel	(*C7*) AltaSelHis	(*C8*) AltaSelHV	(*C9*) AltaSelHis/HV

Execução. Foi usado um *cluster* com 5 nós, cada qual com, no mínimo, 3GB de RAM. *BrOmnImg* foi comparado com SBJ, que representa o trabalho correlato mais próximo da literatura. Cada consulta foi executada 5 vezes, sendo obtida a média das execuções.

Resultados. Na Figura 2 são ilustrados os resultados obtidos, bem como o desvio padrão. Para as configurações que possuem, pelo menos, uma camada perceptual de alta dimensionalidade (*C1*, *C3*, *C4*, *C6*, *C7*, *C9*), *BrOmnImg* proveu ganhos de desempenho que variaram de 60,01% a 65,49% quando comparado com SBJ. Isso está relacionado à complexidade dos cálculos de distância da técnica Omni, sendo dependente da dimensionalidade do conjunto de dados e do número de elementos representativos. A diferença entre o número de dimensões de *Histograma Cores* (256) e o número de elementos representativos correspondente (3) impactou positivamente no desempenho de *BrOmnImg*. Para as configurações que possuem apenas a camada perceptual de baixa dimensionalidade (*C2*, *C5*, *C8*), *BrOmnImg* empatou com SBJ ou proveu ganhos de desempenho de até 10,50%. Nessas configurações, a diferença entre o número de dimensões de *Haralick Variância* (4) e o número de elementos representativos correspondente (3) foi muito pequena. Positivamente, mesmo *BrOmnImg* sendo mais complexo do que SBJ por integrar a técnica Omni para processar o predicado de similaridade, ou ele empata ou provê melhores resultados.

6. Conclusão

Neste artigo é proposto o método *BrOmnImg*, voltado ao processamento eficiente de consultas analíticas estendidas com predicado de similaridade sobre *data warehouses* de imagens. *BrOmnImg* mostra a viabilidade de integração das técnicas de *broadcast join* e Omni em Spark. Comparado com o método mais próximo, *BrOmnImg* garantiu melhora de desempenho expressiva para predicados de similaridade definidos sobre camadas perceptuais de alta dimensionalidade. Trabalhos futuros incluem a realização de novos testes de desempenho considerando outras consultas e diferentes operações de similaridade.

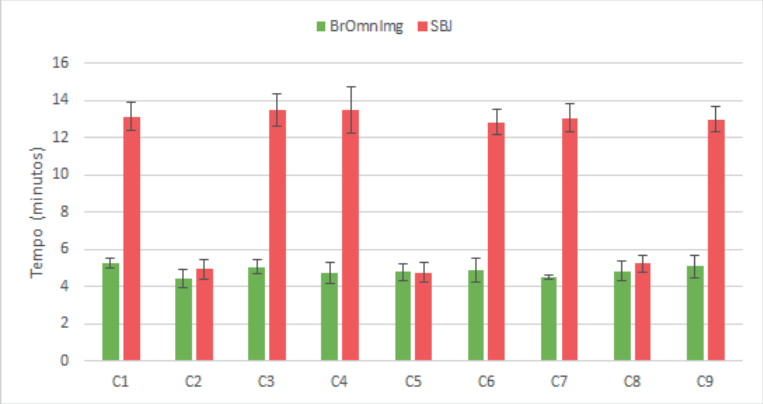


Figura 2. Testes de desempenho comparando *BrOmnImg* com SBJ.

Agradecimentos. Trabalho sendo desenvolvido com recursos financeiros da FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo), processos 2018/10607-3 e 2018/22277-8, e do CNPq.

Referências

Brito, J. J., Mosqueiro, T., Ciferri, R. R., and Ciferri, C. D. A. (2016). Faster cloud star joins with reduced disk spill and network communication. *Procedia Computer Science*, 80:74 – 85.

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Commununications of the ACM*, 51(1):107–113.

Li, D., Zhang, W., Shen, S., and Zhang, Y. (2017). SES-LSH: Shuffle-efficient locality sensitive hashing for distributed similarity search. In *ICWS 2017*, pages 822–827.

Nguyen, T. D. T. and Huh, E.-N. (2017). An efficient similar image search framework for large-scale data on cloud. In *IMCOM 2017*, pages 65:1–65:8.

Nguyen, V.-Q., Ngoc, N., and Kim, K. (2017). Design of a platform for collecting and analyzing agricultural big data. *Journal of Digital Contents Society*, 18:149–158.

Rocha, G. M. and Ciferri, C. D. A. (2018). ImgDW generator: a tool for generating data for medical image data warehouses. In *SBBD 2018 Proc. Companion*, pages 23–28.

Sebaa, A., Chikh, F., Nouicer, A., and Tari, A. (2018). Medical big data warehouse: Architecture and system design, a case study: Improving healthcare resources distribution. *Journal of Medical Systems*, 42(4):59.

Teixeira, J. W., Annibal, L. P., Felipe, J. C., Ciferri, R. R., and Ciferri, C. D. A. (2015). A similarity-based data warehousing environment for medical images. *Computers in Biology and Medicine*, 66:190 – 208.

Traina, C., Filho, R. F. S., Traina, A. J. M., Vieira, M. R., and Faloutsos, C. (2007). The Omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. *The VLDB Journal*, 16(4):483–505.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *USENIX HotCloud 2010*.

Data Warehouse Educacional: Uma visão sobre a Evasão no Ensino Superior

G.A.S.Santos[1], A.L.Bordignon[2], D.B.Haddad[1], D.N.Brandão[1], L.Tarrataca[1], K.T.Belloze[1]

[1] PPCIC - Programa de Pós-graduação em Ciências da Computação (CEFET-RJ)
Caixa Postal 20271-204 – 455, Maracanã, Rio de Janeiro - RJ

[2]Instituto de Matemática, Universidade Federal Fluminense, Niterói - RJ

gustavo.santos@eic.cefet-rj.br, {diego.brandao, kele.belloze}@cefet-rj.br

Abstract. *Dropout is one of the main challenges of educational institutions. In this sense, this paper presents the implementation of a Data Warehouse for data analysis and decision making in a higher education institution in Brazil. The presented Data Warehouse allows integrated views that assist in analysis such as: 1) distribution of students' performance coefficient; 2) identification of student profiles and 3) insight into student achievement by locality. These analyzes are intended to assist academic management in identifying patterns that lead to dropout and thus to promote directions for preventive actions and mainly to expand the use of this analytical database developing new solutions, such as predictive models.*

Resumo. *A evasão mostra-se como um dos principais desafios das instituições de ensino. Nesse sentido, este trabalho apresenta a implementação de um Data Warehouse para análise de dados e auxílio à tomada de decisão em uma instituição de ensino superior do Brasil. O Data Warehouse apresentado permite visões integradas que auxiliam em análises de: 1) distribuição do coeficiente de desempenho do alunos; 2) identificação dos perfis dos estudantes e 3) um dashboard sobre o rendimento dos alunos por localidade. Essas análises têm como propósito auxiliar a gestão acadêmica na identificação de padrões que acarretam na evasão e, desta forma, promover direcionamentos para medidas preventivas e, principalmente, expandir o uso deste banco de dados analítico para desenvolver novas soluções, como por exemplo, modelos preditivos.*

1. Introdução

A gestão acadêmica das instituições de ensino superior compreende diversas atividades. No caso das Instituições Federais de Ensino Superior (IFES) no Brasil, essas atividades consistem em ensino, pesquisa e extensão. Além disso, as IFES, em sua maioria, possuem políticas e programas socioeconômicos com o objetivo de auxiliar e dar suporte a essas atividades. É importante notar que há vários desafios para a gestão acadêmica, muitos dos quais requerem cuidadosa atenção como evasão, retenção e vagas ociosas [Speller et al. 2012].

Especificamente, a evasão é uma situação que ocorre quando os estudantes ocupam vagas e se desassociam das universidades sem concluir o curso em que se matricularam. O custo decorrente devido a evasão é muito maior do que o desejado pelo

governo [dos Santos Baggi and Lopes 2011]. Dados do Censo de Educação Superior no Brasil, em 2016, mostraram que dentre as 10,6 milhões de vagas oferecidas nos cursos de graduação, 26,0% dessas foram oriundas de evasão [INEP 2016]. Por outro lado, o relatório da Organização para a Cooperação e Desenvolvimento Econômico (OCDE) [OECD 2016] revela que o custo médio anual de um estudante do ensino público superior no Brasil, em 2013, foi de US$13.539,90.

Na busca de respostas para o problema de evasão, algumas IFES adotam soluções tecnológicas, baseadas em sistemas de apoio a tomada de decisão. Tais sistemas, dentre os quais o *Data Warehouse* assume um papel de destaque [Olszak and Ziemba 2007], contribuem com a análise de dados históricos e modelos de predição. Esses podem apoiar significativamente na compreensão dos problemas institucionais e, em particular, suas causas e consequências [Shim et al. 2002].

Este trabalho descreve a implementação de um *Data Warehouse* Educacional (EDW - *Educational Data Warehouse*), de modo a fornecer um banco de dados analítico capaz de apresentar análises integradas em uma IFES. O objetivo deste EDW é que as análises geradas apoiem a gestão acadêmica na identificação de padrões que acarretam evasão e desta forma sejam desenvolvidas medidas preventivas em relação ao tema.

Além desta introdução, este artigo está estruturado da seguinte forma: a seção 2 apresenta a implementação do EDW. A seção 3 apresenta o estudo de caso sobre a evasão juntamente com uma apresentação de gráficos gerados pelo sistema. Finalmente, a seção 4 apresenta as considerações finais e trabalhos futuros.

2. O *Data Warehousing* Educacional

Na maioria dos projetos de DW, o desafio consiste em planejar de maneira eficiente o desenvolvimento geral do sistema. A Figura 1 apresenta o fluxo de dados para o processo que incorpora o contexto informacional necessário para a construção do EDW.

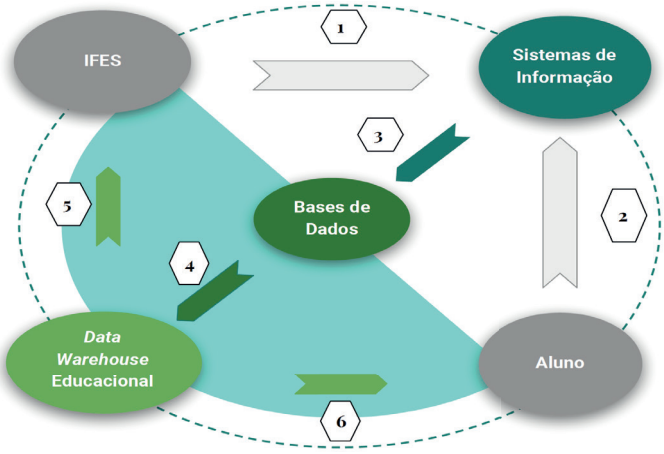


Figura 1. Fluxo de Dados para o *Data Warehouse* Educacional.

Neste processo, estão representadas as seguintes entidades: IFES, Aluno, Sistemas de Informação, Bases de Dados e *Data Warehouse* Educacional. As etapas (1) e (2) lidam, respectivamente, com as interações entre estudantes e funcionários da universidade. Isso é feito no sistema de informação (SI) da universidade por meio de atividades

com foco no gerenciamento acadêmico dos cursos e, consequentemente, nos alunos. Na etapa (3), as informações inseridas no SI são armazenadas em bases de dados. A etapa (4) representa o objetivo principal deste trabalho que será descrito nas subseções a seguir. Finalmente, as etapas (5) e (6) se relacionam com a aquisição de conhecimento que é obtida por meio do banco de dados analítico implementado pelo EDW. A etapa (5) fornece as informações analíticas que apoiam o processo de tomada de decisão. A etapa (6) fornece informações que podem ser consultadas, bem como recomendações baseadas no perfil do aluno, no curso, e nas interações que são armazenadas no EDW. O desenvolvimento do EDW, focado no tema evasão, é baseado nos requisitos funcionais apresentados na Tabela 1.

Tabela 1. Requisitos funcionais para o EDW

Item	Descrição
01	Analisar a evasão com relação ao desempenho acadêmico
02	Analisar a evasão com base nos perfis dos estudantes
03	Analisar a evasão considerando a localidade de curso

2.1. Fontes de dados: descrição dos Sistemas de Informação

O conjunto de SI escolhido para o EDW é apresentado a seguir. O sistema de identificação única é uma aplicação da IFES que visa centralizar os dados das pessoas que têm ou tiveram algum vínculo com a universidade e também todas as informações acadêmicas referentes aos cursos de graduação. O sistema de iniciação científica foi desenvolvido para administrar: (i) o processo de submissão de projetos de pesquisa; (ii) seleção de projetos e candidatos; (iii) concessão de bolsas de iniciação científica e (iv) avaliação de projetos. O sistema de bolsas foi desenvolvido com o objetivo de administrar bolsas de assistência estudantil. O sistema de monitoria foi desenvolvido com o objetivo de facilitar o processo de submissão das candidaturas para monitoria de disciplinas. Todos esses sistemas foram integrados em um banco de dados analítico por meio dos procedimentos de modelagem multidimensional: Extração, Transformação, Carregamento (*Extract, Transform, Load* - ETL) e visualização de dados.

2.2. Construção do EDW

O EDW é baseado em um modelo multidimensional, o qual permite que os dados sejam integrados e visualizados sob várias dimensões [Inmon and Linstedt 2014]. A fim de permitir que várias questões de negócio possam ser respondidas por meio da evidência dos fatos, é necessário consolidar outras perspectivas de informação, as quais são caracterizadas como dimensões. Para atender aos requisitos de negócios descritos na Tabela 1, foram criadas oito tabelas de dimensões, representando as seguintes entidades: 'Aluno', 'Histórico do Aluno', 'Bolsa', 'Bolsista', 'Curso', 'Acompanhamento do Aluno', '*Status* do Aluno' e 'Tempo'.Além disso, foram criadas duas tabelas de fatos representando as entidades 'Evasão' e 'Histórico Acadêmico'.

Para popular as dimensões e fatos, aplicou-se o processo ETL, o qual ocorreu em várias etapas. Neste trabalho, a primeira etapa é referente à extração dos dados dos sistemas acadêmicos, a qual gera arquivos no formato `.csv` (*comma-separated values*. Em seguida, é executado um procedimento que carrega esses arquivos em um banco de

dados relacional. Feito isso, os dados são ajustados e corrigidos com base nos requisitos e transformados em um banco de dados multidimensional.

3. Estudo de Caso com Foco na Evasão

Nesta seção, alguns resultados obtidos por meio da análise de dados são apresentados. O conjunto de dados produzido pelo EDW contém informações sobre os estudantes como as notas do Exame Nacional do Ensino Médio (ENEM), histórico acadêmico, registro de ação afirmativa (políticas sociais), cor da pele e dados sociodemográficos. O EDW contêm informações sobre aproximadamente 80.000 alunos dos 106 cursos de graduação que são oferecidos pela IFES. Tais dados são fornecidos pela tabela "FATO_EVASAO" do modelo EDW. Este conjunto de dados compreende alunos desde os anos de 2005 até 2018.

Na Figura 2, é possível identificar a distribuição dos alunos por localidade (diferentes *campus* da instituição), considerando o percentual referente à faixa de coeficiente de rendimento (CR) do estudante e localidade. É apresentado também o percentual de CR total da Instituição, o qual mostra que um pouco mais de 37% de alunos apresenta CR abaixo de 4,0.

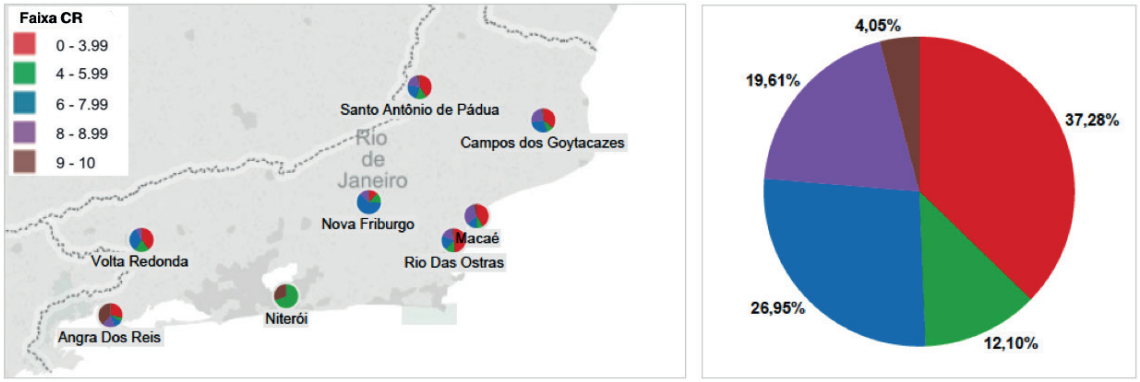


Figura 2. Distribuição do rendimento dos alunos por localidade e total.

Durante a análise exploratória dos dados, algumas perspectivas foram observadas. A primeira perspectiva foi de que o conjunto de dados continha uma representação de classes desbalanceadas, com 76% de estudantes evadidos e 24% graduados. Dessa forma, a Tabela 2 apresenta detalhes quantificando a amostra ("Qtd.") e exibindo para cada classe a mediana dos valores dos atributos: "Idade", "SemestreFinal" (último semestre do ano cursado), "CR (Coeficiente de Rendimento)", "CargaHorariaCursada"(CargaHor) e "TempoPermanencia". Com base nos resultados, é possível observar que o padrão de evasão destaca-se no primeiro semestre de cada ano letivo por alunos com idade próxima a 25 anos, CR mediano de 3,4 e com um total de carga horária cursada próximo 240 horas de currículo (3 anos).

A segunda perspectiva pode ser identificada por meio do histograma apresentado na Figura 3. Nele é possível perceber que há uma frequência maior da amostra de alunos com baixo desempenho nas regiões de CR variando de '0' até '3', sendo boa parte desta frequência de 34% constituída por alunos evadidos, em geral, ocorrendo esta evasão no primeiro ano de ingresso. Essa observação foi possível a partir das análises realizadas no EDW.

Tabela 2. Mediana dos atributos baseada nos perfis “Evadido”e “Graduado”.

Classe	Qtd.	Idade	SemestreFinal	CR	CargaHor	TempoPermanencia
Evadido	9852	25	1º	3.4	240	3 anos
Graduado	3117	24	2º	8.3	3199	5 anos

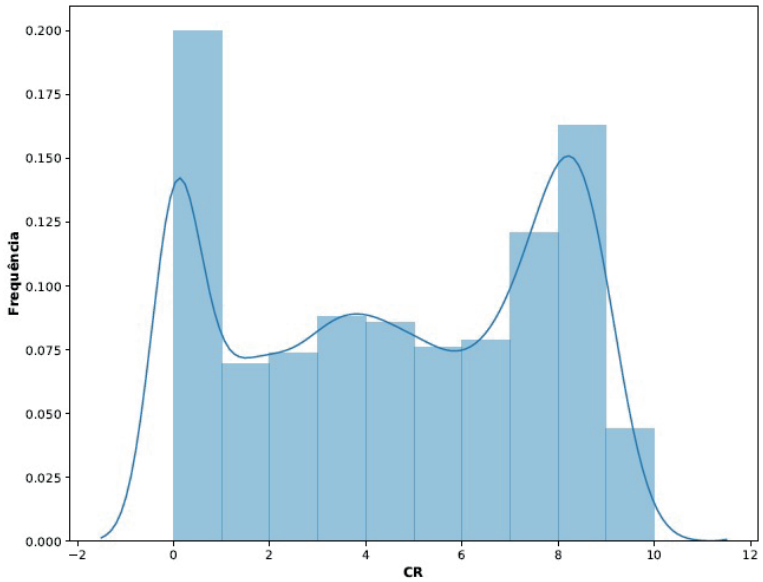


Figura 3. Histograma do CR de toda amostra, incluindo graduados e evadidos.

A partir das perspectivas anteriores, avaliou-se também a necessidade de medir a correlação entre os atributos gerados pelo conjunto de dados observado. Nesse caso, foram priorizados os atributos de ‘CR’ e ‘StatusFormacao’, a fim de avaliar a relação de influência dos demais atributos no aspecto de desempenho acadêmico e no *status* de formação do aluno (‘evadido’ ou ‘graduado’). A Tabela 3 apresenta as correlações, observa-se que as notas do ENEM tem uma correlação baixa com relação ao CR do aluno bem como se ele vai evadir ou não (‘StatusFormacao’).

4. Considerações Finais

Neste trabalho foi apresentado um *Data Warehouse* Educacional para atender a demanda de um sistema de apoio à tomada de decisão sobre uma visão integrada referente ao tema evasão. A proposta mostrou-se como solução capaz de conceder subsídios que podem auxiliar a gestão acadêmica na identificação de padrões que impactam na evasão.

Por meio das análises realizadas foi percebido que as notas do ENEM têm uma influência significativa no CR e *status* de formação do aluno, assim como a carga horária cursada e o tempo de permanência do aluno. Esses são indicadores altamente relevantes na análise de evasão. Além disso, percebeu-se a necessidade de ajustes quanto à entrada de novas informações consideradas importantes, mas que ainda não existem no sistema de gestão acadêmica. Essas informações são referentes ao histórico do ensino fundamental e médio dos alunos de graduação. Foi evidenciado que este tipo de informação implicará em uma melhor percepção quanto ao perfil do aluno de graduação, pois representa boa parte de sua formação escolar e com isso, pode ser possível efetuar uma análise preditiva no primeiro período do aluno.

Tabela 3. Correlação dos atributos “CR” e “StatusFormacao” entre os demais atributos.

Atributo	CR	StatusFormacao
EnemLinguagem	0.141993	0.099262
EnemHumanas	0.094311	0.018340
EnemCiencias	0.103915	0.040897
EnemMatematica	0.067701	0.036235
EnemRedacao	0.139197	0.095648
IdTurno	0.076495	−0.014528
IdTurnoAtual	0.008767	−0.057659
CR	1.000000	0.633797
AnoIngresso	−0.093145	−0.208442
SemestreIngresso	−0.108587	−0.101707
Idade	−0.145779	−0.067144
CargaHorCursada	0.702870	0.901757
Trancamento	0.037985	−0.008887
TempoPermanencia	0.539021	0.480889
StatusFormacao	0.633797	1.000000

Como propostas de trabalhos futuros, é factível utilizar técnicas de aprendizado de máquina para identificar o conjunto de atributos mais relevante para um aluno, e também classificar os alunos baseando-se numa estratégia de análise de risco de evasão, conforme o perfil dos estudantes. Essas propostas de trabalhos podem possibilitar uma maior efetividade na redução dos impactos da evasão e, consequentemente, uma melhor gestão de recursos.

Referências

dos Santos Baggi, C. A. and Lopes, D. A. (2011). Evasão e avaliação institucional no ensino superior: uma discussão bibliográfica. *Avaliação: Revista da Avaliação da Educação Superior*, 16(2).

INEP (2016). Censo da educação superior - notas estatísticas 2016. In *Diretoria de Estatísticas Educacionais (DEED)- Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP)*. Acessado em 13 de Outubro de 2018.

Inmon, W. H. and Linstedt, D. (2014). *Data architecture: a primer for the data scientist: big data, data warehouse and data vault*. Morgan Kaufmann.

OECD (2016). *Education at a Glance 2016*.

Olszak, C. M. and Ziemba, E. (2007). Approach to building and implementing business intelligence systems. *Interdisciplinary Journal of Information, Knowledge, and Management*, 2(1):135–148.

Shim, J. P., Warkentin, M., Courtney, J. F., Power, D. J., Sharda, R., and Carlsson, C. (2002). Past, present, and future of decision support technology. *Decision support systems*, 33(2):111–126.

Speller, P., Robl, F., and Meneghel, S. M. (2012). Desafios e perspectivas da educação superior brasileira para próxima década. *Oficina de Trabalho. p. 164, 2012. ISBN: 978-85-7652-171-6*.

Definindo e Predizendo Níveis de Saúde de Colônias de Abelhas via Clusterização e Classificação

Antonio Rafael Braga[1], Daniel A. Silva[2], Juvêncio S. Nobre[2], Breno M. Freitas[3], Danielo G. Gomes[1]

[1]Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará (UFC).

[2]Departamento de Estatística e Matemática Aplicada, UFC.

[3]Setor de Abelhas, Departamento de Zootecnia, UFC.

{danielamaral}@alu.ufc.br, {rafaelbraga,juvencio,freitas,danielo}@ufc.br

Resumo. *As abelhas são essenciais à produção de alimentos para o ser humano e para manutenção dos ecossistemas. Esse artigo apresenta uma solução para calcular os níveis de estados de saúde de colônias de abelhas usando dados de sensores internos e externos à colônia e de inspeções in loco realizadas por apicultores. A clusterização foi usada para determinar a quantidade de níveis de saúde e a classificação para criação de um modelo de predição. Obteve-se um modelo de classificação com taxa de acerto de 99.36%.*

Abstract. *Bees are essential for the production of food for humans and the maintenance of ecosystems. This paper presents a solution for calculating bee colony health status levels using data from internal and external colony sensors and on-site inspections by beekeepers. Clustering was used to determine the number of health levels and classification to create a prediction model. We obtained a classification model with an accuracy ratio of 99.36%.*

1. Introdução

As abelhas são essenciais à produção de alimentos para o ser humano e para manutenção dos ecossistemas pois constituem o grupo mais importante de polinizadores [Potts et al. 2016]. No entanto, a diversidade de polinizadores está em declínio. Em paisagens agrícolas, isso é freqüentemente observado em grandes monoculturas. Existe a preocupação de que a expansão urbana e o desmatamento também estejam impactando negativamente a diversidade de polinizadores. Reduções na diversidade e abundância de polinizadores podem afetar a reprodução de espécies de plantas, a produção agrícola, a segurança alimentar e o bem-estar humano [Potts et al. 2010].

Assim, o monitoramento via Rede de Sensores sem Fio tem sido usado mais recentemente para evitar a morte de colônia de abelhas. Associado ao monitoramento, os apicultores fazem uso também de inspeções *in-loco*. Contudo, essas inspeções são prejudiciais à saúde da colônia. Logo, torna-se relevante a predição do nível de saúde de uma colônia para notificação do apicultor (usuário final) para que o mesmo possa atuar através de manejos para restituição da saúde da colônia.

Nesse contexto, este trabalho apresenta uma solução para identificação da quantidade ótima de estados de saúde de colônias de abelhas e dos tipos de estados baseada

em dados de sensores internos e externos às colônias e de inspeção previamente realizadas. A solução utiliza a clusterização (aprendizagem não-supervisionada) e a classificação (aprendizagem supervisionada).

Na etapa de clusterização, o índice de validação *Calinski-Harabasz* (CH) [Calinski and Harabasz 1974] e o algoritmo *k-means* [MacQueen 1967] foram usados para determinar a quantidade ideal de classes de saúde das colônias. Os dados de inspeções *in loco* foram utilizados para fazer a rotulagem dos dados de sensores em relação ao estado de saúde. Após a rotulagem, foram treinados 4 algoritmos de classificação distintos, o k-vizinhos mais próximos (*k-Nearest Neighbors*, k-NN), as Florestas Aleatórias (*Random Forest*, RF), Redes Neurais (*Neural Networks*, NN) e Máquinas de Vetores Suporte (*Support Vector Machine*, SVM).

2. Materiais e Métodos

2.1. O Conjunto de Dados

As colméias utilizadas nesta pesquisa pertencem ao projeto "*Bayer Bee Care*" (https://www.agro.bayer.com.br/beecare) (BBC) e estão localizadas no estado da Carolina do Norte, EUA. Foram usadas 4 colméias, duas do apiário 'BBCC', em Durham, monitoradas no periodo de Junho de 2017 à Abril de 2019, e 2 no apiário Beesboro em Clayton, monitoradas no periodo de Junho de 2017 à Junho de 2018. Ambas possuem abelhas da espécie *Apis mellifera*. Os dados de sensores internos às colméias foram coletados através do sistema Brood Minder (https://broodminder.com/). Os dados de sensores externos foram obtidas de estações meteorológica do Serviço Nacional de Meteorologia dos EUA. Para o apiário BBCC, foi usada a estação do aeroporto internacional de Raleigh-Durham e para o apiário Beesboro a estação do aeroport Johnston County.

Os dados de inspeções foram obtidos com o auxílio do "*Healthy Colony Checklist* (HCC)", que é um documento usado para padronizar a inspeção de uma colônia de abelhas por um apicultor. As inspeções foram realizadas semanalmente. O HCC[1] utilizado neste estudo foi o proposto pelo projeto BBC. O HCC é composto por 6 fatores binários para definir o estado de saúde da colônia, são eles: 1 - Brood (Ninhada), 2 - Bees (Abelhas Adultas), 3 - Queen (Rainha), 4 - Food (Alimento), 5 - Stressors (Estressores) e 6 - Space (Espaço). Assim, se todos os itens forem marcados como "sem problema", então a colônia é considerada 100% saúdável. Por outro lado, cada item marcado como 'com problema', representa uma diminuição de 1/6 do nível de saúde da colônia. Esse nível de saúde calculado é utilizado como variável dependente, classes.

2.2. Pré-processamento

O conjunto de dados foi disponibilizado em arquivos separados por dados de sensores internos, externos e dados de inspeção. Os dados disponibilizados, foram divididos por Apiário e subdivididos por Colméia. A Tabela 1 descreve o conteúdo de cada arquivo disponibilizado.

Assim, para o merge dos arquivos foi utilizada a variável de tempo *Human_timestamp* com um limite de similaridade de 57 minutos para os dados de sensores

[1]https://beehealth.bayer.us/bayer-news-and-resources/setting-the-standard-for-managing-healthy-honey-bee-colonies

Arquivo	Variáveis
Sensores Internos	Human_timestamp, Brood_Temp, Brood_Humidity, Hive_Temp, Hive_Humidity, Scale_Temp, Scale_Humidity, Weight, Hive e Apiary
Sensores Externos	Human_timestamp, Temperature, DewPoint, Pressure, WindDirection, WindSpeed, Sky-Condition, Precipitation1Hr, Precipitation6Hr e Apiary
Inspeção	Human_timestamp, Apiary, Hive, Brood, Bees, Queen, Food, Stressors, Space e Code

Tabela 1. Conteúdo dos dados de sensores internos, externos e de inspeção

para mais ou para menos e para o merge dos dados de inspeção um limite de 1 semana para mais ou para menos. Obtivemos então, um único arquivo com 15.490 amostras contendo as variáveis de sensores internos, externos e de inspeções (rótulo da amostra). Após, removemos algumas colunas não-informativas, as causas de remoção de cada variável são descritas na Tabela 2.

Variáveis	Motivo de Remoção
Hive, Apiary	Queremos um modelo geral, portanto, um modelo que diferencia apenas 2 apiarios e suas colmeias não é o adequado
ScaleTemp, ScaleHumidity	Variáveis sem sentido, já que o sensor Scale só nos dá a informação de peso da colônia.
Pressure	Todos os valores faltantes ocorrem no Apiário "Beesboro". Portanto, a imputação não é uma boa idéia
SkyCondition, Precipitation1Hr, Precipitation6Hr	Alto índice de valores faltantes ($>$90%)

Tabela 2. Motivos de Remoção de variáveis no conjunto de dados

Após a remoção das variáveis não-informativas, notamos diferentes padrões de medição e de escalonamento. Para fins de padronização do experimento transformamos as unidades das variáveis de temperatura de $Fahrenheit$ para $Celsius$ e as variáveis de peso de $Pounds$ para $Kilo$. Após as conversões, reescalonamos as variáveis $Temperature$ e $DewPoint$. Para a consistencia dos dados definimos um limite de -10º à 50º para as variáveis de temperatura, 0 à 100 para as variáveis de humidade e 1 à 100 Kg para a variável peso. Caso a variável em estudo não se enquadre nesses limites ela será imputada pelo método *Multivariate Imputation by Chained Equations* (MICE) [Buuren and Groothuis-Oudshoorn 2010].

Como existem variáveis com medidas diferentes e com escalas e limites muito diferentes, foi realizada a padronização dos dados através da transformação z-score. Para adicionar mais informações aos modelo de predição, foram adicionadas duas variáveis independentes, são elas: Season (Estação do Ano) e TurnDay (Turno do Dia). Por fim, foi feita a conversão das variáveis nominais Season e TurnDay na abordagem Casela de Referência.

3. Algoritmos Propostos

A seguir, serão apresentados os algoritmos propostos nesse trabalho. O Algoritmo 1 nos fornece o melhor valor do número de clusters com base no indice CH, em síntese, esse algoritmo recebe vários candidatos ao número de clusters e para cada candidato o laço for (l. 2-9) agrupa o conjunto de dados de interesse pelo algoritmo k-means (l. 3-7). Para esse candidato em específico é calculado o índice CH associado (l. 8). Então, o melhor valor do número de clusters é aquele com maior valor associado ao vetor de índices CH.

O Algoritmo 2 utiliza o resultado do Algoritmo 1 para realizar uma busca exaustiva do melhor agrupamento, em síntese, esse algoritmo recebe um valor ótimo k do número de cluster, fatores de inspeção e um conjunto de dados. É inserido, então, todos

os possíveis agrupamentos em k clusters dos fatores de inspeção (l. 2). No laço for (l. 5-10) é realizada a rotulagem do conjunto de dados e o cálculo da acurácia de um classificador genérico. A rotulagem (l. 6-7) é realizada através de uma associação direta entre a quantidade de itens de inspeção saúdáveis e a saúde da colônia, ver Seção 2.1. Ao final do algoritmo, é retornada um vetor com o desempenho/acurácia de cada agrupamento de k clusters pelo modelo de classificação escolhido. A partir desses resultados pode se escolher o melhor agrupamento de fatores e o melhor algoritmo a ser utilizado.

Algoritmo 1 Algoritmo para a escolha do melhor valor de k via *k-médias* e o índice CH

Entrada: **K** (vetor com os possíveis números de clusters)
D (um conjunto de dados)
Saída: Um vetor com os índices CH associados ao vetor **K**
1 indices = {}
2 para cada $k \in$ **K faça**
3 **escolha** *k* observações de **D** como os centróides iniciais
4 **repita**
5 **atribua** cada observação de **D** ao cluster com maior similaridade, baseada na média dos objetos no cluster;
6 **atualize** as médias em cada cluster com as observações realocadas;
7 **até convergência**
8 **insira** em **indices** o índice de *CH* associado aos *k* clusters
9 fim
10 retorne indices

Algoritmo 2 Escolhe a melhor configuração de agrupamento dos fatores de inspeção através de um classificador genérico **C**

Entrada: **k** (quantidade de classes desejadas)
F (fatores de inspeção)
D (conjunto de dados correspondente à inspeção)
Saída: Um vetor de acurácias obtidas por um classificador genérico associadas a cada possivel agrupamento de tamanho *k*
1 acuracias = {}
2 agrupamentos = possiveis agrupamentos de fatores
4 soma = soma dos fatores de inspeção
5 para cada *agrupamento* $\in$ **agrupamentos faça**
6 **classe** = {} (classe ou estado de saúde a ser atribuida)
7 **atribua** o valor da classe (0 à k) a variável **classe**, baseada na soma dos fatores de inspeção e no agrupamento da iteração atual
8 **treine** o *classificador genérico* **C** utilizando o conjunto de dados **D** combinado à **classe**, como preditora, em um experimento de validação cruzada.
9 **insira** em **acuracias** a métrica acurácia correspondente ao experimento de validação cruzada na iteração atual
10 fim
11 retorne acuracias

3.1. Validação Cruzada e Ajuste dos Hiper-parâmetros

Dado que já sabemos o valor ótimo do número de clusters ou classes, o Algoritmo 2 entra em ação. Para cada possível agrupamento, são avaliados os 4 classificadores. Em cada avaliação do classificador $j \in \{$k-NN, RF, NN e SVM$\}$, é feito um experimento de validação cruzada de 10-dobras (10-fold) e avaliado a acurácia com um vetor H_j de hiperparâmetros possiveis. O vetor H_j para cada classificador é definido na Tabela 3,

Classificador	Hiperparâmetros Fixos	Hiperparâmetros possíveis
k-NN	Nenhum	k: {1,2, ..., 30}
Random Forest	Nenhum	mtry: {2, 7, 12}, splitrule: {gini, extratrees}
Neural Network	MLP, dim: (400, 200), dropout: 0.45, batch_size: 100, optimizer: Adam	Nenhum
SVM	kernel: RBF, gamma: 1.5, C = 9	Nenhum

Tabela 3. Configurações dos classificadores: k-NN, RF, NN e SVM

4. Resultados

Para execução do Algoritmo 1, foi utilizado um vetor K para os possíveis números de clusters onde $K = 3, ..., 10$. Essa definição se deu com o objetivo de caracterizar no mínimo à partir de 3 estados de saúde, que podem ser entendidos como: saudável, alerta e doente. Para $k = 2$ obtem-se uma classificação binária, em que as duas possíveis classes são extremas e, portanto, não há uma classe que sirva de alerta para o usuário final. O resultado pode ser observado na Figura 1.

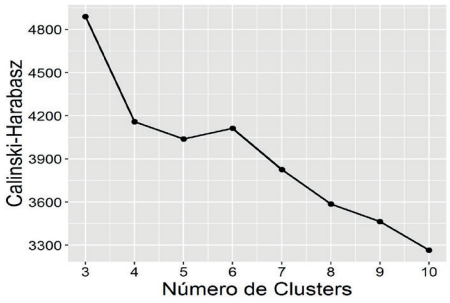


Figura 1. Gráfico resultante da aplicação do Algoritmo 1

O melhor valor de k é 3. Após a definição do numero de classes ($k = 3$), foi possível executar o Algoritmo 2 a fim de definir o melhor agrupamento de fatores de inspeção. O número de possíveis agrupamentos é igual 90. Os resultados são mostrados na Tabela 4, onde a coluna "$Agrupamento$" fornece a identificação de cada possível agrupamento.

Os agrupamentos estão ordenados de acordo com os maiores mínimos de sensibilidade e especificidade interclasses. Essa abordagem nos proporcionará uma configuração de clusters que seja menos penalizada pelos algoritmos no experimento e com alta acurácia. Para escolher o melhor agrupamento tomaremos como base um *trade-off* entre acurácia, sensibilidade e especificidade. Embora, o agrupamento 2456 1 3 nos forneça a maior acurácia dos agrupamentos (99.36%), esse resultado não é o melhor, pois, o que ocasiona essa alta acurácia é o excesso de fatores agrupados. O mesmo ocorre para os agrupamentos 1456 2 3 e 3456 1 2.

Assim, o melhor agrupamento é 2 13 456, que possui o melhor *trade-off* entre acurácia, sensibilidade e especificidade com uma acurácia de 99.23% nas Redes Neurais. É possível observar ainda que as melhores acurácias estão relacionadas à presença dos itens de inspeção 4, 5 e 6 em um mesmo cluster. Assim, esses três itens poderiam ser agrupados para indicação de um nível de saúde. Bem como os itens 1 e 3. Portanto, um alerta poderia ser emitido caso qualquer item do grupo 456 ou 13 apresente um problema.

Vale destacar que os itens 4, 5 e 6 da planilha de inspeção são itens que não estão diretamente ligado à colônia em si, Food (Alimento), Stressors (Estressores) e Space (Espaço), respectivamente. O agrupamento formado pelos itens 1 e 3 possui os itens da planilha de inspeção diretamente ligados ao à rainha. Uma vez que o item 1 (Brood - Ninhada) representa todas as fases da ninhada e ínstares.

5. Conclusões e Trabalhos Futuros

Assim, foi possível obter um modelo de classificação de alta precisão, com taxa de acero de até 99,36%. Foi possível também determinar o melhor agrupamento dos itens da inspeções para definição do nível de saúde de uma colônia de abelhas. Como trabalho futuro, pretende-se aplicar a solução proposta em um conjunto de dados maior.

Agradecimentos

Esse estudo foi financiado em parte pela CAPES - código de financiamento 001. Danielo G. Gomes e Breno M. Freitas agradecem o suporte financeiro do CNPq, processos #302934/2010-3, #311878/2016-4 e #432585/2016-8.

Agrupamento	Acurácia (%)			
	k-NN	R. Forest	NN	SVM
5 12 346	90.11	95.09	91.48	90.73
2 13 456	**98.55**	**99.21**	**99.23**	**98.32**
12 35 46	90.27	94.89	92.55	90.67
2 15 346	89.87	94.89	90.90	90.59
13 25 46	90.18	94.80	91.84	90.21
15 23 46	89.75	94.83	91.61	90.50
1246 3 5	90.50	95.04	91.22	91.12
2 46 135	89.88	94.83	91.61	90.31
13 24 56	94.29	96.69	95.35	94.78
5 23 146	89.57	94.80	91.44	90.21
5 46 123	90.19	95.15	91.97	90.61
2 35 146	89.77	94.88	90.27	90.37
5 13 246	89.95	95.02	92.17	90.85
1456 2 3	**98.50**	**99.11**	**98.60**	**98.49**
2 56 134	94.04	96.65	95.85	94.75
3 56 124	94.56	96.79	96.08	95.07
3 15 246	89.88	94.87	91.54	90.47
1346 2 5	89.97	94.93	92.27	90.71
3 25 146	89.72	94.67	92.40	90.21
2 34 156	94.06	96.65	95.09	94.63
3 24 156	94.26	96.73	95.22	94.76
3 12 456	**98.75**	**99.25**	**99.16**	**98.61**
12 34 56	94.29	96.76	96.30	94.69
3 46 125	90.20	94.87	91.09	90.42
5 34 126	88.88	94.23	90.73	90.01
3 45 126	92.07	95.42	92.96	92.57
1 23 456	**98.52**	**99.12**	**99.16**	**98.33**
14 23 56	93.94	96.56	95.72	94.28
3 14 256	93.95	96.45	95.44	94.54
2 14 356	93.93	96.62	95.50	94.51
4 35 126	88.99	94.13	90.96	89.83
4 23 156	93.91	96.71	95.57	94.25
4 12 356	94.34	96.73	95.65	94.62
2456 1 3	**99.06**	**99.36**	**99.16**	**98.99**
4 13 256	94.00	96.64	95.39	94.38
4 56 123	94.26	96.80	95.80	94.59
1256 3 4	94.28	96.66	95.76	94.86
5 14 236	88.53	94.00	90.40	89.55
4 15 236	88.40	94.09	89.93	89.59
1236 4 5	89.13	94.19	90.85	89.94
4 25 136	88.63	93.98	90.73	89.37
14 26 35	88.66	93.99	90.98	89.50
1 24 356	94.31	96.83	95.44	94.84
2356 1 4	94.15	96.58	95.59	94.60
5 26 134	88.82	94.17	91.18	90.21

Agrupamento	Acurácia (%)			
	k-NN	R. Forest	NN	SVM
1356 2 4	93.82	96.53	95.93	94.49
3 26 145	91.83	95.29	93.80	92.38
5 24 136	88.78	94.22	91.03	89.95
2 45 136	91.89	95.40	93.44	92.26
13 26 45	91.80	95.28	93.37	92.36
3456 1 2	**98.68**	**99.32**	**99.03**	**98.72**
1 34 256	94.04	96.58	95.01	94.74
1 56 234	94.42	96.88	95.93	95.16
5 16 234	89.05	94.25	91.33	90.08
4 26 135	88.57	94.00	91.39	89.59
2 36 145	91.87	95.27	93.01	92.50
3 16 245	92.29	95.37	93.89	92.89
5 36 124	89.27	94.17	92.43	90.37
15 26 34	88.53	94.15	90.83	89.70
1 35 246	89.95	94.98	91.80	90.82
1 45 236	91.92	95.28	93.37	92.14
1 26 345	92.01	95.43	93.72	92.67
1 36 245	92.27	95.48	93.89	92.71
1 46 235	90.15	94.62	91.22	90.76
6 25 134	88.97	94.24	90.85	89.81
1235 4 6	89.18	94.06	90.75	90.04
1234 5 6	89.62	94.42	91.67	90.62
6 12 345	92.52	95.54	93.69	92.81
15 24 36	88.77	94.30	90.42	89.83
6 35 124	89.51	94.12	91.78	90.50
6 23 145	92.09	95.29	93.74	92.31
6 24 135	88.99	94.09	91.20	90.14
16 25 34	88.79	93.94	90.70	89.89
16 23 45	91.89	95.24	93.93	92.25
6 15 234	89.14	94.20	90.64	90.15
4 16 235	88.87	93.87	91.09	89.73
14 25 36	88.55	93.96	91.05	89.49
6 13 245	92.30	95.58	93.87	92.92
12 36 45	92.07	95.39	93.87	92.47
6 45 123	92.30	95.46	94.04	92.78
2345 1 6	92.75	95.66	93.78	93.48
16 24 35	88.88	94.18	89.76	90.16
1 25 346	90.09	94.94	91.74	90.70
6 14 235	88.83	93.86	89.99	89.86
4 36 125	88.97	93.99	90.53	89.74
6 34 125	89.28	94.28	90.75	90.10
1245 3 6	92.85	95.54	93.63	93.28
1345 2 6	92.20	95.47	93.09	92.94
2 16 345	91.96	95.42	93.57	92.55
2346 1 5	90.32	95.27	91.48	90.92

Tabela 4. Resultado da aplicação do Algoritmo 2

Referências

Buuren, S. v. and Groothuis-Oudshoorn, K. (2010). Mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, pages 1–68.

Calinski, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Potts, S., Biesmeijer, J., Kremen, C., Neumann, P., Schweiger, O., and Kunin, W. (2010). Global pollinator declines: Trends, impacts and drivers. *Trends in ecology & evolution*, 25:345–53.

Potts, S. G. et al. (2016). Safeguarding pollinators and their values to human well-being. *Nature*, 540:220–229.

Desenvolvimento de Modelos de Armazenamento em Sensores com Reutilização de Código

Alexandre R. Ordakowski[1], Marcos A. Carrero[3], Martin A. Musicante[2], Aldri L. dos Santos[1], Carmem S. Hara[1]

[1]DINF – Universidade Federal do Paraná – UFPR – Paraná, Brasil
[2]DIMAp – Universidade Federal do Rio Grande do Norte – UFRN – Natal, Brasil
[3]FAE Centro Universitário – Paraná, Brasil

mam@dimap.ufrn.br, {arordakowski,macarrero,aldri,carmem}@inf.ufpr.br

Resumo. *Sensores são componentes essenciais da Internet das Coisas (IoT). Em consonância com a tendência de armazenar dados próximos às fontes de dados, preconizada pela computação de borda e de névoa, alguns sensores podem desempenhar o papel de repositório dos dados capturados. Dado o crescimento da quantidade de dispositivos e de aplicações para IoT, há a necessidade de investigar e desenvolver novos modelos de armazenamento para sensores. No entanto, poucos trabalhos propõem uma abordagem que trate da modelagem e implementação de sistemas de armazenamento e consulta dos dados da rede de uma maneira sistemática. Este artigo investiga a aplicação de um modelo de componentes para a geração de código para os dispositivos sensores, com a elaboração do framework RCBM-S (RCBM for Sensor devices). No RCBM-S a orquestração dos componentes é baseada em máquinas de estados. Um estudo de caso mostra a reutilização de código promovida pelo RCBM-S para códigos em nesC para o sistema TinyOS.*

1. Introdução

Os sensores são componentes computacionais de coleta de dados essenciais ao provimento da Internet das Coisas (IoT). Previsões apontam que a IoT, em suas diversas formas de redes, atingirá a marca de 20 bilhões de objetos conectados em 2020[1]. Em razão do grande volume de dados trocados, o armazenamento eficiente da enorme quantidade de dados gerados por estes dispositivos tem sido um tópico de investigação de vários trabalhos recentes. Uma técnica comum, preconizada pela computação de borda e de névoa, é processar e armazenar os dados em uma extremidade próxima da fonte de dados. Em consonância com esta tendência, é possível designar alguns dispositivos sensores para desempenhar o papel de repositório dos dados capturados. Em razão do crescimento da quantidade de dispositivos, bem como novas aplicações para IoT, surge a necessidade de investigar e desenvolver novos modelos de armazenamento voltado para as redes de sensores sem fio (RSSF), que são componentes fundamentais da IoT.

Os sensores são dispositivos com recursos normalmente limitados, tanto de bateria, como de processamento e armazenamento. Assim, os sistemas desenvolvidos para as RSSFs buscam considerar as características específicas da aplicação a fim de explorar

[1]https://www.informationweek.com/mobile/mobile-devices/gartner-21-billion-iot-devices-to-invade-by-2020/d/d-id/1323081

seus escassos recursos. Ou seja, cada aplicação requer uma análise e desenvolvimento de modelos de armazenamento específicos. Contudo, o desenvolvimento de tais sistemas é muitas vezes complexo. Observa-se que poucos trabalhos na literatura propõem uma abordagem que trate da modelagem e implementação de sistemas de armazenamento e consulta dos dados da rede de uma *maneira sistemática*, que são requisitos necessários para atender à grande demanda por serviços ubíquos de grande escala. Estas questões foram tratadas por [Carrero et al. 2017] através do desenvolvimento do framework RCBM, para apoiar o desenvolvimento de *códigos de simulação* para RSSFs.

Este artigo investiga a aplicação de técnicas similares para a geração de código para implantação nos *dispositivos sensores*, com a elaboração do framework RCBM-S (*RCBM for Sensor devices*). Ao contrário do RCBM, que foca no simulador de redes NS-2, a linguagem utilizada pelo RCBM-S é o nesC. O nesC é atualmente uma das linguagens de programação mais empregadas para RSSFs [Gay et al. 2014] e foi desenvolvido para o sistema operacional (SO) TinyOS. O RCBM-S promove a reutilização de código com a especificação de componentes, cuja orquestração é expressa por uma máquina de estados. Esta máquina permite que o fluxo de execução da aplicação seja modelada com um alto nível de abstração.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta trabalhos relacionados; o framework RCBM-S e o estudo de caso que determina o porcentual de reuso obtido são apresentados nas Seções 3 e 4, respectivamente; a Seção 5 finaliza o artigo apresentando trabalhos futuros.

2. Trabalhos Relacionados

Grande parte dos modelos de desenvolvimento propostos que são para sensores levam em conta máquinas de estados, dada a natureza dos dispositivos, baseados em eventos. Dentre eles podem ser citados o Tokenit [Taherkordi et al. 2015] e o *Communicating X-Machine* (CXM) [de Lima Braga et al. 2010]. As estratégias usadas pelo Tokenit integram um ambiente de modelagem e implementação, no qual a máquina de estados é descrita no formato XML e o compilador do *framework* gera automaticamente código para a plataforma de execução do SO Contiki. Já o *Communicating X-Machine* propõe a definição de *X-Machines Stand-Alone* (isoladas) para a modularização da aplicação, e o uso do método formal CXM para promover a troca de mensagens entre os módulos.

O Wiselib [Baumgartner et al. 2010] é um *framework* para o desenvolvimento de aplicações para RSSF a partir de *templates* genéricos que geram código para diferentes plataformas de dispositivos, como o TinyOS e Contiki. Apesar de fornecer um ambiente de desenvolvimento favorável à reutilização de aplicações para plataformas heterogêneas, o Wiselib não fornece um arcabouço para o reaproveitamento de códigos de aplicações distintas. O RCBM (*Reusable Component-Based Model*) [Carrero et al. 2017] é um modelo baseado em componentes, desenvolvido para sistemas de armazenamento em RSSF em ambientes de *simulação*. Ele promove a reutilização de componentes de software e adota uma máquina de estados com transições baseadas em eventos e lógica para a especificação do fluxo da aplicação. O objetivo do RCBM-S é adotar o mesmo formalismo para o desenvolvimento de códigos para *dispositivos sensores*. Assim, a mesma especificação pode ser usada para desenvolver o código para a validação por simulação, e para a implantação do código em dispositivos para ambientes de sensoriamento reais.

Dentre os trabalhos relacionados, o Wiselib e o X-machine focam no desenvolvimento de código em múltiplas plataformas. No entanto, eles consideram diferentes plataformas de dispositivos sensores e não em um *framework* que atenda tanto a simulação para a validação quanto o desenvolvimento para dispositivos sensores. É possível notar que o RCBM-S contém diversas semelhanças com a ferramenta *Communicating X-Machine*. Entretanto, além do RCBM-S abranger mais plataformas e gerar códigos para nesC e OTcl, ele também utiliza um método formal diferente do utilizado pelo *Communicating X-Machine*. Ambas as ferramentas focam na utilização de máquinas de estado que se comunicam e realizam a transições por eventos, porém o RCBM-S propõe a transição de estados lógica, que é resultante do processamento de informações no estado atual.

3. O Framework RCBM-S

O RCBM-S é um framework para dar suporte ao desenvolvimento de modelos de armazenamento de dados em sensores baseado em componentes, cujo fluxo de execução é especificado por uma máquina de estados. A Figura 1 (a) apresenta a arquitetura do RCBM-S. Embora a arquitetura seja similar à proposta para o RCBM [Carrero et al. 2017], ela foi completamente re-implementada para o sistema TinyOS, na linguagem nesC. A arquitetura tem 3 camadas: especificação (bloco superior), implementação (bloco intermediário) e comunicação (bloco inferior). O objetivo de separar a especificação da implementação é promover a reutilização de código, com a criação de componentes conectáveis, desde que sua interface seja mantida. A camada de comunicação fornece a infraestrutura de comunicação entre os dispositivos. No caso das RSSFs a comunicação é realizada via rádio. Podem ser identificados três tipos de componentes: (i) componentes de biblioteca, que implementam funcionalidades úteis para os modelos, como funções de agregação; (ii) componentes de aplicação, que compõem o modelo de armazenamento e possuem um conjunto de funções que podem ser chamadas por outros componentes ou pelo coordenador; (iii) coordenador: controla o fluxo de execução do sistema, implementando uma máquina de estados.

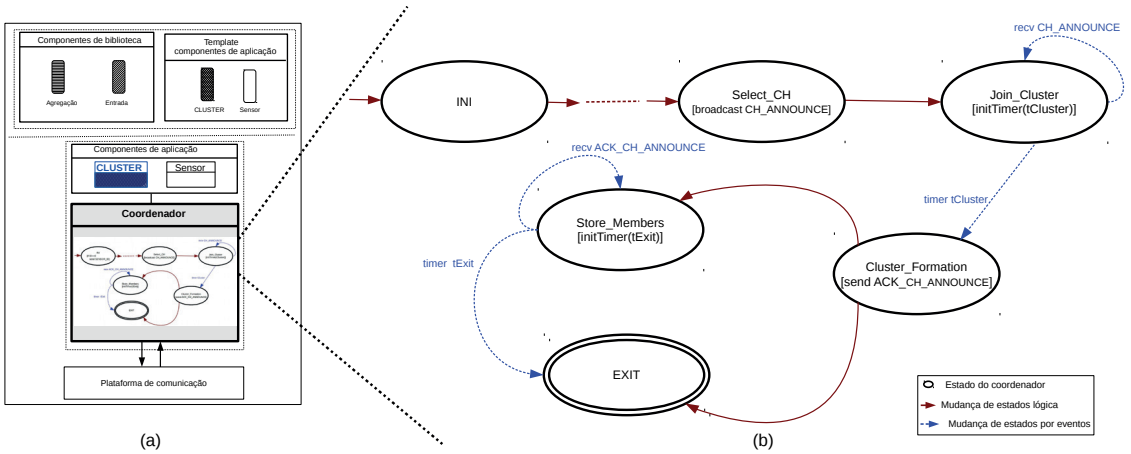


Figura 1. Arquitetura do RCBM-S

Componentes de Biblioteca. Os componentes de biblioteca fornecem funções genéricas e úteis para o desenvolvimento de aplicações. A linguagem nesC já provê alguns, como o componente `Timer`, que fornece um cronômetro que pode ser usado para gerar eventos

em intervalos de tempo regulares. Mas é possível definir novos componentes, como o de funções de agregação, cuja interface é apresentada na Listagem 1. Este componente foi desenvolvido para trabalhar com conjuntos de dados retornando um único valor, como o máximo (max), o mínimo (min), a média (avg) ou a soma (sum) dos seus elementos.

```
1 interface Aggregation {
2     command int max( double val[ ] );
3     command int min( double val[ ]  ] );
4     command double avg( double val[ ] );
5     command int sum( double val[ ] ); }
```

Listing 1. Interface do Componente de Biblioteca de Agregação.

Componentes da Aplicação. Os componentes de aplicação estão diretamente ligados às entidades do seu domínio. Tendo em vista que o RCBM-S foca em modelos de armazenamento de repositório, os componentes de aplicação incluem: (i) dispositivo sensor; e (ii) *cluster*, que é formado por um conjunto de sensores (*cluster-members* - CM) e um sensor que desempenha o papel de *cluster-head* (CH). O CH é responsável por armazenar informações de todo o *cluster*, bem como responder as requisições de consultas. A Listagem 2 apresenta a interface para a entidade *cluster*. São definidas duas funções: a eleição do CH (`selectCH`) e a associação de um membro a um *cluster* (`join`).

```
1 interface Cluster {
2   command int selectCH( int neighbors[] );
3   command int join( int candidates[] );}
```

Listing 2. Template do componente *Cluster*.

Coordenador. O coordenador define o fluxo de execução da aplicação, que é baseado em máquinas de estados, como apresentado na Figura 1 (b). Esta máquina possui transições lógicas (oriundas de um processamento lógico dentro de algum estado) e transições por eventos (que acontecem de maneira reativa na ocorrência de um evento) [Carrero et al. 2018]. O exemplo ilustra parte do processo de formação de *clusters* para o armazenamento de dados no modelo repositório. Na implementação atual do RCBM-S, o coordenador é implementado por funções que tratam as mudanças de estado por um `Timer` (`timer.fired`), o recebimento de mensagens (`receive`) e os estados.

A Listagem 3 apresenta as funções `timer.fired` e `receive`, correspondentes à máquina de estados da Figura 1. O `timer.fired` é um evento disparado ao final de um `Timer`. Por exemplo, no estado *Join_Cluster* é iniciado um `Timer` *tCluster* que, quando finalizado, promove uma transição por evento para o estado *Cluster_Formation*. Isto correponde às Linhas 2-3 da Listagem 3. O evento **receive** trata do recebimento de mensagens. Na Listagem 3, ao receber uma mensagem, é realizado seu desempacotamento e a verificação do tipo da mensagem (Linhas 9 e 11). Embora neste exemplo, a ação seja o armazenamento do conteúdo da mensagem, uma possível ação é acionar uma transição de estados.

```
1 event void Timer0.fired(){
2     if (currentState == JOIN_CLUSTER){
3       state_Cluster_Formation();
4     } else if (currentState == STORE_MEMBERS){
5       state_Exit();}}
6
```

```
7  event message_t* receive(message_t* msg, void* content){
8      GENERAL_MSG* btrpkt = (GENERAL_MSG*)content;
9      if (btrpkt->msgId == CH_ANNOUNCE) {
10         storeMSG(CANDIDATE, btrpkt->sensorId);
11     } else if (btrpkt->msgId == ACK_CH_ANNOUNCE) {
12         storeMSG(MEMBER, btrpkt->sensorId); }}
```

Listing 3. Implementação do coordenador.

Cada estado da máquina corresponde a uma função do coordenador. A Listagem 4 apresenta como exemplo o estado `state_Cluster_Formation`. Ela chama a função `join` do componente `Cluster` que determina qual o CH para o sensor (Linha 3). Caso o CH seja o próprio sensor, ele faz uma transição lógica para o estado `state_Store_Members` (Linhas 4-5). Caso contrário, o sensor transmite a mensagem `ACK_CH_ANNOUNCE` informando que será membro do cluster com líder CH) e muda para o estado `state_Exit`(Linhas 7-8).

```
1  void state_Cluster_Formation(){
2      currentState = CLUSTER_FORMATION;
3      CH = call Cluster.join(candidates_CH);
4      if (CH == TOS_NODE_ID ){
5          state_Store_Members();
6      } else {
7          sendMsg(ACK_CH_ANNOUNCE, CH);
8          state_Exit(); }}
```

Listing 4. Implementação de um estado.

4. Estudo de Caso

Para determinar a efetividade do RCBM-S na promoção da reutilização de código, foram implementados dois modelos de armazenamento em sensores baseados em *cluster*: LCA e LEACH. A Tabela 1 apresenta os resultados obtidos. Como os modelos usam o mesmo fluxo de execução para a formação de *clusters* (apresentado na Figura 1), seus coordenadores são praticamente idênticos, com uma semelhança de mais de 95% das linhas de código. Já na implementação dos componentes, a semelhança é em torno de 75%, visto que os modelos aplicam diferentes critérios para a seleção de CHs e criação de *clusters*.

O LCA elege como CH o nó com o menor identificador único (ID) dentre seus vizinhos e na fase de formação dos *clusters*, os nós não eleitos se unem ao CH no seu alcance com o menor ID. Já o LEACH elege o CH de maneira probabilística, dividida em rodadas, tal que haja uma alternância de CHs e cada sensor do *cluster* seja eleito um número de vezes semelhante. Para a formação dos *clusters*, os sensores se juntam ao CH que tem a maior intensidade de sinal das mensagens de anúncio de CH recebidas. Dadas estas particularidades, grande parte das linhas de código que diferem na implementação dos modelos deve-se ao código das funções `selectCH` e `join`.

O estudo de caso mostra que o desenvolvimento baseado em componentes facilita o desenvolvimento dos modelos, delimitando de forma clara os trechos de código que precisam ser alterados para o desenvolvimento de novos modelos de armazenamento. Estes resultados comprovam que o framework RCBM-S pode promover um grande ganho na reutilização e estruturação do código-fonte de aplicações voltadas ao armazenamento de dados em dispositivos sensores.

Tabela 1. Reuso de código na implementação dos modelos.

	Componentes			Coordenador		
Modelo de Sistema	**Total de Linhas**	**Qtd. Linhas Idênticas**	**% Linhas Idênticas**	**Total de linhas**	**Qtd. Linhas Idênticas**	**% Linhas Idênticas**
LCA	113	89	78.76 %	193	192	99.48 %
LEACH	119	89	74.78 %	199	191	95.97 %

5. Conclusão

Este artigo apresentou o framework RCBM-S, que tem como objetivo apoiar o desenvolvimento de modelos de armazenamento em RSSFs para a implementação do código em dispositivos sensores. Ele facilita a implementação de modelos que se adequem às particularidades das aplicações e que explorem de forma eficiente os escassos recursos dos dispositivos sensores. O estudo de caso mostrou que a proposta promove a reutilização de código de 75% dos componentes e 99% do coordenador no desenvolvimento dos modelos LCA e LEACH. No trabalho de [Carrero et al. 2019] foi proposta a linguagem chamada SLEDS, para o desenvolvimento do coordenador do *framework* RCBM no simulador de redes NS-2. Ela é baseada no modelo de máquinas de estado. Como trabalho futuro é planejada a tradução automática de um programa SLEDS para nesC. Assim, um mesmo código na linguagem SLEDS poderá ser utilizado tanto para a geração de código de simulação, bem como para a geração de código para dispositivos sensores.

Referências

Baumgartner, T., Chatzigiannakis, I., Fekete, S., Koninis, C., Kroller, A., and Pyrgelis, A. (2010). Wiselib: A generic algorithm library for heterogeneous sensor networks. In *European Conference on Wireless Sensor Networks*, pages 162–177. Springer.

Carrero, M., Zamproni, K., Musicante, M. A., Santos, A., and Hara, C. (2018). Uma máquina de estados para especificação de códigos de simulação para redes de sensores sem fio urbanas. In *Simpósio Brasileiro de Computação Ubíqua e Pervasiva*.

Carrero, M. A., Musicante, M. A., dos Santos, A. L., and Hara, C. S. (2017). A reusable component-based model for wsn storage simulation. In *Proceedings of the 13th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 31–38.

Carrero, M. A., Musicante, M. A., dos Santos, A. L., and Hara, C. S. (2019). Sleds: A dsl for data-centric storage on wireless sensor networks. *Communications in Computer and Information Science*, 926:74–89.

de Lima Braga, M., de Jesus dos Santos, A., and de Lucena Junior, V. F. (2010). Modelagem e geração de código para redes de sensores sem fio usando communicating x-machine. In *Proc. of the 9th Int. Information and Telecommunication Technologies Symposium*.

Gay, D., Levis, P., Von Behren, R., Welsh, M., Brewer, E., and Culler, D. (2014). The nesc language: A holistic approach to networked embedded systems. *Acm Sigplan Notices*, 49(4):41–51.

Taherkordi, A., Johansen, C., Eliassen, F., and Römer, K. (2015). Tokenit: Designing state-driven embedded systems through tokenized transitions. In *2015 Int. Conf. on Distributed Computing in Sensor Systems*, pages 52–61. IEEE.

Um Estudo Comparativo de Mecanismos de Privacidade Diferencial sobre um *Dataset* de Ocorrências do ZIKV no Brasil*

Daniel de Oliveira[1], Eduardo Rodrigues[2], Serafim Costa[2], Paulo Amora[2], Asley Caldas[2], Marco Horta[3], Ana Maria de Filippis[3], Kary Ocaña[4], Vânia Vidal[2], Javam Machado[2]

[1]Universidade Federal Fluminense (UFF)

danielcmo@ic.uff.br

[2]Universidade Federal do Ceará (UFC)

{eduardo.rodrigues, serafim.costa, paulo.amora, javam.machado}@lsbd.ufc.br

[3]Laboratório Nacional de Computação Científica (LNCC)

karyann@lncc.br

[4]Fundação Oswaldo Cruz (Fiocruz)

marco.horta@fiocruz.br

Resumo. *Nos últimos anos, o Governo Brasileiro tem organizado uma série de iniciativas para informatizar o SUS, com o objetivo de melhorar sua eficiência. Uma dessas iniciativas é o GAL (Gerenciador de Ambiente Laboratorial). O GAL tem como objetivo proporcionar a gerência das rotinas laboratoriais e o acompanhamento das etapas para realização dos exames. Adicionalmente, o GAL permite a extração de dados, que podem ser usados por gestores nas diversas esferas. Entretanto, essa exportação de dados pode não ser confiável, e levar a sérios riscos de violação de privacidade, uma vez que exibe dados pessoais de indivíduos. Simplesmente mascarar os elementos de identificação (nome, CPF, etc.) ou disponibilizar apenas resultados agregados pode não proporcionar proteção suficiente. Nesse cenário, técnicas mais elaboradas de privacidade de dados, como a Privacidade Diferencial (PD), se fazem necessárias. Este artigo apresenta um estudo que compara a aplicação de diferentes mecanismos de PD sobre os dados extraídos do GAL. Em especial, utilizamos como estudo de caso os dados de exames de casos suspeitos do Vírus da Zika (ZIKV) no Brasil.*

1. Introdução

Durante as últimas décadas, as comunidades da computação e da saúde têm despendido grandes esforços para prover soluções computacionais na área da saúde [Li et al. 2019]. Segundo [Shishvan et al. 2018], essas soluções variam desde o monitoramento de pacientes internados até a gerência de recursos na área da saúde. Esse tipo de iniciativa pode ser percebido também no Brasil [Silva et al. 2019]. Um dos exemplos é o *Sistema Gerenciador de Ambiente Laboratorial* (GAL)[1] do SUS. O GAL tem como objetivo informatizar a rede laboratorial de saúde pública brasileira, *i.e.,* ele registra informações de amostras de origem humana e animal que possam ter sido expostas à doenças, possibilitando que profissionais da saúde possam consultar, extrair e inferir conhecimento a partir dos dados para desempenhar vigilâncias epidemiológicas.

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Os autores agradecem também ao CNPq e FAPERJ por financiarem parcialmente a pesquisa.

[1]https://gal.nacional.sus.gov.br/

Os dados exportados pelo GAL são fundamentais para identificar a circulação, distribuição e epidemiologia de diversos vírus no país.

Entretanto, essa exportação dos dados para análises e descoberta de padrões pode não ser totalmente confiável, e levar a sérios riscos de violação de privacidade, uma vez que pode exibir dados pessoais de pacientes. A área de saúde tem sido um alvo importante no que tange o acesso indevido à informação sensível [Dagher et al. 2018], uma vez que os registos de saúde contêm frequentemente informações privadas e sensíveis dos pacientes, *e.g.,* nomes, CPFs, endereço e particularmente o acometimento de doenças. Dessa forma, a anonimização dos dados na área de saúde se torna uma tarefa prioritária.

A anonimização de dados é uma técnica de preservação de privacidade que objetiva modificar valores dos atributos de um *dataset* com o objetivo de ocultar a identidade e/ou informações sensíveis de indivíduos. Todavia, essa não é uma tarefa simples, e apenas mascarar os elementos de identificação do indivíduo ou disponibilizar resultados agregados pode não proporcionar proteção suficiente. Por exemplo, consideremos um *dataset* contendo 100 exames de pacientes, dos quais apenas 2 são relativos à população indígena. Mesmo que os nomes, CPFs e endereços sejam mascarados ou omitidos, ainda pode ser possível identificar os indivíduos por outros atributos como a etnia. O desafio se encontra então em disponibilizar dados anonimizados para consulta que apresentem um nível de perturbação que não permita identificar os indivíduos nesse *dataset*, mas que ao mesmo tempo não torne os dados "inúteis". Assim, a escolha da técnica de anonimização é uma tarefa importante, uma vez que esta modificação pode acarretar em perda de informação, o que implica na diminuição da utilidade dos dados.

Diversas técnicas de privacidade e anonimização de dados já foram propostas na literatura [Warner 1965, Erlingsson et al. 2014, Dwork et al. 2006]. Uma das técnicas mais utilizadas é a Privacidade Diferencial [Dwork et al. 2006] (PD), que provê garantias de privacidade por meio do uso de um algoritmo aleatório, denominado mecanismo. Os mecanismos mais comuns existentes na literatura para garantir a PD são o Exponencial e o de Laplace. Além disso, outros mecanismos como a Resposta Randômica (*Randomized Response - RR*) [Warner 1965] podem ser aplicados. Cada um desses mecanismos possui vantagens e desvantagens, e pode ser mais adequado para um determinado *dataset*. Avaliar a utilidade de cada um deles para um determinado *dataset* se torna uma importante tarefa a ser desempenhada. O presente artigo tem como objetivo realizar um estudo comparativo de um conjunto de mecanismos de PD sobre um *dataset* extraído do GAL. Esse *dataset* contém os exames realizados com pacientes com casos suspeitos ou confirmados do Vírus da Zika (ZIKV) em diversos estados do Brasil (com nomes e CPFs sintéticos). Por meio de uma série de consultas analíticas, o estudo busca determinar o mecanismo mais adequado para ser aplicado para este tipo de *dataset*. Esse artigo se encontra organizado em 3 seções além da Introdução. Na Seção 2, o referencial teórico e os trabalhos relacionados são discutidos. A Seção 3 apresenta o estudo comparativo, e, finalmente a Seção 4 conclui o presente artigo.

2. Privacidade de Dados

A Privacidade Diferencial (PD), técnica de privacidade para anonimização de dados, é um modelo matemático que tem como objetivo permitir análises estatísticas sobre um conjunto de dados, sem comprometer a privacidade dos indivíduos. Ela assegura que qualquer resposta à uma determinada consulta, tem ocorrência igualmente possível e independe da presença, ou ausência, de um indivíduo no *dataset*. Dessa forma, a PD insere um mecanismo de aleatoriedade na adição do ruído à resposta de consultas realizadas sobre o *dataset*. Em especial, neste artigo iremos realizar uma análise sobre os mecanismos de Laplace, Exponencial e RR. A seguir, discutimos com mais detalhes cada um desses mecanismos.

O mecanismo de Laplace [Dwork et al. 2006] é a forma mais comum de se obter PD para uma dada consulta. O mecanismo adiciona ruído randômico obtido por meio da distribuição de Laplace à resposta original da consulta. A distribuição de Laplace é centralizada em 0 e com parâmetro escala b tem a distribuição $Lap(z|b) = \frac{1}{2b}exp(-\frac{|z|}{b})$. Assim, dada uma consulta $f : \mathbb{N}^{|x|} \rightarrow \mathbb{R}^k$ o mecanismo é expressado como $M_L(x, f, \epsilon) = f(x) + (Y_1...Y_k)$, onde $Y_i \sim Lap(\frac{\Delta f}{\epsilon})$ i.i.d. (variáveis aleatórias independentes e identicamente distribuídas). Portanto, a quantidade de ruído adicionado depende da sensibilidade global Δf e do parâmetro de privacidade ϵ. A sensibilidade é a métrica utilizada para mensurar o impacto de um indivíduo, ao ser removido do *dataset*. Quanto maior for o valor da sensibilidade, maior será a quantidade de ruído adicionada. O mecanismo Exponencial foi projetado para responder perguntas categóricas (não-numéricas), escolhendo a "melhor" resposta. De forma a ser capaz de comparar classes categóricas, é necessário uniformizar as entradas do mecanismo. Assim, a uniformização é realizada por meio de uma função de utilidade, que deve ser capaz de prover uma ordem para as classes. A função de utilidade mapeia um registro a um determinado valor para todos os registros e a sensibilidade Δf é calculada por $\Delta f = \max_{x,y:||x-y||_1 \leq 1} |u(x) - u(y)|$. De posse de Δf, o mecanismo é expresso como $M_E(x, u) = f(x)+(Y_1...Y_k)$, onde $Y_i \sim exp(\frac{\epsilon u(x)}{2\Delta f})$. A Resposta Randômica (RR) é uma técnica originalmente aplicada para obter respostas privadas em entrevistas. O objetivo é garantir que os entrevistados respondam questões sensíveis, *e.g.,* sexualidade ou crença religiosa, mantendo a confidencialidade das respostas [Warner 1965]. Para isto, é adicionado um processo de aleatoriedade na resposta do entrevistado, mascarando se a resposta é verdadeira ou não. Por exemplo, é perguntado ao usuário se ele é a favor da legalização das drogas. Como processo de aleatoriedade, o entrevistado joga uma moeda, em segredo, e responde "sim" se der cara, ou responde a verdade se der coroa. O RAPPOR (*Randomized Aggregatable Privacy-Preserving Ordinal Response*), proposto por [Erlingsson et al. 2014], é um mecanismo para coleta de dados estatísticos com fortes garantias de privacidade que utiliza a técnica de RR. Para qualquer valor coletado, o RAPPOR entrega uma forte garantia de privacidade para o indivíduo, que limita a informação privada divulgada, medida pelo limiar ϵ-privacidade diferencial.

2.1. Trabalhos Relacionados

Alguns trabalhos na literatura executam estudos comparativos de técnicas de privacidade de dados. [Nascimento et al. 2018] comparam técnicas de anonimização e perturbação de dados em um *dataset* da área de saúde. Em especial, os autores focam nas abordagens *k-anonimato* e *l-diversidade*. Diferentemente da PD, tanto o *k-anonimato* quanto a *l-diversidade* são modelos sintáticos, *i.e.,* se usadas essas técnicas, elas podem permitir um usuário malicioso com conhecimento prévio ser capaz de identificar os indivíduos no *dataset*. [Kifer and Machanavajjhala 2011] analisam o mecanismo de Laplace para verificar se o mesmo pode ser aplicado para dados de redes sociais. O objetivo dos autores é utilizar o teorema *no-free-lunch* para afirmar que é impossível fornecer privacidade e utilidade sem realizar suposições sobre os dados. Entretanto, os autores não comparam outros mecanismos em sua avaliação.

3. Estudo Comparativo

Nessa seção são apresentadas as características do *dataset* utilizado no estudo comparativo, as consultas utilizadas como base para o estudo e as discussões acerca dos resultados obtidos. Para realização do estudo, foi exportado um *dataset* do GAL contendo 104 atributos e 25.133 registros relativos à exames realizados por pacientes suspeitos e confirmados do ZIKV. A qualidade dos dados foi avaliada de acordo com o percentual dos registros preenchidos para cada atributo.

Foram utilizados os parâmetros do SINAN para qualidade (ruim ($< 70\%$), regular ($70\% - 89\%$) e excelente ($\geq 90\%$)). Foi utilizado nesse estudo um *dataset* com completude regular. De forma a comparar os diferentes tipos de mecanismos de PD, fizemos uso de um conjunto de consultas, definidas por especialistas, que pode ser submetido ao *dataset* do GAL. Para cada uma das consultas, aplicamos cada um dos mecanismos sobre a resposta destas consultas e calculamos a medida de erro relativo ($ERel$) entre resposta original e a anonimizada. $ERel$ pode ser definido como $ERel = \frac{|x-x'|}{x}$, onde x representa o valor original e x' o valor com ruído. Desta forma, planeja-se analisar, quanto à utilidade, como cada mecanismo se comporta para as consultas sobre o *dataset*. As consultas utilizadas foram: (i) "Qual o percentual de grávidas infectadas com o ZIKV no Brasil (Q1)?"; (ii) "Qual a quantidade de casos suspeitos por Estado (Q2)?"; (iii) "Qual a quantidade de casos suspeitos por mês no ano de 2016 de um determinado estado (Q3)?".

A Figura 1(a) apresenta os resultados obtidos para a consulta Q1, que retorna apenas um valor numérico percentual. As barras azuis apresentam o valor da consulta antes da aplicação do mecanismo e as barras laranja os valores anonimizados. Pode-se perceber que apesar de os três mecanismos oferecerem resultados equivalentes em termos de $ERel$, o mecanismo que apresentou o menor $ERel$ foi o RR com $0,63\%$, seguido pelo mecanismo de Laplace com $2,25\%$, e, finalmente, do mecanismo exponencial com $10,79\%$. As Figuras 1(b), 1(e) e 1(h) apresentam os resultados das consultas Q2 para os mecanismos Exponencial, RR e Laplace, respectivamente. Nas barras azuis, são apresentados os resultados originais das consultas, enquanto que nas barras laranja, o resultado anonimizado pelo respectivo mecanismo. Podemos perceber que para todos os mecanismos a anonimização foi bem sucedida para os estados com as maiores quantidades de ocorrências. Entretanto, o $ERel$ não pode ser negligenciado no caso dos estados com poucas ocorrências no período. Dessa forma, a consulta Q3 foi avaliada para cada um dos mecanismos para os estados do Rio de Janeiro (RJ - maior quantidade de casos) e da Bahia (BA - menor quantidade de casos).

No caso da BA e do RJ para a consulta Q3 e uso do mecanismo exponencial (Figuras 1(c) e 1(d), respectivamente), observamos que o $ERel$ é bastante diferente para cada um dos estados. No caso do RJ, dada a grande quantidade de ocorrências, temos que $ERel < 2\%$. Entretanto, no caso da Bahia, o $ERel$ não pode ser negligenciado, onde $ERel_{BA} = 26,6\%$ em média, apesar de ter sido aplicada uma técnica que distribui o ruído ao longo dos meses considerados (o cenário de janeiro é o mais complicado, pois só foram considerados 4 casos). Dessa forma, a anonimização para o RJ foi capaz de manter a utilidade dos dados, enquanto que para a BA, pode-se perceber um maior $ERel$. O mesmo comportamento é encontrado para o RR (Figuras 1(f) e 1(g)), onde $ERel_{BA} = 59\%$, e para o mecanismo de Laplace (Figuras 2(a) e 2(b)), onde $ERel_{BA} = 61,1\%$. Apesar de os três mecanismos apresentarem $ERel$ não negligenciáveis para o estado da BA, existe uma diferença de qualidade entre os mesmos. Tomemos como exemplo o mês de junho de 2016. Nesse mês, temos 8 casos confirmados, porém os mecanismos exponencial, RR e Laplace retornam os valores 10, 4, 25, respectivamente, mostrando que mesmo com a utilidade reduzida, o mecanismo exponencial ainda foi o que apresentou o melhor resultado para essa consulta.

Em especial, no caso do mecanismo RR, o $ERel$ aumenta à medida que a quantidade de categorias também aumenta, como nas consultas Q2 e Q3. Isso acontece pois a quantidade de categorias tem um efeito inverso na probabilidade de se obter respostas verdadeiras. Logo, o mecanismo RR pode ser indicado para consultas com poucas categorias, como a Q1. Para contornar esta restrição do RR, é possível aplicar o mecanismo apenas sobre a resposta da consulta agregada (e não antes).

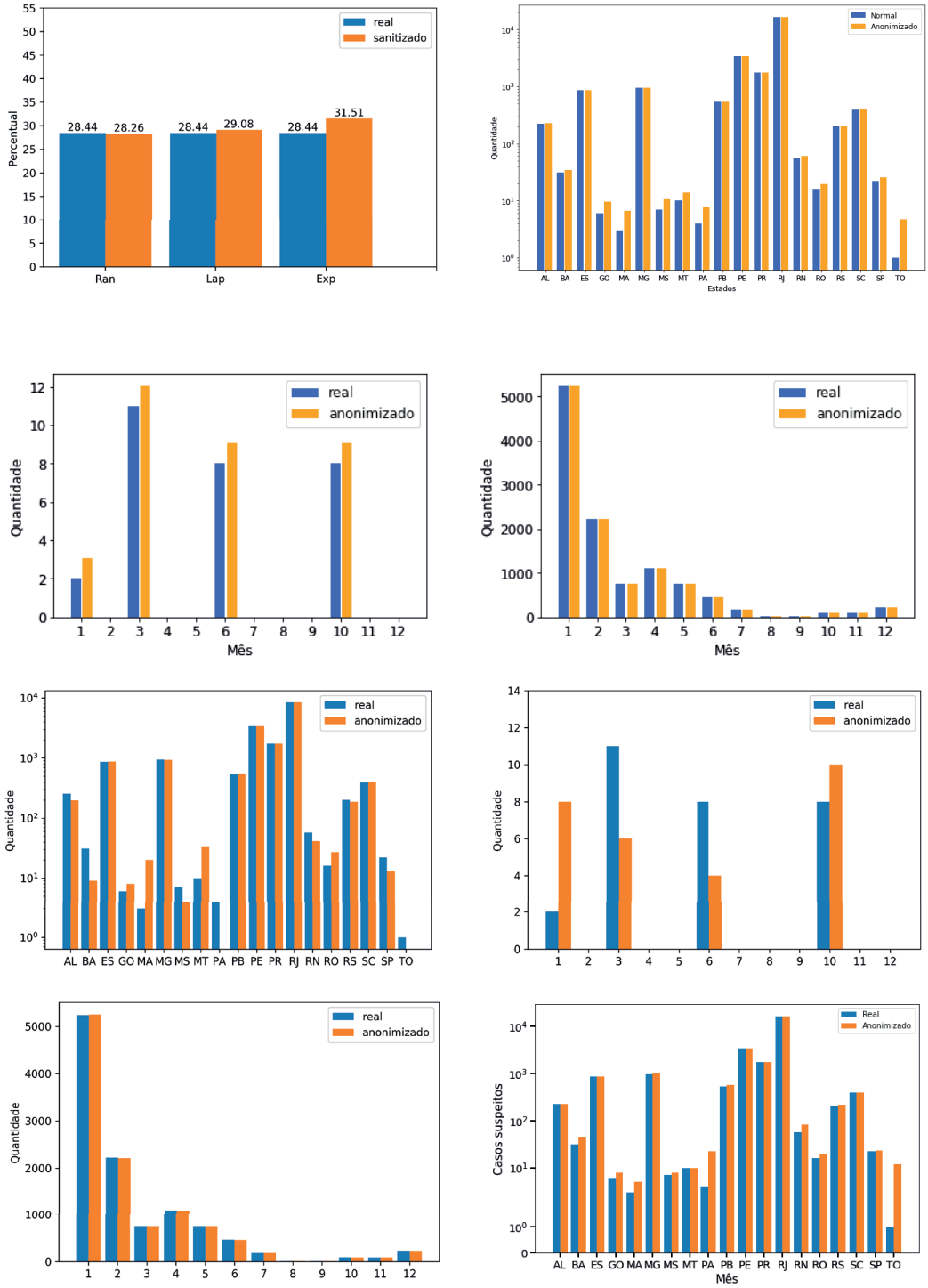


Figura 1. Repostas das consultas originais e anonimizadas (a) Q1 (b) Q2 - Mecanismo Exponencial (c) Q3/BA - Mecanismo Exponencial (d) Q3/RJ - Mecanismo Exponencial (e) Q2 - Randomized Response (f) Q3/BA - Randomized Response (g) Q3/RJ - Randomized Response (h) Q2 - Mecanismo Laplace

4. Conclusão

A comunidade da saúde precisa divulgar dados para a sociedade. Ao mesmo tempo, a privacidade dos pacientes terá que ser assegurada. Entretanto, mesmo quando utilizamos técnicas elaboradas como a Privacidade Diferencial, pode não ser simples definir qual o melhor mecanismo a ser usado para o *dataset*. O presente artigo comparou três mecanismos de Privacidade

Diferencial aplicados sobre um *dataset* de ocorrências do ZIKV no Brasil. A partir dos resultados obtidos, foi possível inferir qual dos mecanismos mantém um certo nível de utilidade dos dados anonimizados. Pode-se observar que o mecanismo de Respostas Randômicas funciona melhor para consultas que retornam poucas categorias, enquanto que o mecanismo Exponencial apresentou melhores resultados para consultas com muitas categorias. Como trabalhos futuros, pretendemos executar experimentos em *datasets* maiores de ZIKV, além de incorporar os mecanismos avaliados na camada de consulta de um *Data Lake* que vem sendo desenvolvido em conjunto com a Fundação Oswaldo Cruz.

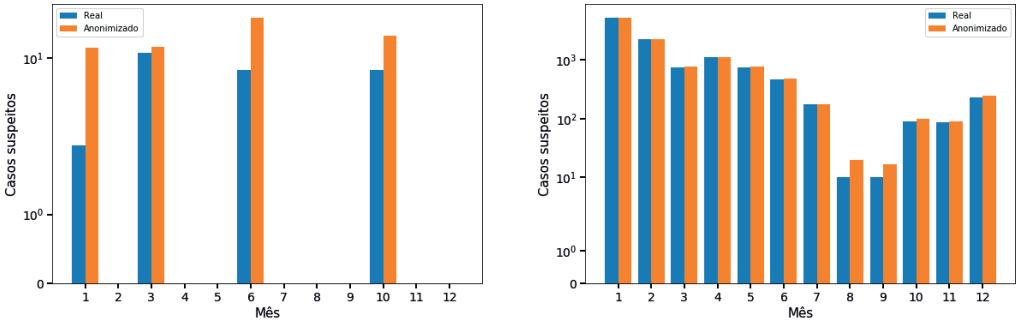


Figura 2. Repostas das consultas originais e anonimizadas (a) Q3/BA - Mecanismo Laplace (b) Q3/RJ - Mecanismo Laplace

Referências

[Dagher et al. 2018] Dagher, G. G., Mohler, J., Milojkovic, M., and Marella, P. B. (2018). Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable Cities and Society*, 39:283 – 297.

[Dwork et al. 2006] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Halevi, S. and Rabin, T., editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Erlingsson et al. 2014] Erlingsson, Ú., Korolova, A., and Pihur, V. (2014). RAPPOR: randomized aggregatable privacy-preserving ordinal response. *CoRR*, abs/1407.6981.

[Kifer and Machanavajjhala 2011] Kifer, D. and Machanavajjhala, A. (2011). No free lunch in data privacy. In *Proc. of the 2011 SIGMOD*, SIGMOD '11, pages 193–204, New York, NY, USA. ACM.

[Li et al. 2019] Li, S., Bamidis, P. D., Konstantinidis, S. T., Traver, V., Car, J., and Zary, N. (2019). Setting priorities for EU healthcare workforce IT skills competence improvement. *Health Inf. Journal*, 25(1).

[Nascimento et al. 2018] Nascimento, F., Vale, K. O., and Gorgônio, F. L. (2018). Um estudo comparativo entre algoritmos de proteção da privacidade aplicado à bases de dados na área de saúde. In *XXXIII SBBD, Rio de Janeiro, RJ, Brazil, August 25-26, 2018.*, pages 301–306.

[Shishvan et al. 2018] Shishvan, O. R., Zois, D., and Soyata, T. (2018). Machine intelligence in healthcare and medical cyber physical systems: A survey. *IEEE Access*, 6:46419–46494.

[Silva et al. 2019] Silva, A. B., Guedes, A., Síndico, S., Vieira, E., and de Andrade Filha, I. (2019). Registro eletrônico de saúde em hospital de alta complexidade: um relato sobre o processo de implementação na perspectiva da telessaúde. *Ciência & Saúde Coletiva*, 24:1133–1142.

[Warner 1965] Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69.

Uma Análise Experimental da Utilização de Diferentes Tecnologias de Armazenamento em um SGBD Relacional

Francisco D. B. S. Praciano[1], **J. Filipe L. de Sousa**[1], **Javam C. Machado**[1]

[1]Laboratório de Sistemas e Bancos de Dados (LSBD)
DC/UFC – CEP 60440-900 – Fortaleza – CE – Brasil

{daniel.praciano,filipe.lobo,javam.machado}@lsbd.ufc.br

Abstract. *Traditional Database Management Systems (DBMS) are built on the premise that data is stored on hard disks drives (HDD). Recently, alternatives to HDDs have emerged, such as solid state drives (SSD), non-volatile memories (NVM) and new main memories (DRAM). Different characteristics of these devices may impact the performance of DBMSs. In this work, we propose to analyze a DBMS that stores its data in four different ways, in HDD, SSD NVM, DRAM and in a hybrid way, using the three devices together. To do this, we use a TPC-C workload and discuss the reasons that give rise to the results obtained for each type of storage.*

Resumo. *Os Sistemas Gerenciadores de Banco de Dados (SGBD) tradicionais são construídos com a premissa de que os dados estão armazenados em discos rígidos (HDD). Recentemente, surgiram várias alternativas aos HDDs, tais como as unidades de estado sólido (SSD), as memórias não voláteis (NVM) e as novas memórias principais (DRAM). As diferentes características desses dispositivos podem impactar no desempenho dos SGBDs. Neste trabalho, nos propomos a analisar um SGBD que armazena seus dados de quatro formas distintas, em HDD, SSD NVM, DRAM e de forma híbrida, utilizando os três dispositivos em conjunto. Para isso, usamos a carga de trabalho TPC-C e discutimos os motivos que dão origem aos resultados obtidos para cada tipo de armazenamento.*

1. Introdução

Dada a natureza dos Sistemas Gerenciadores de Banco de Dados (SGBDs), o desenvolvimento de novas tecnologias de armazenamento juntamente com a evolução tecnológica dos dispositivos trazem benefícios tais como o aumento na vazão de transações efetuadas. É muito comum que esses sistemas tenham sido desenvolvidos considerando as características dos dispositivos disponíveis, dificultando o acompanhamento da evolução tecnológica e a capacidade de tirar proveito das melhorias trazidas pelos novos dispositivos de armazenamento. Pensando nisso, esse trabalho tem como objetivo realizar uma avaliação experimental de um SGBD relacional a fim de pontuar e avaliar o impacto que a substituição da tecnologia de armazenamento subjacente pode causar nesses sistemas. Para tanto, nos propomos a observar a performance do PostgreSQL com os diferentes dispositivos de armazenamento, HDD, SSD NVM e DRAM. A investigação apresentada nesse trabalho busca discutir esse impacto por meio dos seguintes questionamentos: 1) Qual o impacto das tecnologias de armazenamento na vazão de um SGBD tradicional? 2) Dada as características dos dados e da carga de trabalho, é possível construir um modelo híbrido de armazenamento? 3) Qual seria a relação custo-benefício desse armazenamento híbrido?

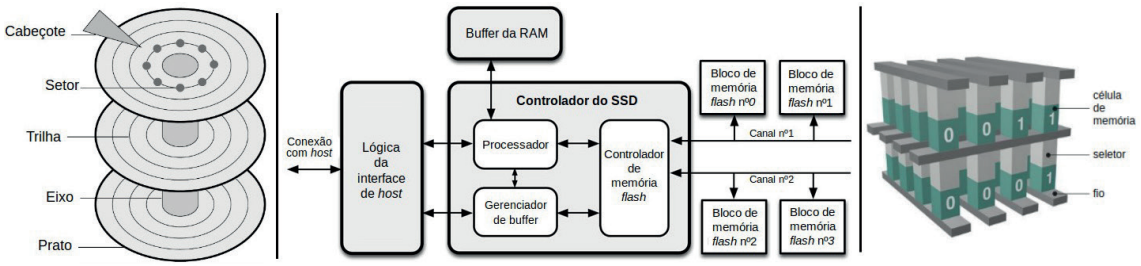


Figura 1. Esquema de funcionamento do HDD, do SSD e do NVM 3D XPoint, respectivamente. Modificado de [Zukowski 2009, Leo Kelion 2015].

2. Tecnologias de Armazenamento

Os discos rígidos (HDD) são os dispositivos de armazenamento mais utilizados pelos SGBDs relacionais. Esses sistemas adotam várias premissas considerando que o armazenamento dos dados seria feito em um HDD, por exemplo a adoção de uma interface de armazenamento orientada a blocos. Nos últimos anos, novos dispositivos de armazenamento têm sido propostos, como as unidades de estado sólido (SSD) tanto baseadas em memórias *flash* quanto nas novas tecnologias de memórias não-voláteis (NVM).

A Figura 1 mostra os componentes da arquitetura desses dispositivos em que a ausência de partes mecânicas nos SSDs, como eixo de rotação ou cabeçote de leitura, fornece armazenamento persistente com taxas de acesso mais altas do que dos atuais HDDs, enquanto que seu custo por *byte* armazenado tem consistentemente diminuído a cada ano [Shah et al. 2008]. O SSD NVM utilizado nesse trabalho, Intel Optane, é construído baseado na tecnologia *3D XPoint* com uma estrutura em três dimensões (3D) e que possibilita o acesso direto às células de armazenamento. Essa organização, mostrada na Figura 1, traz melhorias tanto no desempenho quanto na densidade das células quando comparado aos SSDs baseados na tecnologia *flash*.

A evolução das tecnologias de DRAMs tornou viável a construção de SGBDs totalmente voltados para a memória principal. Esse tipo de sistema potencializa a maneira de se organizar os dados armazenados de tal modo a aproveitar o endereçamento por *bytes* fornecido por esse tipo de dispositivos. Nesse trabalho, o SGBD utilizado é o PostgreSQL que implementa uma interface orientada a blocos no seu módulo de armazenamento. Por esse motivo, vamos utilizar o sistema de arquivos *ramfs* que permite o endereçamento por blocos da memória principal para realizar a avaliação nesse tipo de dispositivo em paridade com os outros dois que são orientados a blocos.

3. Trabalhos Relacionados

Vários trabalhos na literatura se propuseram a realizar estudos sobre o impacto de armazenar dados em tecnologias distintas nos sistemas de bancos de dados. [Xu et al. 2015] apresentou uma caracterização juntamente com uma análise experimental detalhada do desempenho dos SSDs que fazem uso do padrão NVMe. Para tanto, foram utilizados o Cassandra, o MongoDB e o MySQL. O nosso trabalho também aborda o desempenho de sistema relacional através de uma avaliação experimental, mas diferencia-se quer seja no fato de utilizar um dispositivo diferente, quer seja na maneira de avaliação.

Em um outro trabalho, [Brayner and Monteiro 2016] investigaram o desempenho

de três SGBDs comerciais usando como dispositivo de armazenamento tanto HDD quanto SSD argumentando que, no futuro, esses sistemas deveriam ser conscientes do *hardware* subjacente. A avaliação foi feita com a utilização das cargas de trabalhos TPC-H e TPC-E, diferente da nossa escolha pela carga TPC-C. Além disso, sua avaliação fez uso de dispositivos SSDs de interface SATA e de uso pessoal, diferentemente da nossa estratégia que avalia o armazenamento em um SSD 3D XPoint de alto desempenho.

Um estudo experimental dos SSDs NVMe foi realizado em [Son et al. 2016] com o uso de *microbenchmarks* e também com o TPC-C, sendo o MySQL usado para gerenciar os dados. Os autores mostraram os resultados obtidos quando várias configurações de E/S tanto do sistema operacional quanto do SGBD são usadas. Além da diferença do SGBD escolhido, experimentamos outras configurações utilizando tecnologia diferente.

4. Avaliação do Impacto

Este trabalho investiga, por meio da execução de cargas transacionais, o desempenho medido por vazão e latência de um SGBD relacional quando tecnologias distintas de armazenamento são utilizadas. Para cargas transacionais, o *benchmark* TPC-C é considerado padrão, assim optamos por fazer uso da implementação OLTP-Bench [Difallah et al. 2013].

Para a realização dos experimentos, foi utilizada uma máquina com HDD, SSD NVMe e DRAM conforme a Tabela 1, um processador Intel Xeon E5-2609 v3 1.9GHz, 15M Cache, 6 núcleos, com a versão do kernel GNU/Linux 4.15.0. Além disso, o PostgreSQL (versão 11.1) foi escolhido como SGBD relacional. Para obter cada resultado apresentado nas próximas subseções foram realizados 10 execuções independentes em cada um dos dispositivos do respectivo experimento e, então, a média da métrica de interesse foi calculada. Na avaliação de cada dispositivo, nos asseguramos de eliminar E/Ss nos outros dispositivos. Por fim, destaca-se que o cache utilizado pelo PostgreSQL é composto de duas partes: uma própria e uma outra fornecida pelo sistema operacional (SO). Como essa última pode impactar no resultado obtido, por exemplo melhorando o desempenho do HDD, modificamos o nosso *setup* de forma a assegurar que o cache do SO não afete o desempenho dos dispositivos.

4.1. O Impacto no Desempenho

Antes de avaliarmos o PostgreSQL com os dispositivos mencionados, inicialmente fizemos uso de duas ferramentas, *dd* e *fio*, que estão presentes nos sistemas Linux a fim de pontuarmos as diferenças de desempenho entre os dispositivos sem ainda utilizar um sistema complexo como um SGBD. A variação da taxa de transferência entre os dispositivos é mostrada na Figura 2 quando fazemos uso da ferramenta *dd* que permite a leitura ou escrita de arquivos em um dispositivo específico. Esta Figura mostra a taxa de transferência alcançada para ler e escrever um arquivo de 1GB em cada um dos dispositivos. É possível observar que, tomando como referência a taxa de transferência do HDD, o SSD NVM tem taxa 5 vezes maior, enquanto que para a DRAM a mesma taxa é 15 vezes maior.

Da mesma forma, usamos o *fio* a fim de obter mais detalhes sobre as possíveis maneiras de realizar operações de E/S: leitura aleatória (LA), escrita aleatória (EA), leitura sequencial (LS) e escrita sequencial (ES). Na Figura 3, é apresentada a vazão (número de operações de entrada e saída por segundo - IOPS) dessas operações em cada um dos dispositivos. Chama a atenção a pequena vazão alcançada pelo HDD quanto às operações realizadas de maneira aleatória. Isso é esperado devido às características do HDD

Característica	HDD 7.2K RPM	SSD NVM	DRAM
Fabricante	Dell	Intel	Smart
Capacidade (GB)	1000	375	24
Latência (ms)	$8,3$	$0,01$	$5 \cdot 10^{-7}$
Largura de Banda (GB/s)	1,2	2,4	17
Preço/bit (R$/GB)	1,99	20,27	43,75

Tabela 1. Características dos dispositivos de armazenamento.

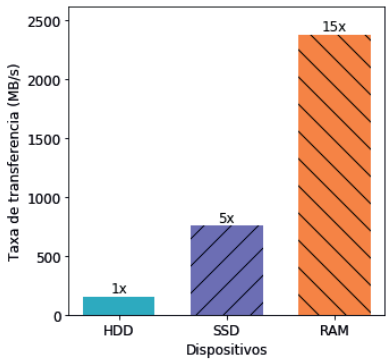


Figura 2. Taxa de transferência utilizando *dd*.

que fazem com que as partes mecânicas tenham um grande impacto no desempenho. Por outro lado, ressalta-se o ganho de desempenho desse dispositivo quando as operações são sequenciais. Essa diferença entre operações aleatórias e sequenciais não ocorre nos outros dispositivos. Por fim, a constante evolução dos dispositivos secundários permite que a vazão alcançada por eles se aproxime daquela alcançada pelas memórias principais.

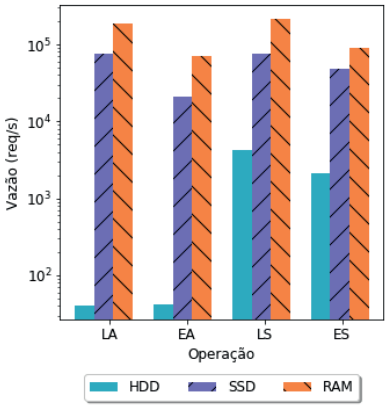


Figura 3. Vazão (IOPS) dos dispositivos obtidos por meio da ferramenta *fio*.

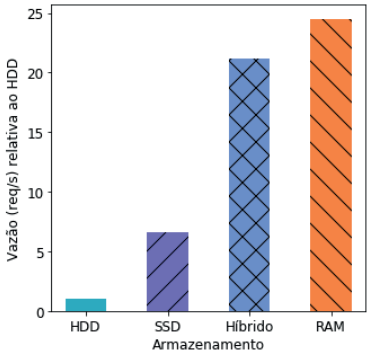


Figura 4. Vazão (req/s) em cada um dos dispositivos ao executar o TPC-C.

Visando responder ao primeiro ponto levantado no início desse trabalho, experimentamos a carga de trabalho padrão do TPC-C (i.e., porcentagem 45-43-4-4-4 de cada uma das transações) com tempo de *warm-up* de 5 segundos e fator de escala igual a 50 juntamente com a configuração padrão do PostgreSQL cujo resultado está apresentado nas Figuras 4 e 5. A Figura 4 mostra que a vazão do dispositivo de armazenamento subjacente pode acelerar em até 24 vezes o processamento de transações pelo PostgreSQL. Somente a troca do HDD pelo SSD já traz uma melhora de 7 vezes na vazão dos SGBDs. Conclui-se que a troca do dispositivo, sem mudanças na configuração do SBGD, já traz um ganho para o desempenho desses sistemas ao lidar com cargas de trabalho transacionais. Esse efeito também é visualizado na queda da latência das transações, conforme a Figura 5. Esta melhora se dá porque as transações do TPC-C realizam operações aleatórias de entrada e saída, característica marcante das cargas transacionais. Consequentemente, o desempenho é impactado pelos motivos apresentados acima ao utilizar o *fio*.

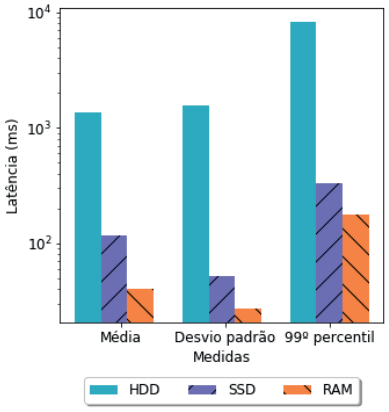


Figura 5. Medidas da latência (ms) em cada um dos dispositivos ao executar o TPC-C.

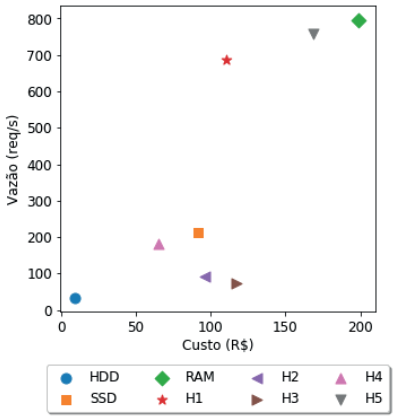


Figura 6. Relação custo-benefício do HDD, SSD e das formas híbridas.

4.2. Armazenamento Híbrido

O TPC-C é constituído por 9 relações e 5 tipos de transações. No armazenamento híbrido, os três dispositivos são usados conjuntamente de modo a realizar a divisão do banco de dados entre eles. Para tanto, utilizamos duas maneiras para decidir em qual dispositivo cada relação deveria ser armazenada, sendo que ambas seguiram uma distribuição uniforme, isto é, cada um dos dispositivos recebeu 3 relações. Enquanto que na primeira maneira é utilizado o tamanho das relações como fator de decisão, na segunda é utilizada a quantidade de operações de leitura e escrita que são realizadas nas relações. Cinco configurações (H1 - H5) foram geradas, sendo que H2 e H5 utilizaram a primeira forma de alocação e as restantes utilizaram a segunda. Como pode ser visto na Figura 4, a vazão alcançada pela configuração H1, a qual obteve o melhor custo-benefício, apresentou uma melhora em torno de 20 vezes quando comparado com a vazão do HDD. Nessa configuração, as relações TPC-C *Warehouse*, *District* e *Customer* foram alocadas na RAM, enquanto que as *Order-Line*, *Order* e *Stock* no SSD e, por fim, as *History*, *Item* e *New-Order* no HDD. Observa-se portanto que é possível construir um modelo híbrido de armazenamento de forma a considerar a carga de trabalho e ainda assim obter um bom desempenho.

4.3. Relação Custo-Benefício

Mesmo que a troca direta do HDD pela RAM traga melhoria significativa no desempenho do PostgreSQL ao se executar uma carga transacional, não se pode desconsiderar o custo adjacente relativo a essa troca. Muito embora as tecnologias de memória principal tenham evoluído constantemente ao longo dos anos e, dessa forma, melhorado o custo-benefício desse tipo de tecnologia de armazenamento, ainda existe uma diferença relevante quando se compara o custo dessas tecnologias. Por exemplo, no caso dos dispositivos utilizados nesse trabalho, essa diferença está em torno de 40 reais por cada GB. Com o advento das novas tecnologias de NVM, como a 3D XPoint aqui utilizada, essa lacuna diminui, mas ainda é considerável. A diferença entre o custo por GB do HDD para o SSD utilizado é de aproximadamente 18 reais. A Figura 6 mostra a relação entre o custo de manter o banco de dados e a vazão obtida, quer seja utilizando um dos dispositivos, quer seja realizando as várias formas híbridas descritas anteriormente. Observe as configurações híbridas H1 e H4. Com um custo 2 vezes menor e mantendo uma vazão bem próxima

do SSD, a configuração H4 apresenta uma melhor relação custo-benefício em relação ao SSD, podendo dessa forma ser uma alternativa viável. A configuração H1 obteve o maior ganho na relação custo-benefício, visto que essa apresentou um desempenho similar ao da RAM com um custo menor. Pegando como base essa configuração, temos que a relação custo-benefício do armazenamento híbrido seria de 6, enquanto que a RAM é de 4.

5. Conclusões e Trabalhos Futuros

Este trabalho apresenta uma análise experimental da influência que as diferentes tecnologias de armazenamento podem trazer para o desempenho de um SGBD relacional, PostgreSQL, quando são utilizadas como o hardware subjacente. Baseado nos resultados experimentais, discutimos os três questionamentos levantados nesse trabalho. Constatou-se que a rápida evolução das tecnologias de armazenamento traz uma melhoria significativa para os SGBDs. Em particular, pontuamos que a troca direta de HDD para DRAM causou um impacto positivo de 24 vezes na vazão de uma carga transacional. Não obstante, o custo dos dispositivos pode ser proibitivo. Assim, mostramos que é possível construir um armazenamento híbrido de tal forma a melhorar o custo-benefício em até 2 vezes em relação à RAM. Futuramente, pretendemos aprofundar o estudo com o intuito de propor uma solução autônoma para escolher a melhor organização dos dados no formato híbrido.

Agradecimentos

Esta pesquisa foi parcialmente apoiada pela CAPES (processo #1782887) e LSBD/UFC. Agradecemos ao Ítalo Cavalcante de Abreu pelo auxílio dado neste trabalho.

Referências

Brayner, A. and Monteiro, J. M. (2016). Hardware-aware database systems: A new era for database technology is coming - vision paper. In *31º Simpósio Brasileiro de Banco de Dados, SBBD 2016, Salvador, Bahia, Brasil, October 4-7, 2016.*, pages 187–192.

Difallah, D. E., Pavlo, A., Curino, C., and Cudre-Mauroux, P. (2013). Oltp-bench: An extensible testbed for benchmarking relational databases. *Proc. VLDB Endow.*, 7(4):277–288.

Leo Kelion, BBC, I. M. (2015). 3d xpoint technology. `https://www.bbc.com/news/technology-33675734`. Accessed: 2019-07-15.

Shah, M. A., Harizopoulos, S., Wiener, J. L., and Graefe, G. (2008). Fast scans and joins using flash drives. In *4th Workshop on Data Management on New Hardware, DaMoN 2008, Vancouver, BC, Canada, June 13, 2008*, pages 17–24.

Son, Y., Kang, H., Han, H., and Yeom, H. Y. (2016). An empirical evaluation and analysis of the performance of NVM express solid state drive. *Cluster Computing*, 19(3):1541–1553.

Xu, Q., Siyamwala, H., Ghosh, M., Suri, T., Awasthi, M., Guz, Z., Shayesteh, A., and Balakrishnan, V. (2015). Performance analysis of nvme ssds and their implication on real world databases. In *Proceedings of the 8th ACM International Systems and Storage Conference, SYSTOR 2015, Haifa, Israel, May 26-28, 2015*, pages 6:1–6:11.

Zukowski, M. (2009). Balancing vectorized query execution with bandwidth-optimized storage. *Journal of Computational Physics - J COMPUT PHYS*.

Unsupervised Rank Fusion for Diverse Image Metasearch*

José Solenir L. Figuerêdo[1], **Rodrigo Tripodi Calumby**[1]

[1] University of Feira de Santana – Feira de Santana – BA – Brazil

jslfigueredo@ecomp.uefs.br, rtcalumby@uefs.br

Abstract. *For a given query and a set of image ranked lists retrieved from multiple search engines, the metasearch technique aims at combining these lists to build an unified ranking with improved relevance. Rank aggregation is an approach that has been widely used to support this task. This paper investigates the use of rank aggregation methods in the metasearch scenario for diverse image retrieval. Although metasearch systems are usually driven by the relevance of the final result, the impact on diversification has also been analyzed. The experimental findings suggest metasearch based on rank aggregation allows significant improvements, both in terms of relevance and diversity.*

1. Introduction

In recent years, advances in data capture and storage technologies allowed the production of large amounts of digital content, enabling advanced studies in many fields. These collections have been explored in several contexts such as healthcare, biodiversity, social networks, and digital libraries [Bahri et al. 2019]. However, given the user needs and the large amount of information available, effective techniques are demanded to explore these collections. Over time, several methods have been proposed with the goal of generating better results for many information retrieval tasks.

In order to maximize the quality of the search results, the scientific and industrial communities developed robust systems to exploit as much information as possible for determining the relevance of objects in the databases [Calumby et al. 2016]. This allowed the improvement of ranking algorithms and consequently to better meet users' expectations in their search routines. However, given the complexity of the task, different systems tend to give different answers to the same information need of a given user. In this sense, the results achieved by each system tend to be complementary. A proposed solution to this scenario, known as metasearch, is the combination of results obtained from multiple databases or different search systems. This integration can be performed in many ways and applying rank aggregation algorithms is a popular approach for such task [McDonald and Smeaton 2005, Farah and Vanderpooten 2007].

In a more specific scenario, users may not be able to properly express their information need, leading to poorly specified or ambiguous queries [Santos et al. 2015]. Moreover, given a retrieval model considering only relevance maximization, systems eventually produce result lists with objects that are considerably similar (redundant) or do not necessarily include the different aspects from the items that are available in the collection (low coverage). An approach used to tackle these problems is the promotion of diversity into the result set and several algorithms have been proposed to fulfill this

*This work was partially supported by PIBIC/CNPq (grant number 158204/2018-2).

goal [Calumby et al. 2017]. The purpose of such algorithms is to maximize the information gain within the retrieved items and attenuate redundancy whilst responding to different interpretations from the same query. Hence, it allows simultaneously considering the query intents from different users.

Given the aforementioned challenges, this paper describes and investigates the use of rank aggregation methods for metasearch in the diverse image retrieval scenario. Although metasearch systems are usually focused on the relevance of the final result, in this paper we analyze the rank aggregation of diversified rankings and the impact on the relevance and the diversity of the final ranking.

2. Related Work

Rank aggregation algorithms can be divided in: score-based or order-based. In the former, the aggregation function takes as input the ranking scores associated to each object in the original rankings. In the latter, only the ordering among of items is required to perform the aggregation. Many score-based methods have been proposed, e.g., CombMAX, CombMIN, CombSUM, CombANZ, CombMNZ) [Muñoz et al. 2015]. In turn, Borda Count, Median Rank Aggregation, and Reciprocal Rank Fusion are popular order-based methods [Muñoz et al. 2015]. These algorithms have been applied in many contexts including metasearch. However, these methods do not consider diversification explicitly. Nevertheless, some studies have investigated the application of data fusion for diversification tasks. In particular, as stated in [Wu et al. 2019], data fusion is expected to ensure a broader coverage of different types of relevant documents fostering diverse promotion.

In [Liang et al. 2014] the diversification includes three steps. Initially, the aggregation relies on traditional methods: CombSUM and CombMNZ. Hence, an inference is made for latent subtopics. Finally, the result generated by the fusion and topic modeling steps is submitted to diversification. Alternatively, in [Xu and Wu 2017] rather than fusing results that are already diversified, an early fusion approach is applied. It consists in: i) Considering only the relevance, a set of results are generated with algorithms of typical searches; ii) The results are combined with a fusion algorithm such as CombMNZ; and iii) An explicit diversification method is applied, e.g., the xQuAD. The experimental findings indicated that the early fusion strategy was as effective as late fusion ones.

Previous works focused on the analysis of diversification through fusion methods in the context web page retrieval. The investigation of such methods in other multimedia scenarios (e.g., images or videos) are still incipient. Beyond it, the image retrieval imposes additional challenges due to the inherent characteristics of the tasks and the heterogeneity of data collections. This work experimentally investigates the effectiveness of several rank aggregation methods for metasearch in the context of diverse image retrieval.

3. Evaluation Scenario and Experimental Setup

For the experimental evaluation, the collection provided by the *Information Fusion for Social Image Retrieval & Diversification Task* [Ramírez-de-la-Rosa et al. 2018] was used. It includes results from the many image search systems proposed and evaluated between 2013 and 2016 in the *MediaEval Retrieving Diverse Social Images tasks* [Ionescu et al. 2014, Ionescu et al. 2015, Ionescu et al. 2016a,

Ionescu et al. 2016b]. There are ranked results for numerous queries. Moreover, it includes relevant and diverse results with different levels of quality. The dataset is organized in development, validation and test sets (Table 1). The test set was not considered in the experimental evaluation, since ground-truth was not publicly available.

Table 1. Overview of the collection used in the experimental evaluation.

Dataset		# Queries	# Rankings	Topic Category
Devset	devset1	346	39	Single-topic
	devset2	60	56	Single-topic
Validset		139	60	Single/Multi-topic
Testset	seenIR	63	56	Single-topic
	unseenIR	64	29	Multi-topic

Many rank aggregation methods were considered. The score-based methods evaluated were CombMAX, CombMIN, CombSUM, CombANZ, CombMNZ, CombMED, and Multiplication Scores (MScores) [Li et al. 2014]. In turn, the order-based methods Borda Count, Median Rank Aggregation (MRA), and Reciprocal Rank Fusion (RRF) were assessed.

Precision and Cluster-Recall [Zhai et al. 2003] measures were used for effectiveness assessment. Precision represents the quality of the ranking in terms of relevance and Cluster-Recall computes the percentage of conceptual clusters that were represented in the final diversified result. These metrics were computed based on the available ground-truth. For effectiveness analysis these measures were computed for up to the 50th position of the ranking. As the baseline, in addition to the one provided by [Ramírez-de-la-Rosa et al. 2018], we also considered the best input ranking systems.

For the comparison to the baseline, we computed the relative gain of the best aggregation method for each rank depth. The selection of the best systems to be used as baselines relied on the Precision (PR@20) e Cluster-Recall(CR@20). The cutoff at 20 simulates the content of a single page of a typical web image search engine and reflects user behavior, i.e., inspecting the first page of results [Ramírez-de-la-Rosa et al. 2018].

4. Results and Discussion

The comparative analysis against the baselines was performed based on the relative gains at multiple ranking depths. Initially, Table 2 shows the effectiveness of the fusion methods, including the baselines. The highest values are highlighted in boldface. These values are used for the comparison with the baselines. Considering the aggregation methods, RRF was the top performing one. It was not the most frequent top performer for Devset1. However, for the Deveset2 and the Validset it outperformed the other methods for most of the evaluation measures and ranking depths. For the Devset2, the RRF method, besides improving the diversity, did not negatively impact the relevance of the results, which is a desirable behaviour for diverse image retrieval systems.

Table 3 presents the comparison between the fusion methods (taking the highest values) and the baselines. It indicates the relative gains of the fusion methods over the baselines (positive gains are highlighted in bold). In Table 3, *ICPR* stands for the baseline provided with the collection as part ICPR challenge [Ramírez-de-la-Rosa et al. 2018]. In turn, *Best_P* and *Best_CR* represent the best input system considering Precision and Cluster-Recall as the selection criteria, respectively.

Table 2. Results for the rank aggregation methods and baselines. Top values are highlighted in boldface.

Devset1												
Method	P@5	P@10	P@20	P@30	P@40	P@50	CR@5	CR@10	CR@20	CR@30	CR@40	CR@50
Baseline(ICPR)	0.7883	0.7558	0.7289	0.7194	0.7080	0.6877	0.2331	0.3649	0.5346	0.6558	0.7411	0.7988
Baseline(best_P)	0.8947	0.8936	0.8788	0.8569	0.8265	0.7891	0.2047	0.2963	0.4410	0.5476	0.6416	0.7122
Baseline(best_CR)	0.7409	0.7330	0.7487	0.7603	0.7145	0.5915	0.2614	0.4291	0.6314	0.7228	0.7473	0.7484
CombMAX	0.7602	0.7512	0.7512	0.7349	0.7242	0.7031	0.2531	0.4176	0.6069	0.7315	0.8158	0.8639
CombMIN	0.6544	0.6626	0.6744	0.6790	0.6736	0.6582	0.2049	0.3580	0.5455	0.6822	0.7647	0.8201
CombSUM	0.8287	0.8243	0.8034	0.7867	0.7626	0.7315	0.2694	0.4410	0.6255	**0.7437**	**0.8261**	0.8738
CombANZ	0.7573	0.7561	0.7469	0.7347	0.7205	0.6971	0.2500	0.4020	0.5905	0.7234	0.8073	0.8552
CombMED	0.8287	0.8243	0.8034	0.7867	0.7626	0.7315	0.2694	**0.4410**	0.6255	0.7437	0.8261	0.8738
CombMNZ	0.8456	0.8289	**0.8098**	**0.7888**	**0.7645**	**0.7344**	0.2753	0.4370	**0.6258**	0.7412	0.8238	0.8708
MScores	0.8240	0.8228	0.8044	0.7870	0.7624	0.7313	0.2672	0.4383	0.6235	0.7413	0.8235	**0.8745**
Borda Count	0.6129	0.6143	0.6213	0.6256	0.6246	0.6150	0.1884	0.3090	0.4642	0.5883	0.6780	0.7437
RRF	**0.8491**	**0.8316**	0.8092	0.7874	0.7616	0.7314	**0.2781**	0.4327	0.6235	0.7421	0.8203	0.8655
MRA	0.7614	0.7567	0.7361	0.7262	0.7092	0.6872	0.2544	0.4157	0.6168	0.7351	0.8186	0.8690
Devset2												
Baseline(ICPR)	0.8100	0.8067	0.8058	0.8056	0.8025	0.7917	0.1316	0.2135	0.3435	0.4517	0.5364	0.5977
Baseline(best_P)	0.8933	0.8933	0.8550	0.8167	0.8021	0.7850	0.1461	0.2377	0.3809	0.4750	0.5531	0.6210
Baseline(best_CR)	0.8867	0.8600	0.8492	0.8189	0.8017	0.7933	0.1682	0.3003	0.4697	0.5594	0.6274	0.6788
CombMAX	0.7467	0.7417	0.7558	0.7639	0.7333	0.7257	0.1378	0.2534	0.4209	0.5375	0.6186	0.6725
CombMIN	0.4667	0.5000	0.5025	0.5189	0.5312	0.5463	0.0878	0.1663	0.2804	0.3914	0.4790	0.5565
CombSUM	0.8667	0.8550	0.8267	0.8194	0.8075	0.7980	0.1650	0.2861	0.4430	0.5579	0.6390	0.7046
CombANZ	0.4733	0.5433	0.5808	0.5944	0.6096	0.6147	0.0917	0.1882	0.3374	0.4480	0.5427	0.6045
CombMED	0.8667	0.8550	0.8267	0.8194	0.8075	0.7980	0.1650	0.2861	0.4430	0.5579	0.6390	0.7046
CombMNZ	0.8933	0.8650	0.8417	0.8361	**0.8292**	0.8123	0.1685	0.2820	0.4525	**0.5692**	**0.6417**	0.7041
MScores	0.8733	0.8517	0.8308	0.8250	0.8075	0.8013	0.1665	0.2797	0.4433	0.5616	0.6362	0.7018
Borda Count	0.4700	0.4567	0.4750	0.4889	0.4925	0.4993	0.0885	0.1541	0.2694	0.3665	0.4238	0.4948
RRF	**0.8967**	**0.8817**	**0.8508**	**0.8372**	**0.8292**	**0.8260**	**0.1692**	**0.2906**	**0.4586**	0.5676	0.6367	**0.7052**
MRA	0.6633	0.6733	0.6775	0.6733	0.6725	0.6797	0.1287	0.2370	0.4066	0.5185	0.6141	0.6831
Validset												
Baseline(ICPR)	0.7281	0.7086	0.7000	0.6952	0.6838	0.6776	0.1489	0.2402	0.3684	0.4616	0.5284	0.5851
Baseline(best_P)	0.8101	0.8108	0.7906	0.7736	0.7646	0.7534	0.1904	0.2908	0.4051	0.5005	0.5703	0.6246
Baseline(best_CR)	0.7755	0.7633	0.7309	0.7002	0.6899	0.6790	0.1935	0.3163	0.4963	0.6112	0.6933	0.7514
CombMAX	0.7439	0.7209	0.7065	0.7041	0.7007	0.7014	0.1658	0.2704	0.4207	0.5230	0.6084	0.6716
CombMIN	0.5597	0.5813	0.5968	0.5986	0.6031	0.6029	0.1188	0.2019	0.3282	0.4152	0.4876	0.5440
CombSUM	0.7626	0.7554	0.7320	0.7271	0.7214	0.7173	0.1757	0.2812	0.4256	0.5404	0.6246	0.6875
CombANZ	0.6230	0.6201	0.6399	0.6441	0.6550	0.6544	0.1390	0.2274	0.3755	0.4639	0.5436	0.5956
CombMED	0.7626	0.7554	0.7320	0.7271	0.7214	0.7173	0.1757	0.2812	0.4256	0.5404	0.6246	0.6875
CombMNZ	0.7683	0.7662	0.7471	0.7331	0.7243	0.7219	0.1772	0.2893	0.4344	0.5494	0.6326	0.6951
MScores	0.7612	0.7597	0.7367	0.7254	0.7212	0.7168	0.1760	0.2825	0.4298	0.5391	0.6254	0.6861
Borda Count	0.5669	0.5727	0.5770	0.5878	0.5946	0.5994	0.1208	0.2049	0.3161	0.3972	0.4643	0.5191
RRF	**0.7813**	**0.7698**	**0.7536**	**0.7405**	**0.7318**	**0.7281**	**0.1776**	**0.3008**	**0.4570**	**0.5596**	**0.6388**	**0.7025**
MRA	0.6619	0.6590	0.6680	0.6743	0.6761	0.6753	0.1546	0.2641	0.4085	0.5310	0.6119	0.6744

Table 3. Relative gains (%) of top performing methods against the baselines.

Devset1												
	P@5	P@10	P@20	P@30	P@40	P@50	CR@5	CR@10	CR@20	CR@30	CR@40	CR@50
Gain Over ICPR	**7.71**	**10.03**	**11.10**	**9.65**	**7.98**	**6.79**	**19.31**	**20.86**	**17.06**	**13.40**	**11.47**	**9.48**
Gain Over Best_P	-5.10	-6.94	-7.85	-7.95	-7.50	-6.93	**35.86**	**48.84**	**41.90**	**35.81**	**28.76**	**22.79**
Gain Over Best_CR	**14.60**	**13.45**	**8.16**	**3.75**	**7.00**	**24.16**	**6.39**	**2.77**	-0.89	**2.89**	**10.54**	**16.85**
Devset2												
Gain Over ICPR	**10.70**	**9.30**	**5.58**	**3.92**	**3.33**	**4.33**	**28.57**	**36.11**	**33.51**	**26.01**	**19.63**	**17.99**
Gain Over Best_P	**0.38**	-1.30	-0.49	**2.51**	**3.38**	**5.22**	**15.81**	**22.25**	**20.40**	**19.83**	**16.02**	**13.56**
Gain Over Best_CR	**1.13**	**2.52**	**0.19**	**2.23**	**3.43**	**4.12**	**0.59**	-3.23	-2.36	**1.75**	**2.28**	**3.89**
Validset												
Gain Over ICPR	**7.31**	**8.64**	**7.66**	**6.52**	**7.02**	**7.45**	**19.27**	**25.23**	**24.05**	**21.23**	**20.89**	**20.06**
Gain Over Best_P	-3.56	-5.06	-4.68	-4.28	-4.29	-3.36	-6.72	**3.44**	**12.81**	**11.81**	**12.01**	**12.47**
Gain Over Best_CR	**0.75**	**0.85**	**3.11**	**5.76**	**6.07**	**7.23**	-8.22	-4.90	-7.92	-8.44	-7.86	-6.51

Considering the Devset1, there were relative gains for most of the ranking depths considered. Beyond it, gains over *ICPR* occurred at all ranking depths. For the *Best_P*, there was no gain on Precision. However, for this system the gains in terms of diversity were quite expressive, with gains above 20% for all observed depths. Moreover, for the *Best_CR*, there were gains in both relevance and diversity. Overall, the highest gains were achieved in the top positions of the ranking, except for the *Best_CR*, with the fusion methods achieving higher gains at deeper levels.

Regarding the Devset2, there were positive gains against all baselines for most of the considered depths. It indicates that when querying using the metasearch approach, the user obtained more relevant and diverse results. Similar to Devset1, taking *ICPR* and *Best_CR* baselines, the highest gains were achieved at the beginning and at the end of the rankings, respectively. In turn, the highest gains over *Best_P* occurred at the end of the ranking for Precision and at the beginning of the ranking for Cluster-Recall. Notice, specially considering the *ICPR* and *Best_P* baselines, that the fusion methods achieved higher gains in terms of diversity and maintained acceptable relevance.

The gains in the Validset were not expressive, except over the *ICPR* baseline. For the other baselines the gains achieved were unidirectional, that is, for *Best_P* there is no gain in Precision, but only in Cluster-Recall. For the *Best_CR* there was no gain considering Cluster-Recall, but only in Precision. This may be a consequence of the characteristics of the Validset, which unlike the Devsets (with only single-topic queries) also contains multi-topic queries. Hence, it demands further investigations of this challenge and the development of suitable rank fusion methods.

5. Conclusions

This paper described and investigated the use of rank aggregation methods in the metasearch scenario, considering both the impact on relevance and diversification. Our experimental results suggest that fusion methods tend to allow better search results than independent systems. In addition, it was observed that the greatest gains were in terms of diversity, although there were gains in terms of relevance as well. Our findings validated the idea that metasearch systems may allow improvements in the diversity of the results.

It was also found that some fusion methods were flexible enough to improve one objective while maintaining a competitive performance for the other. For some cases, in addition to achieving high gains in terms of diversity, the fusion methods also maintained acceptable results in relevance. However, we noticed that for the multi-topic queries the metasearch did not outperform the best individual system. This highlights the demand of new investigations for the development of novel rank fusion methods able to enhance both relevance and diversity for the case of multi-topic queries.

References

Bahri, S., Zoghlami, N., Abed, M., and Tavares, J. M. R. S. (2019). Big data for healthcare: A survey. *IEEE Access*, 7:7397–7408.

Calumby, R. T., Gonçalves, M. A., and da Silva Torres, R. (2016). On interactive learning-to-rank for IR: overview, recent advances, challenges, and directions. *Neurocomputing*, 208:3–24.

Calumby, R. T., Gonçalves, M. A., and da Silva Torres, R. (2017). Diversity-based interactive learning meets multimodality. *Neurocomputing*, 259:159–175.

Farah, M. and Vanderpooten, D. (2007). An outranking approach for rank aggregation in information retrieval. In *SIGIR'07, Amsterdam, The Netherlands, July 23-27*, pages 591–598.

Ionescu, B., Gînsca, A., Boteanu, B., Lupu, M., Popescu, A., and Müller, H. (2016a). Div150multi: a social image retrieval result diversification dataset with multi-topic queries. In *MMSys'16, Klagenfurt, Austria, May 10-13*, pages 46:1–46:6.

Ionescu, B., Gînsca, A., Zaharieva, M., Boteanu, B., Lupu, M., and Müller, H. (2016b). Retrieving diverse social images at mediaeval 2016: Challenge, dataset and evaluation. In *MediaEval'16 Workshop, Hilversum, The Netherlands, October 20-21*.

Ionescu, B., Popescu, A., Lupu, M., Gînsca, A., Boteanu, B., and Müller, H. (2015). Div150cred: A social image retrieval result diversification with user tagging credibility dataset. In *MMSys'15, Portland, USA, March 18-20*, pages 207–212.

Ionescu, B., Radu, A., Menéndez, M., Müller, H., Popescu, A., and Loni, B. (2014). Div400: a social image retrieval result diversification dataset. In *MMSys'14, Singapore, Mar 19-21*, pages 29–34.

Li, L. T., Pedronette, D. C. G., Almeida, J., Penatti, O. A. B., Calumby, R. T., and Torres, R. d. S. (2014). A rank aggregation framework for video multimodal geocoding. *Multimedia Tools and Applications*, 73(3).

Liang, S., Ren, Z., and de Rijke, M. (2014). Fusion helps diversification. In *SIGIR'14, NY, USA*, pages 303–312. ACM.

McDonald, K. and Smeaton, A. F. (2005). A comparison of score, rank and probability-based fusion methods for video shot retrieval. In *CIVR'05, Singapore, July 20-22, Proceedings*, pages 61–70.

Muñoz, J. A. V., da Silva Torres, R., and Gonçalves, M. A. (2015). A soft computing approach for learning to aggregate rankings. In *CIKM'15, Melbourne, Australia, October 19 - 23*, pages 83–92.

Ramírez-de-la-Rosa, G., Villatoro, E., Ionescu, B., Escalante, H. J., Escalera, S., Larson, M., Müller, H., and Guyon, I. (2018). Overview of the multimedia information processing for personality & social networks analysis contest. In *ICPR'18, Beijing, China, August 20-24, Revised Selected Papers*, pages 127–139.

Santos, R. L. T., MacDonald, C., and Ounis, I. (2015). Search result diversification. *Foundations and Trends in Information Retrieval*, 9(1):1–90.

Wu, S., Huang, C., Li, L., and Crestani, F. (2019). Fusion-based methods for result diversification in web search. *Information Fusion*, 45:16–26.

Xu, C. and Wu, S. (2017). The early fusion strategy for search result diversification. In *ACM TUR-C'17, New York, USA*, pages 47:1–47:6.

Zhai, C. X., Cohen, W. W., and Lafferty, J. (2003). Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *ACM SIGIR, Toronto, Canada*, pages 10–17.

Differentially Private Group-by Data Releasing Algorithm

Iago Chaves[1], **Javam Machado**[1]

[1]Database and Systems Laboratory – Federal University of Ceará – Brazil

Abstract. *Privacy concerns are growing fast because of data protection regulations around the world. Many works have built private algorithms avoiding sensitive information leakage through data publication. Differential privacy, based on formal definitions, is a strong guarantee for individual privacy and the cutting edge for designing private algorithms. This work proposes a differentially private group-by algorithm for data publication under the exponential mechanism. Our method publishes data groups according to a specified attribute while maintaining the desired privacy level and trustworthy utility results.*

1. Introduction

Nowadays privacy is a trending topic for consumers and companies. For companies, privacy concern is growing fast because of data protection regulations around the world. And for the consumers, since companies are leaking information about individuals, such as Netflix leakage [Harmanci and Gerstein 2016]. To tackle privacy issues, the differential privacy was proposed [Dwork 2011]. Differential privacy is a strong and measurable guarantee that some method or mechanism do not leak information, depicted by equation 1. Intuitively, it ensures that the presence or absence of someone in the dataset does not change the answer of a query at all. In equation 1, $\mathcal{A}$ represents a mechanism, $o \in \mathcal{O}$ some possible output, ε symbolizes the privacy budget and x, y any neighboring datasets. The definition of neighboring datasets is that two datasets differ only in one user, so $x, y : |x-y| \leq 1$. The differential privacy has been made to run interactively e.g. average, count, sum. Someone submits a query to a database, and it returns via anonymization algorithms an answer with noise [Mendonça et al. 2017]. Occasionally, it is necessary to publish the data instead of statistics about it [Chen et al. 2011].

$$Pr[\mathcal{A}(x) = o] \leq \exp^{\varepsilon} Pr[\mathcal{A}(y) = o] \tag{1}$$

$$Pr[\mathcal{A}(x) = o] \propto \exp(\frac{\varepsilon u(x, o)}{2\Delta u}) \tag{2}$$

$$\Delta u = \max_{o \in \mathcal{O}} \max_{x,y:|x-y|\leq 1} |u(x, o) - u(y, o)| \tag{3}$$

A widely known differential privacy mechanism is the exponential mechanism [McSherry and Talwar 2007]. It works well with categorical answers. The set of all possible query answers is $\mathcal{O}$. The utility function u maps each dataset element, i.e. a user, and a possible result to a score number. The probability of the exponential mechanism outputs o as the answer with utility function u applied in the dataset x is showed in equation 2. But it is necessary to calculate the global sensitivity of Δu, which represents the highest impact regarding the presence or absence of someone in the data.

This paper proposes two methods for grouped data releasing under differential privacy framework. The first algorithm tries to find the optimal representation of the actual group but suffers from computational complexity. The second one reduces the space

search of the former method by a heuristic and achieves reliable results. The PINQ platform [McSherry 2009] address a similar problem, however only to statistical aggregation queries. In section 2 we introduce the two proposed methods TRIODE and TRIODE-H, afterward the experimental results, and finally the conclusion.

2. Our Method

Our goal is to proposes a privacy-preserving method that group a dataset by an attribute and release the groups. We intend to release data through a privacy-preserving group-by algorithm. In other words, a group-by query q response over the dataset $\mathcal{D}$ must do not leak information about individuals. To achieve our goal, we propose two methods via differential privacy mechanisms.

Our first method, named as TRIODE[1], finds each attribute-group applying the exponential mechanism. It is necessary to evaluate the score of all possible subsets, commonly named as powerset of $\mathcal{D}$ and denoted by $\mathbb{P}(\mathcal{D})$. The $\mathbb{P}(\mathcal{D})$ has cardinality $2^{|\mathcal{D}|}$. The second proposed method is TRIODE-H, a TRIODE-based approach that implements a new heuristic to assess the scores, circumventing the powerset complexity problem.

2.1. TRIODE

The TRIODE method groups the dataset by attributes and their values. In order to perform it in a differentially private manner, our method applies the exponential mechanism. So, our dataset $\mathcal{D}$ is composed by $\{a_0, a_1, \ldots, a_k\}$ categorical attributes, and for each attribute a_i, such that $0 \leq i \leq k$, it has κ_i categorical values, represented by $C_i = \{c_i^0, \ldots, c_i^{\kappa_i}\}$. When the group-by query is over the a_i attribute, the method response must contain $|C_i|$ groups.

As aforementioned, a score function $u : \mathcal{D} \times R \to \mathbb{R}$ maps an individual from the dataset and a possible result to a score. The TRIODE answers set are all possibles subsets of $\mathcal{D}$, since a group can be empty or the entire dataset either. The answers set size grows exponentially with the dataset size, once $|R| = 2^{|\mathcal{D}|}$. It is necessary to evaluate the score of each possible category a_i separately. The score must measure a possible result $r \in R$ of a query q grouping over the attribute-value c_i^j concerning the ground truth grouped set $\mathcal{D}_{c_i^j} \subseteq \mathcal{D}$, where $0 \leq j \leq \kappa_i$. Consequently, an answer $r \in R$ with higher similarity with $\mathcal{D}_{c_i^j}$ must imply in a higher score.

$$u_{c_i^j}(r, \mathcal{D}) = 2 \times \frac{|r \cap \mathcal{D}_{c_i^j}|}{|r| + |\mathcal{D}_{c_i^j}|} \tag{4}$$

The score function is defined by equation 4. The fraction numerator is the number of matches between two sets, and the denominator represents the cardinality sum of the sets. Moreover, we need to calculate the global sensitivity of the utility function:

$$GS_u = \max_{\mathcal{D}, \mathcal{D}', r \in R : |\mathcal{D} - \mathcal{D}'| \leq 1} |u(r, \mathcal{D}) - u(r, \mathcal{D}')| \tag{5}$$

$$\leq \frac{2}{2|\mathcal{D}| - 1} \tag{6}$$

[1] Acronym for differen*T*ially p*RI*vate gr*O*up-by *D*ata r*E*leasing

From equation 5 to 6 we assume that $|r|$ is at most $|\mathcal{D}|$, since $r \in \mathbb{P}(\mathcal{D})$. Once the scores and global sensitivity were calculated, it is possible to get the group conforming to the exponential mechanism. The method TRIODE is ε-DP.

2.2. TRIODE-H

The measurement task of all scores in $\mathbb{P}(\mathcal{D})$ is a computationally tough task. For this purpose, we designed a TRIODE-based method, called TRIODE-H. The core concept behind TRIODE-H is to firstly reduce the search space using a heuristic to achieve results within larger datasets, timely. Furthermore, we find the group length through differential privacy to build the groups.

To prune our search space we randomly split our dataset $\mathcal{D}$ into m disjoint fragments, $\{F_1, \ldots, F_m\}$, which each fragment has a fixed length h. Evaluating it with the score function $\pi : F_v \times T_v \to \mathbb{R}$ for all subsets, where $0 \leq v \leq m$ and $T_v = \mathbb{P}(F_v)$ is the set of all possible results for F_v. It is defined as:

$$\pi_{c_i^j}(t, F_v) = -\left|\ |F_v^{c_i^j}| - |t|\ \right| \tag{7}$$

Where $F_v^{c_i^j}$ represents the group F_v grouped by attribute C_i with a value equal to c_i^j and $t \in T_v$. Now we are capable to calculate the score $s : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ for each individual $d \in \mathcal{D}$ separately. For this, we define the individual-based scoring function in equation 8, which calculates the sum of scores, provided by π, for all subsets $\{g \subseteq \mathbb{P}(F_v) \,|\, d \in g\}$.

$$s_{c_i^j}(d, F_v) = \sum_{g \in \mathbb{P}(F_v)} \psi(d, g, \pi_{c_i^j}(g, F_v)) \tag{8}$$

$$\psi(d, g, \pi) = \begin{cases} \pi & \text{if } d \in g \\ -|g| & \text{otherwise} \end{cases} \tag{9}$$

It is important to notice that the $s_{c_i^j}(d, F_v)$ index describes the score function for the group $F_v^{c_i^j}$. Now it is necessary to compute the global sensitivity of s:

$$\begin{aligned} GS_s &= \max_{F_v, F_v', d \in F_v : |F_v - F_v'| \leq 1} |s(d, F_v) - s(d, F_v')| && (10)\\ &\leq |F_v| \times 2^{|F_v|-1} && (11)\\ &= h \times 2^{h-1} && (12) \end{aligned}$$

From equation 10 to 11, we explore that $F_v' = F_v \setminus d$ where $d \in F_v$ is an individual, and $\mathbb{P}(F_v') = 2^{|F_v|-1}$. Now each individual from $\mathcal{D}$ has a score for c_i^j, so we need to publish its group. To do this, it is necessary to discover the size of $\mathcal{D}_{c_i^j}$ in a private manner. Therefore, we defined the set of possible answers for $|\mathcal{D}_{c_i^j}|$ as $L = \{0, \ldots, |\mathcal{D}|\}$ and a score function $\lambda : \mathcal{D} \times L \to \mathbb{R}$.

$$\lambda(\mathcal{D}, \ell) = -|\ |\mathcal{D}| - \ell\ | \tag{13}$$

$$GS_\lambda = \max_{\mathcal{D}, \mathcal{D}', \ell \in L : |\mathcal{D} - \mathcal{D}'| \leq 1} |\lambda(\mathcal{D}, \ell) - \lambda(\mathcal{D}', \ell)| \leq 1 \tag{14}$$

Algorithm 1: 2ε-TRIODE-H

Input: $\mathcal{D}$, $c_i^j \in C_i$, 2ε
Result: group $\tilde{\mathcal{D}}_{c_i^j}$ published by the differentially private method
$\{F_1, \ldots, F_m\} \leftarrow$ split($\mathcal{D}$)
foreach $F_v \in \{F_1, \ldots, F_m\}$ **do**
  $T_v \leftarrow \mathbb{P}(F_v)$
  foreach $t \in T_v$ **do**
    foreach $d \in F_v$ **do**
      scores[d] $\leftarrow s(d, F_v)$
$L = \{0, \ldots, |\mathcal{D}|\}$
foreach $\ell \in L$ **do**
  scores$_L[\ell] \leftarrow \lambda(\mathcal{D}_{c_i^j}, \ell)$
$\ell_{c_i^j} \leftarrow$ ExponentialMechanism($\mathcal{D}$, scores$_L$, ε, $GS = 1$)
foreach $u \in 0, \ldots, \ell_{c_i^j}$ **do**
  $\tilde{\mathcal{D}}_{c_i^j}[u] \leftarrow$ ExponentialMechanism($\mathcal{D}$, scores, $\varepsilon/\ell_{c_i^j}$, $GS = h \times 2^{h-1}$)
return $\tilde{\mathcal{D}}_{c_i^j}$

Where $\ell \in L_{c_i^j}$ and to proof the $GS_\lambda \leq 1$ we used the triangle inequality. Once the scores and global sensitivity were calculated, it is possible to get the group size $\ell_{c_i^j}$ conforming to the exponential mechanism. After all, we are capable to discover the group of C_i with a value equal to c_i^j via private way, i.e. the set $\tilde{\mathcal{D}}_{c_i^j}$. To do this, it is necessary get one individual $\ell_{c_i^j}$ times from $\mathcal{D}$ via exponential mechanism with privacy budget $\frac{\varepsilon}{\ell_{c_i^j}}$ to populate $\tilde{\mathcal{D}}_{c_i^j}$. The algorithm 1 shows clearly the necessary flow for achieving a diferentially private group-by data release. Finally, we reached that the TRIODE-H is 2ε-differential private. The ε budget for querying the group length, and ε for create the group via sequential composition [McSherry 2009].

3. Results

Our experimented data are the Adult Dataset from UCI Machine Learning Repository [Dua and Graff 2017] with 48,842 real individuals and 14 attributes. In this data, for this work, we are interested only in the "sex" attribute, that has two possible value ''Female" and "Male". When the data are grouped, the size of the ''Female" group is 16,192 and so, the size of the "Male" group is 32,650.

The chosen metrics are Precision, Recall, and F1-score [Powers 2011]. The precision is defined by the number of true positives divided by the sum of true positives with false positives. Precision tells us what proportion of our private data are actually relevant. The recall is the number of true positives divided by the sum of true positives with false negatives, which expresses the proportion of actual positive cases that are in the private answer. Finally, the F1-scores is the harmonic mean of precision and recall: F1-score $= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. The TRIODE method explained in section 2.1, has high computational complexity due to the calculation of $\mathbb{P}(\mathcal{D})$. Besides, the dataset size for this method was set as 20 individuals randomly chosen. Some experiments with 25 and 30 individuals lead to out of memory error, and as consequence, the dataset size is limited

to 20 persons.

All the experiments were made with 5 well-known privacy budgets: $\{0.01, 0.1, ln(2), 1, ln(3), 10\}$ [Dwork 2008]; and for each privacy budget, the experiment was run 10 times. The privacy budgets tries to represent environments from high to low privacy regime, where $\varepsilon = 0.01$ is a strong privacy parameter, and $\varepsilon = 10$ is a weak privacy guarantee. The ε controls the trade-off between utility and privacy. The higher ε more utility, and less privacy you have.

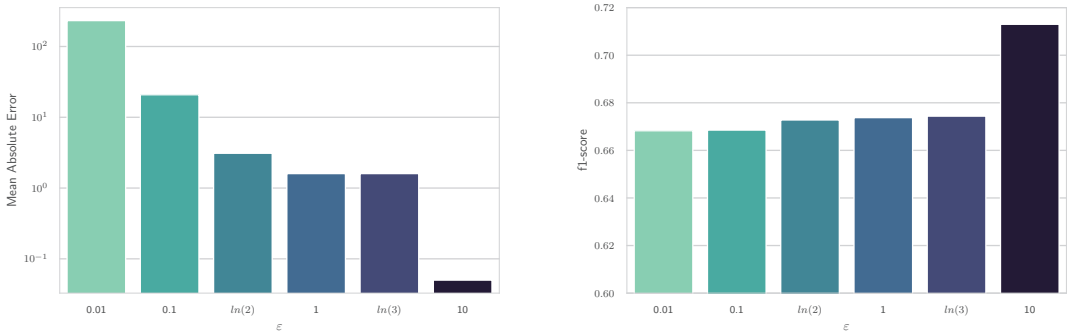
ε	Precision	Recall	F1-score
0.0100	0.7625	0.8133	0.7871
0.1000	0.7438	0.7933	0.7677
$ln(2)$	0.7500	0.8000	0.7742
1.0000	0.7812	0.8333	0.8065
$ln(3)$	0.7750	0.8267	0.8000
10.0000	0.9375	1.0000	0.9677

Table 1. TRIODE average results for the dataset with the length of 20 persons

ε	Precision	Recall	F1-score
0.0100	0.6683	0.6682	0.6682
0.1000	0.6686	0.6685	0.6685
$ln(2)$	0.6728	0.6727	0.6728
1.0000	0.6738	0.6737	0.6738
$ln(3)$	0.6744	0.6743	0.6744
10.0000	0.7131	0.7129	0.7130

Table 2. TRIODE-H average results applied to the entire Adult dataset

Table 1 shows the results for the query: $q =$ `FROM` $\overline{\mathcal{D}}$ `GROUP BY sex`; where $\overline{\mathcal{D}}$ is the dataset $\mathcal{D}$ with only 20 individuals, randomly chosen. Thus, we achieved good and trustworthy results. For the strict budget $\varepsilon = 0.01$ the mean of F1-scores is ≈ 0.79, and for $\varepsilon = 10$ the average F1-score is ≈ 0.97. This is a fine result, but the TRIODE bottleneck is the complexity (time and space), making the experiment with more data unfeasible.



(a) Mean Absolute Error from finding the group size

(b) The average F1-score for the group answer

Figure 1. TRIODE-H results applied to the entire dataset

For TRIODE-H method, the privacy budgets and the query q was the same used in TRIODE experiments. The query q is over the entire data $\mathcal{D}$. Firstly, was necessary to differentially private choose the group length. Figure 1a shows the mean absolute error for each privacy budget. It is worth mentioning that the mean absolute error axis is in logarithmic scale. Once the group length is known, we can settle the answer for query

q via the TRIODE-H technique. The dataset $\mathcal{D}$ has 48,842 rows, and it was split into disjoint fragments with size $m = 10$. The experiments with TRIODE-H was shown in Table 2 and Figure 1 shows the average F1-score for each ε.

4. Conclusion

In this work, we proposed two differentially private data releasing techniques for group-by queries. The first technique TRIODE achieved good utility results. However, the lack of scalability makes the method unfeasible for a large amount of data. The second method TRIODE-H addressed the scalability deficiency by splitting the entire dataset into small fragments. The TRIODE-H was performed with the complete dataset $\mathcal{D}$, and it attains quality results with small privacy budget. For future works we expect to measure the utility bounds for TRIODE and TRIODE-H and to find the optimal, or nearly optimal, space search for TRIODE; It might be valable to experiment with attributes with more than 2 possible classes and also to define an optimization problem to find the optimal set of parameters.

Acknowledgments

Thank you André Mendonça, Daniel Praciano, Eduardo Duarte and Israel Vidal. This research was partially supported by Lenovo Brasil, as part of its R&D investment under Brazil's Informatics Act, CAPES, LSBD/UFC.

References

Chen, R., Mohammed, N., Fung, B. C., Desai, B. C., and Xiong, L. (2011). Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, 4(11):1087–1098.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Dwork, C. (2008). Differential privacy: A survey of results. In Agrawal, M., Du, D., Duan, Z., and Li, A., editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dwork, C. (2011). Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340.

Harmanci, A. and Gerstein, M. (2016). Quantification of private information leakage from phenotype-genotype data: linking attacks. *Nature methods*, 13(3):251.

McSherry, F. and Talwar, K. (2007). Mechanism design via differential privacy. In *FOCS*, volume 7, pages 94–103.

McSherry, F. D. (2009). Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30. ACM.

Mendonça, A. L., Brito, F. T., Linhares, L. S., and Machado, J. C. (2017). Dipcoding: A differentially private approach for correlated data with clustering. In *Proceedings of the 21st International Database Engineering & Applications Symposium*, pages 291–297. ACM.

Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

Agregação Não Supervisionada de Rankings para Redução de Cold-Start em Recuperação Multimodal de Imagens

Wanderson Bezerra da Silva[1], **Rodrigo Tripodi Calumby**[1]

[1]Universidade Estadual de Feira de Santana – Feira de Santana – BA – Brazil

wbsilva@ecomp.uefs.br, rtcalumby@uefs.br

Abstract. *In content-based image retrieval systems, the objective of relevance feedback techniques is to enable the user to express her need without specific knowledge of the low-level image features. For a proper behaviour of this technique, the result set prior to the first interaction of the user must present relevant results. Aiming at attenuating the cold-start problem and improving the initial set, this work experimentally evaluated several rank aggregation methods to combine results obtained with different image ranking features. The results showed promising effectiveness when compared to the baselines considering different modalities of features.*

Resumo. *Em sistemas de recuperação de imagens por conteúdo, a técnica de realimentação de relevância visa permitir ao usuário exprimir a sua necessidade de busca sem conhecer propriedades de baixo nível das imagens. Para o bom funcionamento desta técnica, o conjunto de resultados anterior à primeira interação do usuário deve apresentar resultados relevantes. Para atenuar o problema de cold-start e aprimorar o conjunto inicial de resultados, neste trabalho realizou-se uma avaliação experimental de métodos de agregação de rankings para combinar resultados obtidos a partir de diferentes critérios de ranqueamento de imagens. Resultados considerando diferentes modalidades de características indicam eficácia promissora em relação aos baselines.*

1. Introdução

Dadas as necessidades dos usuários e a grande quantidade de informação disponível, é imprescindível o uso de técnicas eficazes para exploração destas coleções. Em se tratando de imagens, a abordagem mais comum para busca baseia-se na utilização de informações textuais (metadados, palavras-chaves, páginas web, etc.) e no uso de consultas tradicionais em bancos de dados para recuperá-las. Um outro paradigma utiliza a descrição das propriedades visuais (cor, forma, textura, etc.) das imagens para indexá-las e buscá-las. Nos sistemas de recuperação de imagens por conteúdo, a atividade consiste em, dada uma imagem de consulta, calcular a sua similaridade em relação às outras armazenadas. Várias técnicas são utilizadas para capturar e representar as informações visuais das imagens [Torres and Falcão 2006]. Neste processo, a noção de similaridade entre as imagens pode variar de acordo com o usuário que realiza a busca. Com frequência os descritores de conteúdo per si não são capazes de representar apropriadamente o conteúdo conceitual de uma imagem. Este problema é conhecido como gap-semântico e acentua-se em situações de consultas complexas. De modo a atenuar este problema, descritores são combinados para adaptar a busca às necessidades do usuário, o que não é uma tarefa

fácil [Atrey et al. 2010]. Para isso, uma técnica que tem sido empregada com sucesso é a realimentação de relevância, em que o usuário interage com o sistema, indicando a relevância dos itens no resultado e o sistema retorna outros itens possivelmente mais relevantes. Assim, a máquina de busca pode utilizar técnicas de aprendizado de máquina para determinar padrões e criar modelos de representação das necessidades do usuário.

Tradicionalmente, mecanismos de aprendizado de máquina têm sido empregados para aprimoramento de buscas utilizando realimentação de relevância. A proposta da técnica de realimentação de relevância é possibilitar ao usuário exprimir a sua necessidade sem ter que conhecer propriedades de baixo nível da imagem. Esse processo é realizado iterativamente e interativamente [Calumby et al. 2014]. A cada iteração, o algoritmo de aprendizado busca capturar quais as propriedades melhor definem as imagens informadas como relevantes pelo usuário.

Neste contexto, dada a complexidade da consulta ou a pouca informação disponível para a configuração de um sistema de busca, o conjunto inicial de resultados, anterior à primeira interação do usuário, pode não apresentar informações relevantes suficientes, caracterizando o problema do *cold-start*. Isso pode acarretar em um *feedback* pobre (com pouca informação) e consequentemente dificultar a etapa de aprendizado. Por ser uma abordagem iterativa, a influência do primeiro resultado propaga-se para as demais iterações, dado que um resultado inicial ruim limita a troca de informação entre o usuário e o sistema [Calumby et al. 2017] e consequentemente os modelos de aprendizado da intenção do usuário. Visando atenuar este problema, é imprescindível que o resultado da primeira interação seja o melhor possível.

Uma abordagem proposta para este cenário, conhecida como agregação de *rankings* [Lin 2010], é a combinação de resultados obtidos a partir de diferentes critérios de *ranking*, por exemplo, diferentes características visuais das imagens ou medidas de *ranking* baseadas no texto associado à elas. Esta fusão pode ser realizada de diferentes formas, incluindo algoritmos de agregação de *rankings* e algoritmos de rerranqueamento [Mei et al. 2014]. Utilizar métodos de fusão de *rankings*, de modo geral, permite resultados superiores à utilização dos critérios de modo isolado ou com técnicas simples de combinação de escores de relevância.

Considerando a importância de um bom conjunto inicial de resultados nos sistemas de recuperação interativa, este trabalho avalia experimentalmente a redução do *cold-start* por meio da exploração de métodos de agregação de *rankings* considerando diferentes critérios de ranqueamento de imagens.

2. Trabalhos Relacionados

As técnicas de agregação de *rankings* podem ser divididas em duas categorias principais: baseadas em escores ou baseadas em ordem. No primeiro grupo, a função de agregação utiliza as informações de pontuação dos objetos de cada lista. Na segunda, apenas a ordem relativa entre os itens é considerada [Vargas Muñoz et al. 2015]. Dentre as técnicas baseadas em escore, pode-se destacar a família Comb* [Shaw and Fox 1994] (e.g., CombMIN, CombMAX, CombSUM, CombMED, CombMNZ, e CombANZ). Em relação aos métodos baseados em posição, pode-se citar o Median Rank Aggregation (MRA) [Fagin et al. 2003], Reciprocal Rank Fusion (RRF) [Cormack et al. 2009] e Borda [Young 1974].

Para melhorar o ranking inicial de um sistema de recuperação interativa de imagens, [Calumby et al. 2014] apresenta uma adaptação do método de agregação baseado em escore proposto por [Ferreira et al. 2011]. Neste método, a similaridade entre dois objetos é definida como o valor médio de todas as medidas de similaridade disponíveis considerando múltiplas características visuais e textuais. Para consultas com mais de um objeto, os itens de coleção são classificados com base no valor mínimo de distância para cada objeto presente na consulta.

3. Metodologia Experimental

Nesse trabalho foi utilizada a ImageCLEF Photographic Retrieval Task collection [Arni et al. 2009]. Esta base de dados é formada por 20.000 imagens (fotos tiradas a partir de locais ao redor do mundo), em que cada imagem está associada a metadados textuais, como título, data e descrição dos conteúdos semânticos e visuais da imagem. Esta base de dados conta com 39 consultas compostas por um fragmento de texto e três imagens de exemplo.

Os experimentos basearam-se em sete descritores visuais globais [Penatti et al. 2012]: baseados em cor (GCH, BIC, ACC e JAC) e textura (CCOM, LAS, e QCCH). Para a modalidade textual, utilizou-se seis medidas de similaridade entre o texto da consulta e a descrição das imagens, sendo elas: Cosine [Baeza-Yates and Ribeiro-Neto 2008], BM25 [Baeza-Yates and Ribeiro-Neto 2008], Dice [Lewis et al. 2006], Jaccard [Lewis et al. 2006], tf-idf-sum [dos Santos et al. 2009] e Bag-of-words (intersecção de termos normalizada). Foram aplicadas as técnicas de remoção de *stop words* e *stemming*. Apenas os metadados em inglês foram considerados.

Nos experimentos foram utilizados métodos de agregação baseados em escore e métodos baseadas em ordem. Dentre os baseados em escore, aplicou-se os métodos da família Comb*, sendo eles: CombMAX, CombMIN, CombSUM, CombANZ, CombMNZ, e CombMED. Dentre os baseados em ordem, considerou-se: MRA, RRF e Borda. Para cada um dos descritores, foi gerado um *ranking* utilizando as 20.000 imagens da base. Estes *rankings* foram usados como entrada para os métodos de agregação. Foram definidos três cenários de busca. O primeiro cenário considerou apenas informações visuais para gerar os *rankings* de entrada para os métodos de fusão, enquanto no segundo cenário considerou-se apenas as informações textuais. O terceiro experimento representou um cenário multimodal, ou seja, utilizou-se tanto informações visuais quanto textuais.

4. Resultados e Discussões

Considerando que cada um dos *rankings* gerado pelos descritores contém todas as imagens da base de dados, o método CombSUM e suas derivações (CombMED, CombANZ e CombMNZ) apresentaram o mesmo resultado, visto que as derivações diferenciam-se especificamente pelo modo como levam em consideração a quantidade de *rankings* onde cada imagem está presente. Assim, apresentamos aqui apenas os resultados do método CombSUM. Os *rankings* obtidos com os métodos de agregação foram comparados com aqueles obtidos com o método proposto por [Calumby et al. 2014] (aqui denominado MinAvg). Como critério de avaliação, utilizou-se curvas de *Precisão* (P@N). A Figura 1 apresenta os resultados obtidos utilizando apenas os descritores visuais (vis). A Figura 2 apresenta o comparativo considerando os descritores textuais (txt).

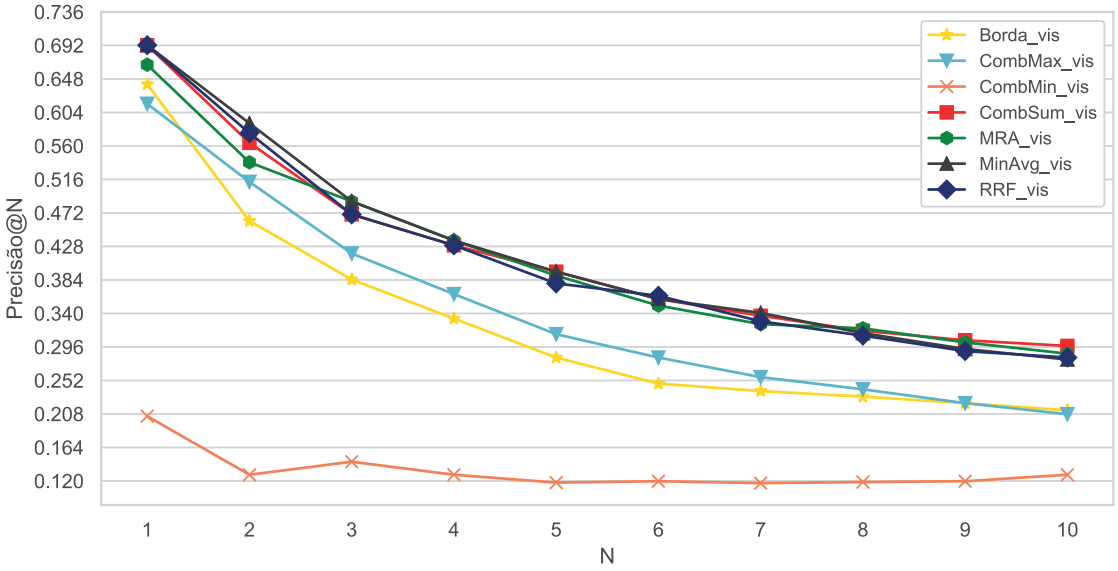


Figura 1. Comparativos dos resultados utilizando apenas informações visuais.

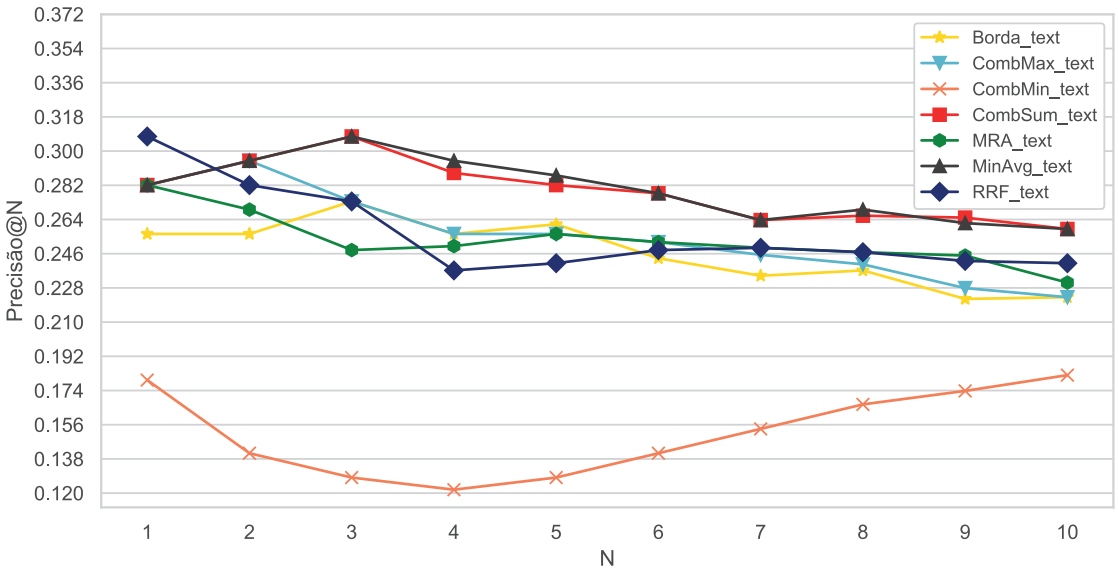


Figura 2. Comparativos dos resultados utilizando apenas informações textuais.

Os resultados demonstram que os métodos avaliados, quando utilizaram rankings baseados em apenas uma modalidade (visual ou textual), não conseguiram obter resultados superiores ao *baseline*. Entretanto, ao analisar os resultados apresentados na Figura 1, percebe-se que há uma sobreposição em termos de P@N entre o MinAvg e os métodos CombSUM, MRA, RRF. Na Figura 2, pode-se perceber que também há uma equivalência entre os métodos MinAvg e CombSUM.

A Figura 3 apresenta os resultados obtidos com cada um dos métodos de agregação no cenário multimodal (mm – informações textuais e visuais). Percebe-se que os métodos MRA e RRF apresentaram ganhos expressivos em termos de P@N em relação ao MinAvg nas primeiras posições do ranking. Além dos ganhos numéricos em termos de P@N nas

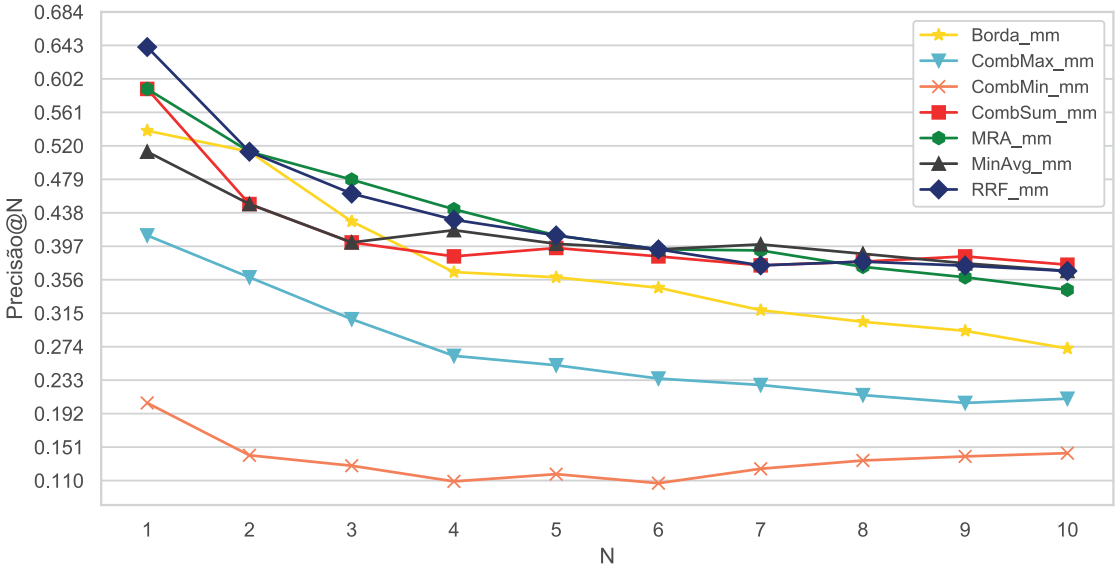


Figura 3. Comparativos dos resultados para o cenário multimodal.

primeiras posições, verificou-se com 95% de confiança por meio do teste de Wilcoxon que esses ganhos são estatisticamente significativos. Para as demais posições dos *rankings* os métodos foram considerados estatisticamente equivalentes.

5. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentada uma avaliação experimental de métodos de agregação de *rankings* em cenários de buscas textuais, visuais e multimodais. Considerando a abordagem multimodal, resultados promissores foram alcançados em relação ao *baseline*. Neste cenário, ganhos expressivos no topo dos *rankings* foram observados. Estima-se que este comportamento tenha ocorrido devido à complexidade das consultas e a variabilidade de qualidade dos descritores na representação quantitativa da relevância. Nestes casos, métodos que usam critérios mais rigorosos para definir a relevância final de uma imagem permitem selecionar um conjunto reduzido, porém altamente relevante e colocá-lo no topo do *ranking*.

Os ganhos no topo do *ranking* são benéficos ao usuário, pois uma maior presença de imagens relevantes nas primeiras posições permite ao usuário fornecer *feedback* significativo sem a necessidade de inspecionar toda a lista de resultados. Vale destacar que nos cenários utilizados nos experimentos, cada um dos descritores gerou um *ranking* utilizando todas as 20.000 imagens contidas na base de dados. Portanto, uma nova etapa de experimentação pode ser realizada para avaliar a qualidade dos *rankings* gerado pelos métodos de agregação quando apenas *rankings* contendo os top-k itens são utilizados como entrada. O ajuste do tamanho dos *rankings* gerados com cada descritor poderá eliminar itens menos representativos presentes nas posições mais profundas, trazendo aos métodos de agregação a possibilidade de operar apenas com os itens mais relevantes encontrados com cada *feature*.

Agradecimentos

Este trabalho contou com o apoio do PIBIC/CNPq (processo nº 134848/2018-7).

Referências

Arni, T., Clough, P., Sanderson, M., and Grubinger, M. (2009). Overview of the imageclefphoto 2008 photographic retrieval task. In *Evaluating Systems for Multilingual and Multimodal Information Access*, pages 500–511. Springer Berlin Heidelberg.

Atrey, P. K., Hossain, M. A., El Saddik, A., and Kankanhalli, M. S. (2010). Multimodal fusion for multimedia analysis: A survey. *Multimedia Syst.*, 16(6):345–379.

Baeza-Yates, R. and Ribeiro-Neto, B. (2008). *Modern Information Retrieval: The Concepts and Technology Behind Search*. USA, 2nd edition.

Calumby, R., R. da S, T., and M. A, G. (2014). Multimodal retrieval with relevance feedback based on genetic programming. *MTAP*, (69):991–1019.

Calumby, R. T., Gonçalves, M. A., and da Silva Torres, R. (2017). Diversity-based interactive learning meets multimodality. *Neurocomputing*, 259:159 – 175.

Cormack, G. V., Clarke, C. L. A., and Buettcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32Nd SIGIR*, pages 758–759. ACM.

dos Santos, K. C. L., de Almeida, H. M., Gonçalves, M. A., and da Silva Torres, R. (2009). Recuperação de imagens da web utilizando múltiplas evidências textuais e programação genética. In *Proceedings of the XXIV SBBD*, pages 91–105.

Fagin, R., Kumar, R., and Sivakumar, D. (2003). Efficient similarity search and classification via rank aggregation. In *Proceedings of the SIGMOD*, pages 301–312. ACM.

Ferreira, C., Santos, J., da S. Torres, R., Gonçalves, M., Rezende, R., and Fan, W. (2011). Relevance feedback based on genetic programming for image retrieval. *Pattern Recognition Letters*, 32(1):27 – 37.

Lewis, J., Ossowski, S., Hicks, J., Errami, M., and Garner, H. R. (2006). Text similarity: an alternative way to search MEDLINE. *Bioinformatics*, 22(18):2298–2304.

Lin, S. (2010). Rank aggregation methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):555–570.

Mei, T., Rui, Y., Li, S., and Tian, Q. (2014). Multimedia search reranking: A literature survey. *ACM Comput. Surv.*, 46(3):38:1–38:38.

Penatti, O. A., Valle, E., and da S. Torres, R. (2012). Comparative study of global color and texture descriptors for web image retrieval. *Journal of Visual Communication and Image Representation*, 23(2):359 – 380.

Shaw, J. A. and Fox, E. A. (1994). Combination of multiple searches. In *TREC-2*, pages 243–252.

Torres, R. D. S. and Falcão, A. X. (2006). Content-based image retrieval: Theory and applications. *Revista de Informática Teórica e Aplicada*, 13:161–185.

Vargas Muñoz, J. A., da Silva Torres, R., and Gonçalves, M. A. (2015). A soft computing approach for learning to aggregate rankings. In *Proceedings of the 24th CIKM*, pages 83–92. ACM.

Young, H. (1974). An axiomatization of borda's rule. *Journal of Economic Theory*, 9(1):43 – 52.

Refinamento do Conjunto Inicial de Resultados baseado em Contexto para Recuperação Interativa de Imagens*

Luciano Araujo Dourado Filho[1], **Rodrigo Tripodi Calumby**[1]

[1]Universidade Estadual de Feira de Santana – Feira de Santana – BA – Brazil

lucianoadfilho@ecomp.uefs.br, rtcalumby@uefs.br

Abstract. *This paper proposes the exploitation of contextual information to refine the initial set of results in interactive image retrieval systems. This context-based method differs from traditional approaches given it considers the rich information present in the relationships between multiple images, rather than simply using traditional methods that computes pairwise distances between two images. In this work, an experimental evaluation of the effectiveness of this technique in different Web search scenarios was carried out. The results showed that the evaluated method was able to significantly improve the effectiveness of the retrieval, specially when applied in conjunction with similarity measures of greater distinctive power.*

Resumo. *Este trabalho propõe a exploração de informações contextuais para refinamento do conjunto inicial de resultados em sistemas de recuperação interativa de imagens. Este método baseado em contexto difere das abordagens tradicionais, pois considera a rica informação presente nas relações entre múltiplas imagens, ao invés de realizar o cômputo de similaridade apenas entre pares de imagens. Neste trabalho, realizou-se uma avaliação experimental da eficácia desta técnica em diferentes cenários de busca na Web. Os resultados mostraram que o método avaliado foi capaz aprimorar significativamente a eficácia da recuperação, especialmente quanto aplicado em conjunto com medidas de similaridade de maior poder distintivo.*

1. Introdução

O avanço das tecnologias de captura e armazenamento de conteúdo digital proporcionou o aumento da quantidade de dados como: imagens, vídeos, *e-mails*, documentos, entre outros. Isso gera demanda por técnicas para sua exploração de forma eficiente e eficaz tendo em vista sua quantidade e disponibilidade crescentes. Esses dados podem ser utilizados para fins diversos, como: medicina, análise forense, estudos da biodiversidade, redes sociais, bibliotecas digitais, entre outros. Assim, a qualidade das técnicas para sua exploração é de fundamental importância na disponibilização de acesso prático e eficaz.

Quando tratam-se de imagens por exemplo, os principais paradigmas de busca consideram informações textuais associadas (anotações, metadados e palavras-chave) para realizar a recuperação a partir de uma base de dados [Baeza-Yates and Ribeiro-Neto 2011]. Outra possibilidade é efetuar a representação das propriedades visuais das imagens, como cor, forma e textura por meio da extração de características baseadas por

*Este trabalho foi desenvolvido com o apoio do PIBIC/CNPq (processo nº 165871/2017-2).

exemplo em *pixels*, segmentos de borda, regiões ou objetos presentes nas imagens. Isso possibilita que, dada uma imagem de consulta, as imagens da base possam ser comparadas por similaridade por meio de uma função de distância e que sejam ranqueadas e retornadas para o usuário [Torres and Falcão 2006]. Para isso, utilizam-se os descritores de imagens[1].

A fim de melhor definir a similaridade entre as imagens de uma base de dados pode-se aplicar técnicas de re-ranqueamento cujo propósito é refinar as distâncias entre elas para torná-las mais adequadas. A motivação por trás disso é que ao melhorar a qualidade da medida de similaridade entre as imagens da base o processo de recuperação torna-se mais eficaz. Isso aumenta a possibilidade de satisfação na experiência de um usuário com o sistema, já que a quantidade de interações que ele precisará realizar para satisfazer suas necessidades tende a ser menor.

Uma abordagem para aprimoramento da eficácia de sistemas de recuperação de imagens é permitir que os usuários interajam com o sistema de modo a indicar a relevância dos itens exibidos nos resultados obtidos para uma consulta, processo chamado de Realimentação de Relevância. Assim, o usuário alimenta o sistema indicando a relevância dos itens encontrados em cada iteração, a partir disso o sistema aprende quais propriedades melhor caracterizam as imagens relevantes e adapta-se para atender às necessidades do usuário [Calumby et al. 2014]. Dessa forma, se o conjunto inicial de resultados apresenta imagens relevantes, a tendência é que o usuário possa fornecer informação de qualidade para o sistema e, assim, os resultados ao longo das iterações subsequentes sejam mais satisfatórios [Calumby et al. 2017]. Por outro lado, abordagens tradicionais de recuperação de imagens produzem resultados baseados no cômputo de similaridade apenas entre pares de imagens, deixando de explorar a informação existente nas relações entre elas [Pedronette et al. 2014]. Alternativamente, é possível combinar técnicas de re-ranqueamento com a realimentação de relevância, a fim de propagar as melhorias na eficácia dos *rankings* iniciais ao longo das iterações de *feedback*.

Neste contexto, este trabalho avalia experimentalmente um método baseado em espaços contextuais que explora as relações entre múltiplas imagens a fim de refinar o computo de distâncias entre elas. Para isso, considera-se uma base de dados heterogênea para avaliar a eficácia de sistemas de recuperação de imagens. O objetivo principal está em estimar o impacto dessa técnica sobre a qualidade do conjunto inicial de resultados considerando o cenário de buscas de imagens na *Web*.

2. Trabalhos Correlatos

Em diversas aplicações, utilizar apenas uma função de distância par-a-par para definir a similaridade entre duas imagens pode não ser uma abordagem eficaz, tendo em vista que deixa-se de explorar as relações existentes entre múltiplas imagens de um banco de dados. Dessa forma, em [Pedronette et al. 2014], os autores propõem uma alternativa para melhor computar as distâncias entre imagens por meio da exploração das informações nas múltiplas relações entre elas em um processo de refinamento iterativo baseado em kNN[2].

De forma simplificada, pode-se descrever o modelo de refinamento por meio de

[1]Par composto por uma função de extração de características de uma imagem e uma função para computar a distância entre duas imagens a partir das suas respectivas características.

[2]k vizinhos mais próximos, do inglês *k Nearest Neighbors*.

dois componentes, d_i e d_j, que representam, respectivamente: a distância entre duas imagens cuja similaridade deseja-se redefinir e a distância do contexto de uma imagem em relação à outra. A partir disso, o grau de similaridade entre as duas imagens é então redefinido conforme a Equação 4. Este processo é formalizado como segue:

A partir de conjunto de imagens $\mathcal{I} = (Img_1, Img_2, Img_3, ..., Img_m)$, define-se:

1. $k \in \mathbb{R}^*_+$, que define o tamanho do contexto (número de vizinhos).
2. $\mathcal{R}ank(Img_i, Img_j)$ como a posição de Img_j numa lista de imagens ordenada por similaridade em relação à Img_i, sendo $i \neq j$.
3. $kNN(Img_r) = \{\ Img_r \in \mathcal{I} \mid \mathcal{R}ank(Img_r, Img_s) < k\ \}$.
4. $kNN(Img_y)_i$ como a i-ésima imagem pertencente ao conjunto do $kNN(Img_y)$.
5. $\delta(Img_A, Img_B)$ como uma função de distância para imagens A e B.

Inicialmente determina-se o valor do componente d_i (Equação 1). Em seguida, calcula-se α (Equação 2) e então o componente d_j (Equação 3). Por fim, a nova distância $\delta'_t(Img_A, Img_B)$ pode ser obtida (Equação 4). O processo é realizado de maneira iterativa com k variando de k_o a k_f para definir o número de vizinhos a ser considerado em cada iteração t, sendo as distâncias reajustadas a partir daquelas da iteração anterior. A intuição por trás disso é que a eficácia dos *rankings* aumente ao longo das iterações de modo que as imagens irrelevantes sejam afastadas das primeiras posições e que o valor de k seja incrementado para levar em consideração mais imagens [Pedronette et al. 2014].

$$d_i = \frac{\delta(Img_A, Img_B)}{k} \tag{1}$$

$$\alpha = \sum_{i=0}^{k} \delta(kNN(Img_A)_i, Img_B) \times (k - i) \tag{2}$$

$$d_j = \alpha / \frac{k \times (k-1)}{2} \tag{3}$$

$$\delta'_t(Img_A, Img_B) = \sqrt{d_i^2 + d_j^2} \tag{4}$$

3. Metodologia Experimental

Para realizar os experimentos, utilizaram-se os mesmos parâmetros propostos em [Pedronette et al. 2014], onde se realizou um processo iterativo determinado pelos valores inicial e final de k (k_o, k_f). Dessa forma, os parâmetros utilizados para k foram: $(k_o, k_f) = \{(1, 10), (2, 10), (3, 10), (4, 10), (5, 10)\}$.

A base de imagens utilizada foi a da *ImageCLEF Photographic Retrieval Task* [Arni et al. 2009], composta por 20.000 imagens de diversos locais ao redor do mundo contendo paisagens, cidades, animais, pessoas, dentre outros. A base inclui 39 consultas e *ground-truth* para avaliação. Nesse contexto, para avaliar o impacto nos resultados, diversas medidas de eficácia foram consideradas: *Precision@20* (P20), *Recall@20* (R20), *Mean Average Precision* (MAP), MAP@20 (MAP20), *Normalized Discount Cumulative*

Gain@20 (NDCG20), *Binary Preference* (BPREF), *Geometric MAP* (GMAP) e *Mean Reciprocal Rank* (MRR). P20 representa a fração de imagens relevantes retornadas em relação ao total de imagens retornadas; R20 representa a fração de imagens relevantes retornadas em relação ao total de imagens relevantes existentes; *Average Precision* (AP) é a média dos valores de *Precision* da curva *Precision*x*Recall* e MAP é o valor médio de AP para múltiplas consultas; BPREF baseia-se numa relação de preferência, penalizando a ocorrência de imagens irrelevantes acima das relevantes; MRR é dado pela média do inverso da posição da primeira imagem relevante no *ranking* de cada consulta.

Os experimentos foram conduzidos a partir do arcabouço de realimentação de relevância apresentado em [Calumby et al. 2014], para isso avaliou-se os *rankings* nas primeiras 20 posições dentre as 1000 retornadas como proposto pelo desafio em [Arni et al. 2009] além do ranking total. Os experimentos foram realizados apenas na modalidade textual, sendo que o texto utilizado descreve o conteúdo semântico das imagens. Foram utilizadas as seguintes medidas de similaridade: Jaccard [Lewis et al. 2006], Cosine [Baeza-Yates and Ribeiro-Neto 2011], BM25 [Lewis et al. 2006], TF-IDF-Sum [Santos et al. 2009], e Dice [Lewis et al. 2006]. Os experimentos foram avaliados com o gabarito fornecido junto à coleção e comparados com o *baseline*, baseado no *ranking* gerado sem a otimização por contexto e dado pela menor distância entre cada imagem do padrão de consulta e as imagem da base.

4. Resultados e Discussões

A qualidade dos *rankings* obtidos com o método proposto foi comparada em relação aos *rankings* obtidos pelos descritores originais sem ajustes das distâncias. Os resultados são apresentados nas Tabelas 1, 2, 3, 4 e 5 (melhores resultados estão em negrito e ganhos são computados do melhor contexto sobre o *baseline*). As tabelas mostram que os descritores utilizados não apresentaram ganhos em termos de GMAP, e que com as medidas JACCARD, DICE e TF-IDF de modo geral, também não houveram ganhos expressivos. Acredita-se que estes resultados estejam associados à simplicidade destas medidas, tendo em vista que, de modo geral, consideram apenas a ocorrência ou não dos termos de consulta nas descrições das imagens para computar a similaridade entre elas.

Diferentemente, as medidas COSINE e BM25, baseadas respectivamente em modelos mais complexos de frequência de termos e abordagem probabilística, apresentaram ganhos superiores de forma consistente para todas as medidas (exceto GMAP). Em termos de P20, o descritor BM25 apresentou resultado inferior quando comparado ao COSINE com ganho relativo de 1,3%, contra 9,3%. Já em termos de R20, ambos apresentaram resultados similares, com ganhos de 7,3% em relação ao baseline. Em termos de BPREF os ganhos foram bastante expressivos, sendo 15,8% (BM25) e 27,6% (COSINE). Considerando MRR, os resultados foram de 18,7% (BM25) e 26,4% (COSINE). As medidas MAP, MAP20 E NDCG20 também indicaram resultados satisfatórios em relação ao baseline, e.g., com MAP20 indicando ganho de 7,4% com COSINE e 13,8% com BM25.

De modo geral, observou-se que, no que diz respeito ao *ranking* inicial, os resultados com utilização do refinamento baseado em contexto alcançaram ganhos relativos médios em termos de *Precision@20* de aproximadamente de 2,8%, 3,0% em *Recall@20*, 4,0% em MAP, e 2,7% em NDCG, em comparação aos resultados sem otimização por contexto. Já em relação ao posicionamento das imagens relevantes ao longo dos *ran-*

kings (BPREF), observou-se um ganho médio relativo aproximado de 10,3% em relação ao baseline.

Tabela 1. Resultados obtidos para JACCARD.

JACCARD								
	P20	R20	MAP20	NDCG20	BPREF	MAP	GMAP	MRR
Baseline	0.1758	0.0706	0.0555	0.1883	0.1362	0.1216	0.0081	0.2687
Contexto: 1-10	0.1758	**0.0706**	**0.0555**	0.1883	0.1362	0.1216	**0.0081**	0.2687
Contexto: 2-10	**0.1795**	0.0677	0.0548	**0.1894**	0.1343	0.1168	0.0040	0.2480
Contexto: 3-10	0.1718	0.0639	0.0531	0.1868	0.1382	0.1195	0.0042	0.2575
Contexto: 4-10	0.1731	0.0657	0.0538	0.1866	0.1431	**0.1232**	0.0049	**0.2707**
Contexto: 5-10	0.1731	0.0653	0.0501	0.1786	**0.1442**	0.1223	0.0052	0.2335
Ganho Relativo	**2.1%**	0.0%	0.0%	**0.6%**	**5.9%**	**1.3%**	0.0%	**0.7%**

Tabela 2. Resultados obtidos para TF-IDF.

TF-IDF								
	P20	R20	MAP20	NDCG20	BPREF	MAP	GMAP	MRR
Baseline	0.1833	0.0807	0.0556	0.1939	0.1514	0.1332	0.0196	0.3240
Contexto: 1-10	**0.1897**	**0.0835**	**0.0572**	**0.1984**	0.1559	**0.1347**	**0.0196**	**0.3240**
Contexto: 2-10	0.1808	0.0793	0.0509	0.1825	0.1553	0.1243	0.0129	0.2803
Contexto: 3-10	0.1718	0.0720	0.0399	0.1703	**0.1608**	0.1239	0.0111	0.2683
Contexto: 4-10	0.1654	0.0689	0.0403	0.1703	0.1607	0.1313	0.0105	0.2938
Contexto: 5-10	0.1641	0.0664	0.0342	0.1609	0.1562	0.1272	0.0105	0.2473
Ganho Relativo	**3.5%**	**3.5%**	**2.9%**	**2.3%**	**6.2%**	**1.1%**	0.0%	0.0%

Tabela 3. Resultados obtidos para DICE.

DICE								
	P20	R20	MAP20	NDCG20	BPREF	MAP	GMAP	MRR
Baseline	0.1756	0.0706	0.0560	0.1890	0.1359	0.1229	0.0084	0.2690
Contexto: 1-10	0.1756	**0.0706**	**0.0560**	**0.1890**	0.1380	0.1229	**0.0084**	**0.2732**
Contexto: 2-10	0.1756	0.0661	0.0548	0.1885	0.1352	0.1185	0.0042	0.2606
Contexto: 3-10	0.1731	0.0646	0.0536	0.1871	0.1385	0.1204	0.0044	0.2555
Contexto: 4-10	**0.1769**	0.0668	0.0539	0.1879	**0.1442**	0.1241	0.0050	0.2713
Contexto: 5-10	0.1718	0.0647	0.0507	0.1804	0.1458	**0.1244**	0.0054	0.2507
Ganho Relativo	**0.7%**	0.0%	0.0%	0.0%	**6.1%**	**1.2%**	0.0%	**1.6%**

Tabela 4. Resultados obtidos para BM25.

BM25								
Método	P20	R20	MAP20	NDCG20	BPREF	MAP	GMAP	MRR
	0.2051	0.0719	0.0426	0.1869	0.1517	0.1291	0.0244	0.2965
Contexto: 1-10	0.2051	0.0763	0.0464	0.2054	0.1517	0.1384	0.0244	0.3494
Contexto: 2-10	**0.2077**	0.0807	0.0482	0.2118	0.1560	0.1346	0.0167	0.3464
Contexto: 3-10	0.1897	**0.0819**	0.0506	0.1953	0.1595	0.1349	0.0191	0.3369
Contexto: 4-10	0.1821	0.0801	0.0511	0.1922	0.1688	0.1412	0.0231	0.3620
Contexto: 5-10	0.1808	0.0790	**0.0528**	**0.2118**	**0.1757**	**0.1485**	**0.0244**	**0.3620**
Ganho Relativo	**1.3%**	**7.3%**	**13.8%**	**3.1%**	**15.8%**	**7.3%**	0.0%	**18.7%**

Tabela 5. Resultados obtidos para COSINE.

COSINE								
	P20	R20	MAP20	NDCG20	BPREF	MAP	GMAP	MRR
Baseline	0.1654	0.0654	0.0379	0.1701	0.1240	0.1157	0.0280	0.2727
Contexto: 1-10	**0.1808**	0.0701	0.0406	0.1870	0.1485	**0.1205**	**0.0280**	0.3133
Contexto: 2-10	0.1731	**0.0702**	**0.0407**	0.1770	0.1457	0.1153	0.0240	0.3094
Contexto: 3-10	0.1769	0.0667	0.0381	0.1852	**0.1582**	0.1155	0.0235	0.3069
Contexto: 4-10	0.1782	0.0687	0.0376	**0.1878**	0.1568	0.1153	0.0225	0.3319
Contexto: 5-10	0.1667	0.0642	0.0333	0.1776	0.1489	0.1097	0.0222	**0.3446**
Ganho Relativo	**9.3%**	**7.3%**	**7.4%**	**10.4%**	**27.6%**	**4.1%**	0.0%	**26.4%**

5. Conclusões e Trabalhos Futuros

Este trabalho realizou uma análise da eficácia do método de Informações Contextuais sobre o conjunto inicial de resultados em cenários de busca de imagens na *Web*. Os resultados encontrados indicaram ganhos satisfatórios quando comparados ao *baseline*. Além disso, foi possível identificar o impacto direto da eficácia das medidas de similaridade textual em capturar as relações de similaridade entre as imagens, sobre o refinamento baseado em contexto. Considerando a aplicação com sucesso do método com ganhos de eficácia nos resultados iniciais de busca, acredita-se que isto gere impacto direto na qualidade da busca de sistemas interativos. Maior relevância nos primeiros resultados exibidos ao usuário permitem melhor troca de informação com o sistema e consequentemente uma melhor captura das intenções de busca do usuário. Neste sentido, trabalhos futuros podem ser realizados para validação experimental do impacto em múltiplas interações.

Referências

Arni, T., Clough, P., Sanderson, M., and Grubinger, M. (2009). Overview of the imageclefphoto 2008 photographic retrieval task. In *Evaluating Systems for Multilingual and Multimodal Information Access*, pages 500–511. Springer Berlin Heidelberg.

Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology Behind Search*. Addison-Wesley, USA, 2nd edition.

Calumby, R. T., da S. Torres, R., and Gonçalves, M. A. (2014). Multimodal retrieval with relevance feedback based on genetic programming. *Multimed Tools Appl*, (69):991–1019.

Calumby, R. T., da S. Torres, R., and Gonçalves, M. A. (2017). Diversity-based interactive learning meets multimodality. *Neurocomputing*, (259):159–175.

Lewis, J., Ossowski, S., Hicks, J., Errami, M., and Garner, H. R. (2006). Text similarity: an alternative way to search MEDLINE. *Bioinformatics*, 22(18):2298–2304.

Pedronette, D. C. G., da S. Torres, R., and Calumby, R. T. (2014). Using contextual spaces for image re-ranking and rank aggregation. *Multimed Tools Appl*, 69(3):689–716.

Santos, K., de Almeida, H. M., Gonçalves, M. A., and da Silva Torres, R. (2009). Recuperação de imagens da web utilizando múltiplas evidências textuais e programação genética. In *XXIV SBBD*, pages 91–105.

Torres, R. and Falcão, A. X. (2006). Content-based image retrieval: Theory and applications. *Revista de Informática Teórica e Aplicada*, (161-185):13(2).

34th Brazilian Symposium on Databases

October 7-10, 2019
Fortaleza - CE - Brazil

TUTORIALS

Promotion

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

Organization

Departamento de Computação, Universidade Federal de Ceará– UFC

Program Chair

Altigran Soares da Silva (UFAM, Brazil)

34th Brazilian Symposium on Databases

October 7-10, 2019
Fortaleza - CE - Brazil

Promotion

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

Organization

Departamento de Computação, Universidade Federal de Ceará– UFC

SBBD Steering Commitee

Ângelo Brayner (UFC)
Bernadette Lóscio (UFPE) coordenadora da CEBD
Carina Dorneles (UFSC)
Sérgio Lifschitz (PUC-Rio)
Fábio Porto (LNCC)
Carmem Hara (UFPR)

SBBD 2019 Commitee

Steering Committee Chair
Bernadette Lóscio (UFPE)

Local Chair
José Maria da Silva Monteiro Filho (UFC, Brazil)

Full Paper Chair
Carina F. Dorneles (UFSC, Brazil)

Short Paper Chair
Fábio Porto (LNCC, Brazil)

Demos and Applications Chair
Robson L. F. Cordeiro (ICMC-USP, Brazil)

Thesis and Dissertation Workshop Chair
Jonice Oliveira (UFRJ, Brazil)

Tutorials Chair
Altigran Soares da Silva (UFAM, Brazil)

Short course Chair
Maria Cláudia Cavalcanti (IME, Brazil)

Workshop Chair
José Antônio Macedo (UFC, Brazil)

Thesis and Dissertation Contest Chair
Caetano Traina Jr. (USP, Brazil)

Graduation Student Workshop Chair
Ticiana Linhares (UFC, Brazil)

Local Organization Committee
SBBD Local Chair: José Maria da Silva Monteiro Filho (DC/UFC)
Leonardo Oliveira Moreira (Instituto UFC Virtual/UFC)
Marum Simão Filho (UNI7)
Angelo Roncalli de Alencar Brayner (DC/UFC)
Javam de Castro Machado (DC/UFC)

Table of Contents (Tutorials)

DuckDB Autopsy: The internals of the "SQLite for Analytics"

Pedro Holanda, Mark Raasveldt

[1] Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands.

{holanda,raasveld}@cwi.nl

Abstract. *The immense popularity of SQLite shows that there is a need for unobtrusive in-process data management solutions. However, there is no such system yet geared towards analytical workloads. We present DuckDB, a novel data management system designed to execute analytical SQL queries while embedded in another process. In our talk, we give an in-depth overview of the internals of DuckDB and the design choices that were made to cater to the use case of embedded analytics. DuckDB is available as Open Source software under a permissive license.*

Resumo. *A imensa popularidade do SQLite demonstra a necessidade de sistemas de gerenciamento de banco de dados (SGBD) embarcados. No entanto, ainda não existe um SGBD embarcado voltado para cargas de trabalho analíticas. Nessa palestra apresentamos o DuckDB, um novo SGBD projetado para executar consultas analíticas enquanto incorporado em outro processo. Apresentamos uma visão geral dos aspectos internos do DuckDB e das decisões de design feitas para atender as cargas analíticas em SGBDs embarcados. O DuckDB já está disponível para download e uso.*

Introduction

In this talk, we present the internal structure of our new system, *DuckDB*. DuckDB is a new purpose-built embeddable relational database management system. DuckDB is available as Open-Source software under the permissive MIT license[1]. DuckDB is no research prototype but built to be widely used, with millions of test queries run on each commit to ensure correct operation and completeness of the SQL interface. DuckDB was built specifically to support the use case of embedded analytics, and focused on fulfilling the following requirements of this use case:

- Efficient transfer of tables to and from the database is essential. Since both database and application run in the same process and thus address space, there is a unique opportunity for efficient data sharing which needs to be exploited.
- High efficiency for OLAP workloads, but without completely sacrificing OLTP performance. For example, concurrent data modification is a common use case in dashboard-scenarios where multiple threads update the data using OLTP queries, and other threads run the OLAP queries that drive visualizations simultaneously.

[1] https://github.com/cwida/duckdb

- High degree of stability, if the embedded database crashes, for example, due to an out-of-memory situation, it takes the host down with it. This can never happen. Queries need to be able to be aborted cleanly if they run out of resources, and the system needs to gracefully adapt to resource contention.
- Practical "embeddability" and portability, the database needs to run in whatever environment the host does. Dependencies on external libraries (e.g., `openssh`) for either compile- or runtime have been found to be problematic. Signal handling, calls to `exit()` and modification of singular process state (locale, working directory, etc.) are forbidden.

Outline

The talk is catered primarily towards people that have a basic understanding of core database systems, and the goal of the talk is to make users more familiar with the internals of modern columnar database systems and specifically the internals of DuckDB. The talk is divided into six sections of 30 minutes each:

1. DuckDB: Introduction and Motivation
2. Parser, Binder and Logical Planner
3. Physical Execution
4. Optimizers
5. Transactions & Storage Layer
6. Interactive Demo

DuckDB: Introduction and Motivation. In the talk, we start by giving a brief overview of DuckDB and the motivation behind embedded analytical systems.

Parser, Binder and Logical Planner. In this section we discuss the parser, binder and logical planner of the DuckDB system. We do this by taking you on a journey of the life of a query inside the database system. We show the internal structures that are created and how the query string is converted into a logical plan.

Physical Execution. After showing how the logical plan is constructed, we show how the system executes the physical plan using its vectorized execution engine [Boncz et al. 2005]. We discuss the techniques that are used for the execution of scans, indexes [Leis et al. 2013], joins, aggregates, sorting and window functions [Leis et al. 2015].

Optimizers. After the base logical plan is created, we show how the optimizers work to create a fast plan. We discuss join order optimization [Moerkotte and Neumann 2008], subquery unnesting [Neumann and Kemper 2015], filter pushdown as well as various simple scalar optimizers such as constant folding and common subexpression elimination.

Transactions & Storage Layer. We discuss the MVCC model [Neumann et al. 2015] that is used by DuckDB and how it works to maintain transactional integrity and atomicity. We also discuss the storage layer of DuckDB. We describe how the data is laid out on disk and talk about how the buffer manager loads and caches data into memory [Leis et al. 2018].

Interactive Demo. We guide people in setting up and using DuckDB in combination with Python. We have a prepared data set and demo use case in which users use DuckDB

to wrangle a set of census data concerning US voters, after which they use *sci-kit learn* to build a full machine learning pipeline that attempts to classify individual voters [Raasveldt et al. 2018].

Referências

Boncz, P. A., Zukowski, M., and Nes, N. (2005). Monetdb/x100: Hyper-pipelining query execution. In *CIDR*.

Leis, V., Haubenschild, M., Kemper, A., and Neumann, T. (2018). Leanstore: In-memory data management beyond main memory. In *ICDE*.

Leis, V., Kemper, A., and Neumann, T. (2013). The adaptive radix tree: Artful indexing for main-memory databases. In *ICDE*.

Leis, V., Kundhikanjana, K., Kemper, A., and Neumann, T. (2015). Efficient processing of window functions in analytical sql queries. *VLDB*.

Moerkotte, G. and Neumann, T. (2008). Dynamic programming strikes back. In *SIGMOD*. ACM.

Neumann, T. and Kemper, A. (2015). Unnesting arbitrary queries. *Datenbanksysteme für Business, Technologie und Web (BTW 2015)*.

Neumann, T., Mühlbauer, T., and Kemper, A. (2015). Fast serializable multi-version concurrency control for main-memory database systems. In *SIGMOD*.

Raasveldt, M., Holanda, P., Mühleisen, H., Manegold, S., et al. (2018). Deep integration of machine learning into column stores. *EDBT*.

FAT in Recommendation Systems

Denis Parra

Pontificia Universidad Católica de Chile (PUC Chile)

Abstract. *In recent years we have experienced an increasing deployment in our daily lives of applications based on Artificial Intelligence technology. Some of these applications are truly amazing, such as self driving cars, translation systems, and automatic detection of illnesses. However, we have also seen applications which are disrupting our daily lives with unintended consequences, such as recommender systems which polarize public opinion and face recognition applications which can hinder our privacy. As the researcher Michael Jordan stated: "Just as early buildings and bridges sometimes fell to the ground — in unforeseen ways and with tragic consequences — many of our early societal-scale inference-and-decision-making systems are already exposing serious conceptual flaws." In this context, a serious and systematic study of FAT (Fairness, Accountability and Transparency) in AI is a critical. The popularity of the recently created FAT conference and several related workshops in academic conferences, especially those related to topics in artificial intelligence, show evidence of how researchers and developers, as well as the society at large, is becoming aware of the effects of AI technology. In terms of recommendation systems, algorithmic transparency has been a topic of study in for at least 10 years, but only recently this area has attracted strong attention in the community. In this tutorial, Prof. Parra will present a general overview of FAT in AI, to then focus on FAT for recommendation systems with a theoretical presentation as well as practical activities.*

REALIZATION

EXECUTION

SUPPORT

SILVER SPONSOR

ACADEMIC SUPPORT