WANQA: Uma Abordagem para Identificar Novas Questões Não Respondíveis em Comunidades de Perguntas e Respostas

Lucas V. Knochenhauer[1], **Carina F. Dorneles**[1], **Leandro K. Wives** [2]

[1]PPGCC / INE - Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

[2]PPGC / INF - Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brasil

{luckasx}@gmail.com, {dorneles}@gmail.com, {wives}@inf.ufrgs.br

Abstract. *Big knowledge repositories are on the web and Question and Answer Communities (CQAs) are one of the most collaborative. Daily, their users post a large volume of questions and a great part of them receives no answers, becoming it useless content. Previous works that aim to solve this problem are dependent on the given characteristics of each community. This article proposes an approach based on a classification that results a model able to classify whether a new question is answerable or not. It uses features available in most CQAs. Experiments with data from different CQAs show that the proposal fulfills its goals.*

Resumo. *Grandes repositórios de conhecimento estão distribuídos pela Web, sendo que um dos mais colaborativos são as comunidades de perguntas e respostas (CQAs). Diariamente, os seus usuários postam grandes volumes de questões e boa parte delas não recebe respostas, tornando-se conteúdo inútil. Trabalhos existentes, que se propõem a resolver esse problema, são dependentes das características presentes em cada comunidade. Neste artigo, é proposta uma abordagem baseada em classificação, que gera um modelo capaz de identificar uma nova questão como respondível ou não, usando características presentes na grande maioria das CQAs. Experimentos com dados de diferentes CQAs mostram que o método proposto cumpre seus objetivos.*

1. Introdução

A disponibilização de informação na Web pode ser feita de várias formas, inclusive através de comunidades de usuários. Nas comunidades de perguntas e respostas, por exemplo, chamadas de *Community Question Answering* (CQA) [Srba and Bielikova 2016], os usuários interagem entre si, postando questões e soluções para determinados assuntos. A interação entre os membros produz diariamente uma grande quantidade de informação e conhecimento sobre diferentes temas. A Figura 1 apresenta um exemplo de questão postada e respostas dadas pelos usuários.

Nessas comunidades, no entanto, um dos problemas é a falta de respostas para determinadas questões, e questões sem respostas não são boas para as CQAs pois acumulam conteúdo inútil para seus usuários. Para ajudar a minimizar essa deficiência, algumas abordagens foram propostas na literatura [Yang et al. 2011a, Dror et al. 2013,

Asaduzzaman et al. 2013, Fong et al. 2015] para determinar se uma nova questão está propensa a receber respostas ou não. Isso porque caso uma questão seja reconhecida como não respondível no momento de sua postagem, isso pode levar o usuário a rever o texto do seu questionamento antes dela ser disponibilizada publicamente. Com isso, aqueles que desejam responder à dúvida apresentada terão menos dificuldades no entendimento, o que facilita a resolução do questionamento.

Figura 1. Exemplo de Perguntas e Respostas

Na literatura, diversos trabalhos [Yang et al. 2011a, Dror et al. 2013, Asaduzzaman et al. 2013, Fong et al. 2015] foram propostos com o intuito de classificar novas questões como respondíveis ou não. De forma geral, a maioria deles faz uso de um conjunto de características próprias de cada CQA. Apesar de otimizar a tarefa, isso gera a necessidade de adaptações em outros contextos, visto que cada comunidade tem sua própria estrutura e forma de interação. Nesses conjuntos não há como inferir quais características melhor identificam as questões. Em algumas das abordagens, por exemplo, ignoraram-se os termos usados nos textos das questões, cuja presença pode revelar se a pergunta está adequada para ser respondida ou não.

Neste artigo, propõe-se um método para classificar uma nova questão como respondível ou não, usando características comumente presentes nas comunidades, independente de suas particularidades, criando-se uma abordagem mais genérica. Porém, a generalidade da proposta pode acarretar diminuição da acurácia em algumas CQAs. Diante disso, para mitigar esse problema, propõe-se a ponderação das características de uma questão, de acordo com a sua importância em uma dada categoria de questão. Por exemplo, a característica "há código de programação" deve ter mais peso para perguntas da categoria "programação" do que para aquelas da categoria de "esportes". Com esse método, é possível treinar um modelo de classificador para cada uma das categorias presentes na CQA e aplicá-lo às postagens dos usuários. As contribuições do presente trabalho podem ser sumarizadas da seguinte forma: (i) definição das características comuns à maioria das comunidades, provenientes das novas questões e dos usuários que as postam; (ii) uso da atribuição de pesos às características a fim de ressaltar sua influência no recebimento de respostas; e (iii) experimentos que confirmam a melhoria da acurácia da classificação com o uso de características comuns ponderadas.

A proposta é validada com experimentos realizados sobre fontes de dados advindos de diferentes CQAs de forma a se comparar a eficácia da proposta e a variação nos resultados com a aplicação de diferentes algoritmos de classificação e de atribuição e variação de pesos. Os conjuntos de dados foram obtidos das comu-

nidades *StackOverflow*[1], *Mathematics*[2] e *Cross Validated* [3]. A mensuração dos resultados foi feita usando as métricas de acurácia, revocação, precisão e medida F [Baeza-Yates and Ribeiro-Neto 2008]. Ao comparar os resultados, pode-se ver que a proposta traz melhorias na classificação em relação aos três *baselines* estabelecidos.

Este artigo está organizado como segue. A Seção 2 detalha a proposta, descrevendo a extração, seleção das características, e o treinamento do modelo. Em seguida, na Seção 3 apresentam-se a configuração dos experimentos executados e os respectivos resultados. Na Seção 4, são discutidos os trabalhos relacionados. Finalmente, na Seção 5 são apresentados trabalhos futuros e conclusões.

2. WANQA - Weighting Analysis for New Question Answerability

O processo de classificação envolve três passos principais: (i) Extração de Características; (ii) Seleção das Características; e (iii) Treinamento do Modelo. A visão geral da proposta é apresentada na Figura 2. Como entrada, são utilizados os dados das novas questões e dos usuários que as postaram. No primeiro passo, é feito um levantamento das características disponíveis nas questões da entrada, incluindo a criação de um vetor *tf-idf*. Logo em seguida, faz-se a atribuição de pesos às características vindas da etapa anterior e filtra-se um subconjunto das características com maiores pesos. No último passo é realizado o treinamento com um algoritmo de classificação, o qual gera um modelo treinado.

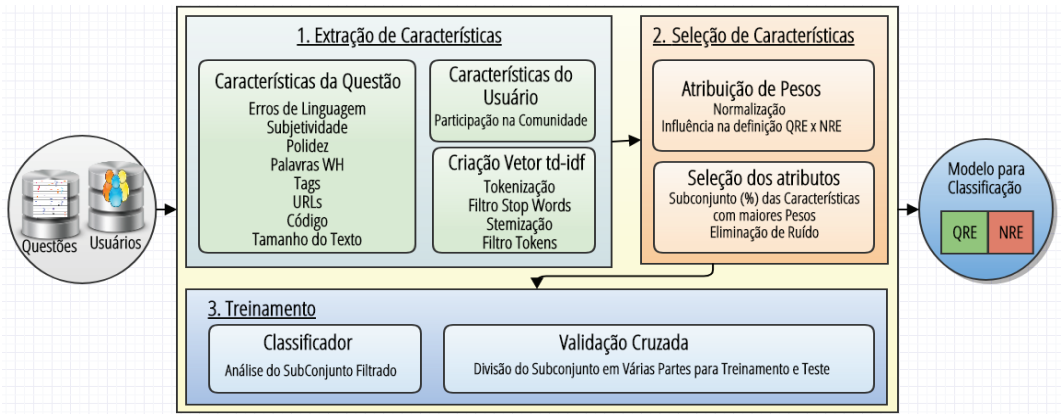

Figura 2. Fluxo Criação do Modelo para Classificação de Questões

A fase de treinamento é realizada através de algoritmos de classificação. Para tanto, foram criadas duas classes: *Questão Respondível - QRE* e *Questão Não Respondível - NRE*. As questões são consideradas respondíveis quando possuem características semelhantes às questões já respondidas na comunidade e na categoria em que esta foi marcada. A classificação das questões não respondíveis funciona do mesmo modo. Através do histórico das questões elenca-se as características que não trazem respostas. Cada categoria possui suas características mais relevantes tanto para o recebimento quanto a falta de respostas. Um exemplo simplificado seria: Questões sobre SQL com termos envolvendo *stored procedures* recebem mais respostas do que Questões sobre SQL com código *javascript*.

[1] `http://www.stackoverflow.com` (Programação)
[2] `http://math.stackexchange.com/` (Matemática)
[3] `http://stats.stackexchange.com/` (Estatística e Análise de Dados)

Nesta abordagem, é empregada a validação cruzada do tipo *k-fold*. Essa validação divide o conjunto de dados em *k* partes das quais $k-1$ são usadas para treinar o algoritmo classificador e a parte restante é usada para testes do modelo treinado, onde o número de classificações corretas e incorretas é medido. O processo é repetido k-1 vezes e uma média dos valores é utilizada para aferir o modelo. Após esse treinamento são gerados o modelo treinado e os resultados contendo o número de classificações corretas e incorretas. O modelo resultante é o que será efetivamente aplicado nas novas questões postadas para classificá-las em QRE ou NRE.

2.1. Extração de Características

Nesta primeira etapa são extraídas características presentes nas novas questões e na descrição do perfil dos usuários. A escolha das características a serem usadas foi feita com base na análise das características consideradas nos trabalhos relacionados e levando em conta os dados disponíveis desde o momento da postagem. Além disso, para que a abordagem não seja dependente de uma comunidade específica, foram analisadas características disponíveis na grande maioria das comunidades.

As características extraídas dos perfis dos usuários são as seguintes: número de dias como membro, número de questões postadas pelo usuário, número de respostas postadas pelo usuário e proporção de respostas em relação à quantidade de perguntas postadas. Para extração do conjunto de características das questões, é necessário obter Título, Descrição, Data e Categorias de cada uma delas.

A Tabela 1 apresenta as 16 características extraídas das novas questões. O Tamanho do Título de uma questão é relevante, pois resume o questionamento em poucas frases. Quando o título é muito vago ele não explica o real problema que se está lidando. Uma característica identificada como relevante é "Título inicia com palavra WH" é uma característica cujo valor é booleano. Isso porque as palavras WH (*what, why, when, who, which, how, whose, whom*) são uma forma de especificar o que deve ser respondido. A presença ou não dessas palavras interrogativas pode ser compreendida como uma forma de medir a objetividade ou subjetividade da pergunta. Erros de Linguagem indicam se os textos estão mal escritos, podendo dificultar o entendimento do problema apresentado e por consequência serem ignorados pelos demais usuários. A descrição de uma questão detalha o problema. Em algumas comunidades esse texto é opcional, a adição dessa informação depende do contexto e da dificuldade da pergunta. A característica "Há Código na Descrição" é um valor booleano indicando se há código-fonte na questão. Geralmente, nos assuntos de programação os problemas estão relacionados aos códigos de algum *software*. Essa característica visa identificar se a inclusão de um código é relevante ou não para a categoria da questão sendo classificada. Seno e Cosseno do Dia e da Hora são características calculadas pelas fórmulas $seno(2\pi * dia/7)$, $cosseno(2\pi * dia/7)$, $seno(2\pi * hora/24)$ e $cosseno(2\pi * hora/24)$. Esses cálculos separam as propriedades Dia da Semana e Hora da Postagem em duas dimensões, positiva e negativa, facilitando a separação de forma linear para os algoritmos de classificação [Zhou and Fong 2016].

Após a extração das características, cria-se um vetor *tf-idf* [Aggarwal 2015] com todos os termos encontrados nas questões e seus respectivos pesos. A determinação do valor *tf-idf* visa identificar os termos que são relevantes para a categoria das questões (Futebol, *Javascript*, etc.) que estão sendo usadas para a criação do modelo. Ressalta-se

que antes da atribuição dos valores é feito um pré-processamento removendo os termos muito comuns, *stop words*, além de *stemização* para reduzir a quantidade de vocábulos. A Figura 3 apresenta um exemplo de valores *tf-idf* atribuídos a termos presentes em algumas questões. Quanto maior o valor atribuído, significa que o termo aparece menos vezes no conjunto de questões em relação aos demais.

Tabela 1. Características extraídas das novas questões

Característica	Descrição
Erros de linguagem	Quantidade de erros gramaticais encontrados na descrição da questão.
Tamanho do Título/Descrição	Quantidade de caracteres no Título/Descrição.
Legibilidade	Facilidade/dificuldade de entendimento do texto apresentado. Valor calculado com base no consenso do cálculo de várias fórmulas da literatura.
Subjetividade	Valor numérico indicando se a postagem busca soluções ou opiniões, pode receber valor no intervalo de 0 (muito objetiva) a 1 (muito subjetiva).
Código na Descrição	Valor booleano indicando se há código de programação na questão.
Polidez	Quantidade de palavras de gentileza *(thank, thanks, please, could, would, help)* utilizadas na descrição.
Seno e Cosseno do Dia	Valores calculados com base na propriedade Dia da Semana.
Seno e Cosseno da Hora	Valores calculados com base na Hora da Postagem. Visam identificar se os horários das postagens influenciam no recebimento de respostas.
Contagem de *tags*	Quantidade de categorias em que a questão foi marcada. Identifica a abrangência da questão em relação aos usuários solucionadores.
Título inicia com palavra WH	Valor booleano que indica se o título da questão inicia com "WH".
Contagem de URL	Quantidade de URLs informadas na descrição. Esse valor mostra se houve uma pesquisa prévia na tentativa de resolver o problema.
Contagem de palavras WH	Quantidade de palavras que iniciam com "WH" no Título/Descrição.

2.2. Seleção de Características

Como o tamanho do conjunto gerado na etapa anterior pode ser muito grande, devido à criação do vetor *tf-idf*, é necessária a definição de um subconjunto das características mais relevantes para uso na fase de treinamento. A relevância é dada pela atribuição de pesos que relaciona as características com as classes *QRE* e *NRE*. Quanto maior o peso atribuído, maior a relevância da característica na classificação. Dessa forma, na segunda etapa do processo, aplica-se o método de *Feature Selection* baseada em filtros [Aggarwal 2015] para detecção das características mais relevantes geradas anteriormente. A Figura 4 apresenta um exemplo de pesos atribuídos a um conjunto de características.

Question	actual	default	known	return	set	string
1	0	0	0,193907	0	0	0
2	0	0	0	0,205024	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0,062495	0	0	0,197223	0,037924	0,05981
6	0,058666	0	0	0	0	0,028073
7	0	0,052391	0	0	0,1026	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0,046418	0	0	0	0,084505	0

Figura 3. Exemplo de valores tf-idf atríbuídos

attribute	weight
QuestionLength	0.257
tagsCount	0.214
bodyHasCode	0.191
android	0.168
LanguageErrors	0.162
titleStartsWithWH	0.152

Figura 4. Exemplo de pesos atríbuídos

A seleção de características baseada em filtros inicia com a atribuição de pesos. Essa atribuição indica um valor que determina o grau de relacionamento da característica com as classes das questões *QRE* e *NRE*. Há diferentes formas para calcular esses pesos. Neste trabalho, foram testadas 3 técnicas diferentes *Information Gain*, *Correlação*

de Pearson e *GINI Index* [Aggarwal 2015], cuja eficácia é apresentada e comparada experimentalmente mais adiante.

Como a produção dos modelos é feita para cada categoria de questão, os treinamentos são realizados individualmente para cada categoria, e as características têm diferentes pesos em cada treinamento. Por exemplo, a característica "Há código na descrição" pode ter um peso maior para as perguntas que envolvem linguagens de programação, mas para dúvidas relacionadas à esportes pode ser irrelevante. Essa estratégia visa reduzir eventuais ruídos que as características irrelevantes podem causar nos modelos. O tamanho do subconjunto com menos ruídos não é conhecido antes do treinamento, por isso é necessário definir nesta etapa quantas das características com melhores pesos são usadas para a criação do modelo. A seleção de um subconjunto tem por objetivo aumentar a acurácia na classificação.

3. Experimentos

Para avaliar a abordagem proposta, foram realizados experimentos usando as características apresentadas na Seção 2, bem como a atribuição de pesos para definir a relevância de cada uma delas. Para fins de análise, a proposta foi comparadas a outras estratégias, gerando comparativos com os seguintes *baselines*: (B1) implementação da proposta de Fong, Zhou e Moutinho [Fong et al. 2015], que usa uma técnica de *swarm* para otimizar a acurácia treinando vários subconjuntos; (B2) conjunto de características sem a criação do vetor *tf-idf*; e (B3) resultados obtidos com o vetor *tf-idf* sem aplicar *feature selection*. O *baseline* B3 equivale às estratégias adotadas por Yang et al. [Yang et al. 2011a] e Dror, Maarek e Szpektor [Dror et al. 2013], onde foram usados os termos encontrados nas questões para, respectivamente, a criação de tópicos e análise da presença desses termos nos textos.

3.1. Conjuntos de Dados

Os experimentos foram executados com 3 conjuntos de questões, cada um deles referente a uma categoria de uma comunidade: categoria "Java", da comunidade *StackOverflow*; (ii) categoria "Álgebra Linear", da comunidade *Mathematics*; e (iii) categoria "Regressão" da comunidade *Cross Validated*. As CQAs citadas pertencem ao mesmo grupo, *StackExchange*[4], porém, os assuntos tratados, características e os membros são distintos uns dos outros. Os conjuntos de questões foram montados de forma que as classes QRE e NRE tivessem um número semelhante de registros. A quantidade de questões empregadas nos experimentos em cada categoria foi de 3000 sobre Java, 2000 de Regressão e 2000 de Álgebra Linear. Na prática, há muito mais questões respondidas do que o contrário, atualmente a distribuição das perguntas sem respostas está como demonstrado na Tabela 2. Quando os dados são desbalanceados os algoritmos classificadores tendem para a classe sobressalente. Sendo assim, optou-se pelo balanceamento através de amostragem com o objetivo de dar equilíbrio aos atributos das questões não respondidas.

3.2. Descrição da avaliação

Os principais objetivos para execução dos experimentos são: (i) realizar uma análise comparativa entre as diferentes formas de classificação (considerando os *baselines* apresentados); (ii) analisar o efeito do uso de características comuns em CQAs; e (iii) investigar a

[4] `http://stackexchange.com/`

Tabela 2. Distribuição das Questões nas Categorias Experimentadas

Comunidade	Categoria	# Questões	# Sem Respostas	% Sem Respostas
Stackoverflow	Java	1.300.000	164.000	12,6%
Mathematics	Álgebra Linear	70.300	9.600	13,6%
Cross Validated	Regressão	15.500	4.800	30,9%

consequência do uso de peso em caraterísticas extraídas das questões, dada a categoria na qual ela foi associada. As métricas de avaliação utilizadas são acurácia (A), precisão (P), revocação (R) e medida-F (F) para as duas classes possíveis, QRE e NRE. O cálculo das métricas é feito com as equações apresentadas na Figura 5, onde TP, TN, FP e FN significam respectivamente verdadeiros positivos e negativos, e falsos positivos e negativos.

$$A = \frac{TP+TN}{TP+FP+TN+FN} \quad P = \frac{TP}{TP+FP} \quad R = \frac{TP}{TP+FN} \quad F = 2 * \frac{P*R}{P+R}$$

Figura 5. Métricas de Avaliação

Na etapa de seleção de características, foram aplicados três métodos: Correlação de *Pearson* (PC), *Information Gain* (IG) e *GINI Index* (GI). A intenção é analisar o relacionamento das características com as classes QRE e NRE e a acurácia resultante conforme cada método de peso empregado. Na etapa de treinamento, foram usados os classificadores disponíveis na ferramenta *Rapidminer Studio*[5]: SVM, Naive Bayes (NB), *Hyperpipes* (HP) e Regressão Logística (RL). Para evitar *overfitting*, ou seja, que o classificador não se ajuste demais ao conjunto de questões do treinamento, foi utilizada a técnica *k-fold* de validação cruzada, sendo k = 10. Dessa forma, o algoritmo classificador executa 10 rodadas dividindo os dados em 10 partes (*folds*) sendo 9 para treinamento e 1 para validação. Para cada método de peso e classificador foram selecionados 9 subconjuntos variando de 10% a 90% das características com os melhores pesos atribuídos. Os resultados de cada um dos 9 subconjuntos têm por propósito verificar a quantidade de características necessárias para obtenção de melhor acurácia. Os dados, códigos e processos manipulados estão disponíveis em `http://github.com/Luckasx/NQClassificationExperiments/`.

3.3. Resultados

No total, foram executadas 360 configurações diferentes em busca da melhor acurácia. Os melhores resultados obtidos nos experimentos estão apresentados na Tabela 3. As linhas na tabela representam a configuração dos *baselines* (B1, B2, B3) e diferentes configurações da proposta apresentada (P.*x*) com os três métodos de peso testados - Correlação de *Pearson* (PC), *Information Gain* (IG) e *GINI Index* (GI) que resultaram em melhor acurácia. Além da acurácia, a tabela exibe a precisão e a revocação obtidas para as classes QRE e NRE nos 3 conjuntos de questões.

A proposta deste trabalho obtém melhor acurácia quando comparada aos demais *baselines*. Dentre os 3 métodos de pesos aplicados, a filtragem com Correlação de *Pearson* (PC) teve melhor acurácia. Pode-se perceber também que os subconjuntos de características nos experimentos P.*x* variam entre 30% e 70% do conjunto total. A acurácia da classificação sem a adição do vetor *tf-idf* (B2) gerou menor acurácia do que o contrário

[5]`http://rapidminer.com/products/studio/`

(B3). Entretanto, a simples adição do vetor *tf-idf* não é suficiente para obter melhora significativa na classificação. O filtro no conjunto total de características (experimentos P.*x*) obteve resultados melhores de 12,94 até 23,8 pontos percentuais do que mantendo todas as características. Isso evidencia que os filtros geram um subconjunto mais adequado à classificação das perguntas. O experimento do *baseline* B1 obteve melhor acurácia do que os experimentos de B2 e de B3. Porém, ressalta-se que os dados continham valores de características específicas às comunidades *StackExchange*, tais como o número de votos que as questões receberam e pontos de reputação dos usuários.

Tabela 3. Melhores Resultados de Acurácia na Classificação

Experimento	Algoritmo	Acurácia	Filtro	% Precisão		% Revocação		% Medida-F	
				QRE	NRE	QRE	NRE	QRE	NRE
Java									
B1	RL	83,20%	-	87,61	79,71	77,33	89,07	82,15	84,13
B2	RL	70,10%	-	69,74	70,47	71,00	69,20	70,36	69,83
B3	RL	74,03%	-	73,45	74,64	75,27	72,80	74,35	73,71
P.PC	**NB**	**86,97%**	**50%**	**83,14**	**91,79**	**92,73**	**81,20**	**87,67**	**86,17**
P.IG	NB	84,13%	70%	82,49	85,96	86,67	81,60	84,53	83,72
P.GI	NB	83,83%	70%	82,06	85,82	86,60	81,07	84,27	83,38
Álgebra Linear									
B1	RL	69,70%	-	69,35	70,06	70,60	68,80	69,97	69,42
B2	RL	64,05%	-	63,39	64,77	66,50	61,60	64,91	63,15
B3	RL	65,30%	-	65,00	65,61	66,30	64,30	65,64	64,95
P.PC	**NB**	**79,65%**	**40%**	**73,89**	**89,06**	**91,70**	**67,60**	**81,84%**	**76,86**
P.IG	SVM	75,30%	30%	73,73	77,09	78,60	72,00	76,09	74,46
P.GI	NB	74,95%	60%	71,13	80,46	84,00	65,90	77,03	72,46
Regressão									
B1	RL	61,50%	-	60,63	62,53	65,60	57,40	63,02	59,86
B2	RL	57,20%	-	56,95	57,47	59,00	55,40	57,96	56,42
B3	SVM	57,65%	-	57,10	58,29	61,50	53,80	59,22	55,96
P.PC	**NB**	**81,45%**	**40%**	**77,61**	**86,53**	**88,40**	**74,50**	**82,65**	**80,07**
P.IG	NB	78,55%	60%	76,76	80,60	81,90	75,20	79,25	77,81
P.GI	NB	77,60%	60%	75,79	79,68	81,10	74,10	78,36	76,79

Em relação à precisão, o método proposto é melhor do que os demais tanto para as QRE quanto para as NRE. Exceto no conjunto de perguntas Java, onde o *baseline* B1 obteve melhor precisão para as QRE. Os valores de Revocação comportam-se de maneira semelhante, mas P.*x* gerou melhor revocação para todas QRE, enquanto a configuração B1 foi melhor nas bases de questões NRE de Java e Álgebra Linear. Os valores da Medida-F ficaram todos próximos à acurácia resultante. Esses resultados mostram que o método proposto está classificando corretamente a maior parte das QRE e NRE.

Dentre os 4 algoritmos classificadores testados, Naive Bayes resultou em melhor acurácia. Todos os classificadores obtiveram melhores resultados em relação aos *baselines* B2 e B3. A melhor acurácia de SVM ficou muito próxima da alcançada por Naive Bayes. A diferença variou entre 0,15 e 3,67 pontos percentuais. *Hyperpipes* atingiu acurácia pouco menor. Apesar de ter revocação maior que 90% em relação às NRE, para a classe QRE alcançou em média 60%, prejudicando a acurácia final. A diferença em relação a Naive Bayes variou de 6,56 a 12,7 pontos percentuais. Os melhores resultados de Regressão Logística decorreram da classificação do menor subconjunto, com apenas 10% das características. Nesse caso, o menor subconjunto gerou melhores resultados por causa da regularização L1, uma maneira de evitar *overfitting* penalizando coeficien-

tes grandes. Os valores resultantes para acurácia foram de 7,9 a 13 pontos percentuais menores do que os obtidos com Naive Bayes. Sendo assim, a partir dos resultados dos experimentos pode-se recomendar o uso dos dois classificadores, Naive Bayes e SVM, de modo a obter melhor acurácia na classificação.

A Figura 6 apresenta os valores de acurácia obtidos na execução de P.PC, P.IG e P.GI para o classificador Naive Bayes em cada subconjunto conforme os pesos aplicados. Os gráficos demonstram que os subconjuntos derivados da Correlação de Pearson tiveram melhor acurácia nos 3 conjuntos de questões. Outro comportamento que se destaca é a melhora da classificação conforme se aumenta o tamanho do subconjunto. A partir do melhor ponto, por volta de 50% do conjunto total, há uma queda brusca na acurácia. Essa piora deve-se ao fato de que boa parte do conjunto torna-se ruído para o algoritmo classificador, ou seja, não foi considerada relevante para identificar as classes das questões.

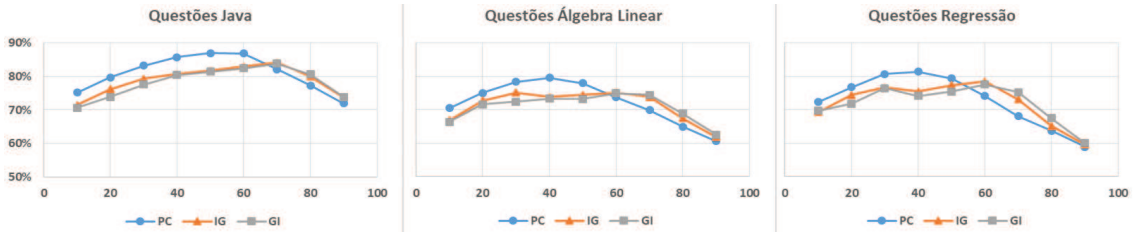

Figura 6. Acurácia X Filtro Naive Bayes

Outro ponto importante consiste em analisar como as características extraídas foram pesadas em cada grupo de questões e métodos de cálculo de peso. A Tabela 4 apresenta a posição da característica dentro do conjunto total. A primeira observação que pode ser feita é que a maioria das características permaneceu na metade superior do conjunto. Dentre as propriedades que estiveram sempre presentes nos subconjuntos com melhor acurácia estão: Tamanho da Descrição, Contagem de *Tags*, Erros de Linguagem, Legibilidade, Polidez, Número de Questões Postadas e Contagem de URL. Por outro lado, apenas a característica Seno Dia foi considerada ruído em todos os grupos de questões. Pode-se verificar também que não há uma equivalência nos pesos gerados pelos diferentes métodos. Por exemplo, nas questões de Álgebra Linear a característica Número de Respostas Dadas foi considerada ruído pelo peso Correlação de Pearson e, em contrapartida, foi considerada a 3ª mais importante tanto para *Information Gain* quanto para *GINI Index*.

A Tabela 4 também ajuda na validação da proposta em relação à filtragem das características. Por exemplo, a propriedade Há Código Na Descrição foi considerada relevante para as Questões Java e Álgebra Linear, porém para Regressão foi apontada como ruído. Outros exemplos seriam Número de Respostas Dadas e Proporção de Respostas/Perguntas consideradas relevantes apenas em Questões Java.

Aplicou-se o teste t para duas amostras pareadas com as seguintes hipóteses: H0: em média as abordagens WANQA e Baseline x tem mesma acuracidade e H1 (em média, a abordagem WANQA tem maior acurácia que o Baseline x). Obteve-se os seguintes valores p: (WANQA/B2)=0,0098 – (WANQA/B3)=0,0189 – (WANQA/B1)=0,0701. Considerando o nível de significância em 5%, pode-se rejeitar as hipóteses de igualdade em relação aos baselines 2 e 3. No baseline 1 não é possível a mesma afirmação, para re-

jeitar a hipótese teria que considerar o nível de significância em 10%. Ressalta-se que o baseline 1 foi proposto para trabalhar com questões do StackOverflow.

Tabela 4. Ranking dos Pesos das Características Extraídas

Característica	# Java (7670)			# Álg. Linear (2865)			# Regressão (3530)		
	PC	IG	GI	PC	IG	GI	PC	IG	GI
Tamanho da Descrição	1	4	3	3	4	4	5	1	1
Contagem de *tags*	2	14	13	1	1	1	6	10	9
Há Código na Descrição	3	11	11	72	347	244	2764	3530	3530
Erros de Linguagem	5	5	5	2	5	5	14	41	35
Título Inicia com Palavra WH	6	30	28	91	542	317	1676	3528	3528
Número de Questões Postadas	7	9	10	191	21	16	1341	593	453
Contagem de Palavras WH no Título	9	35	33	73	345	234	1703	1182	1304
Número de Respostas Postadas	14	2	2	1915	3	3	3278	7	7
Legibilidade	21	25	23	4	12	10	525	201	160
Proporção de Respostas/Perguntas	24	1	1	2142	2	2	2953	6	6
Polidez	36	110	95	71	235	165	10	8	8
Contagem de Palavras WH na Descrição	38	206	184	155	224	159	2306	2127	1213
Número de Dias Como Membro	54	6	7	23	13	11	3442	105	72
Tamanho do Título	203	308	268	2818	247	220	64	14	14
Contagem de URLs	434	1336	1091	32	122	99	640	583	404
Subjetividade	3737	999	784	265	7	6	1400	73	60
Seno Hora	16	101	87	1654	552	357	1382	1065	732
Cosseno Hora	207	363	308	136	159	105	122	328	243
Seno Dia	6588	7670	7670	2111	1861	1110	3143	3529	3529
Cosseno Dia	201	1227	880	177	601	476	1056	2272	2201

4. Trabalhos Relacionados

Nesta seção são apresentados e comparados trabalhos relacionados à tarefa de classificação de questões. Os trabalhos citados são abordagens que analisam o conteúdo das comunidades para identificar os motivos pelos quais parte das questões não possuem respostas. Além disso, as abordagens propõem identificar que as questões receberão ao menos uma resposta, que é também o objetivo desta proposta.

Asaduzzaman et al.[Asaduzzaman et al. 2013] estudaram os motivos de uma questão permanecer sem respostas na comunidade StackOverflow. Nesse trabalho, uma questão não respondida significa que não recebeu respostas até um mês depois da postagem. Os resultados obtidos não foram muito satisfatórios, mas as características analisadas serviram como ponto de partida para elaboração de abordagens de classificação de questões. Saha, Saha e Perry [Saha et al. 2013] analisaram toda a base de perguntas e respostas disponível no StackOverflow. A análise das questões foi feita empregando duas medidas: *Information Gain* e *Information Gain Ratio* sobre 12 características. A partir dos valores encontrados, efetuaram a classificação das questões com dois grupos de características. No primeiro grupo estavam somente as 6 características com as medidas mais altas e no segundo todas as características. A conclusão dos autores é que a diferença nos resultados dispensa o uso de todas as características disponíveis para identificar as questões que não estão respondidas na base.

Chua e Banerjee [Chua and Banerjee 2015] desenvolveram um *framework* para explicar porque algumas questões são respondíveis e outras não. Foram analisados os

metadados e conteúdo das questões. O conjunto de dados utilizado foi composto por 3000 questões sobre Java postadas no *site* StackOverflow. Algumas das inferências encontradas foram que as questões de usuários novos tendem a ser mais respondidas e que descrições breves ou poucas *tags* atraem respostas.

O trabalho de Yang et al. [Yang et al. 2011a] foi o dos primeiros a classificar questões em respondíveis ou não. A análise das questões foi feita usando tópicos latentes descobertos nos textos e características adicionais, como tamanho da questão e hora da postagem. Os autores declaram que os resultados ainda são ruins para uso prático. Baseando-se no trabalho de Yang et al. [Yang et al. 2011a], Dror, Maarek e Szpektor [Dror et al. 2013] buscaram algo ainda mais específico: predizer quantas respostas uma nova questão vai receber. A abordagem proposta consiste em avaliar características e vetores de palavras através de um conjunto de modelos de subárvores pois as categorias na comunidade estudada estão organizadas em estrutura de árvore. Dessa forma, a classificação é feita com a combinação de regressões logísticas dos nodos folha e respectivos nodos superiores. Porém, essa combinação de modelos só deve ser vantajosa quando as categorias estão organizadas em árvore. O estudo que mais se aproxima da proposta deste artigo é o de Fong, Zhou e Moutinho [Fong et al. 2015]. A proposta dos autores foi usar *feature selection* com aplicação de uma meta heurística chamada *Accelerated Particle Swarm Optimization* (APSO)[Yang et al. 2011b]. A técnica consiste em treinar os modelos iterando com vários subconjuntos de características até que se atinja um valor esperado ou um certo número de rodadas. Com a aplicação do APSO obteve-se melhor acurácia. Porém, essa proposta trabalha de forma não determinística, ou seja, nem sempre os melhores subconjuntos de características serão selecionados.

A maioria dos trabalhos estudados não considera o uso das palavras escritas na composição das perguntas. Yang et al. [Yang et al. 2011a] usaram os termos para extração de tópicos, porém, o conjunto extraído não trouxe resultados de uso prático. Dror, Maarek e Szpektor [Dror et al. 2013] usaram a presença ou não dos termos na classificação, mas sem especializar o treinamento em cada categoria. Apenas um dos trabalhos [Fong et al. 2015] empregou seleção de características, diferente das outras propostas que usaram todo o conjunto. A seleção de características faz com que se elimine o ruído das propriedades irrelevantes e assim melhora a acurácia da classificação. Um outro ponto a ser observado, que causa diminuição na acurácia, é classificar ao mesmo tempo, questões de todas as categorias, como visto em [Yang et al. 2011a, Dror et al. 2013, Asaduzzaman et al. 2013, Saha et al. 2013]. O treinamento de cada categoria criando um vetor *tf-idf* traz melhor acurácia, conforme visto nos experimentos, pois os termos mais pertinentes aos assuntos das categorias são ressaltados.

5. Conclusões e Trabalhos Futuros

Neste artigo foi apresentada uma proposta para a classificação de novas questões postadas em CQAs como respondíveis ou não respondíveis. Diferente de trabalhos anteriores estudados, buscou-se apresentar um método aplicável à maioria das CQAs. Os resultados apresentados pelos experimentos confirmam que a adição de um vetor *tf-idf* e a filtragem das características de acordo com os seus pesos melhoram a acurácia dos modelos gerados. Com os resultados obtidos nos experimentos, é possível definir Naive Bayes e SVM como bons classificadores para o problema. Para cálculo dos pesos, na definição da relevância das características, ficou evidente nos resultados apresentados que a Correlação

de Pearson foi melhor que as demais utilizadas. Como trabalhos futuros, pretende-se elaborar um modo automático para definir o tamanho de subconjunto que traz melhor acurácia e testar conjuntos de questões de categorias mais diversas às dos experimentos, como, por exemplo, política ou saúde.

Referências

[Aggarwal 2015] Aggarwal, C. C. (2015). Mining text data. In *Data Mining: The Textbook*, chapter 13, pages 288–291;429–433. Springer Publishing Company, Incorporated.

[Asaduzzaman et al. 2013] Asaduzzaman, M., Mashiyat, A. S., Roy, C. K., and Schneider, K. A. (2013). Answering questions about unanswered questions of stack overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 97–100, Piscataway, NJ, USA. IEEE Press.

[Baeza-Yates and Ribeiro-Neto 2008] Baeza-Yates, R. and Ribeiro-Neto, B. (2008). *Modern Information Retrieval*. Addison-Wesley Publishing Company, USA, 2nd edition.

[Chua and Banerjee 2015] Chua, A. Y. and Banerjee, S. (2015). Answers or no answers: Studying question answerability in stack overflow. *Journal of Information Science*, 41(5):720–731.

[Dror et al. 2013] Dror, G., Maarek, Y., and Szpektor, I. (2013). Will my question be answered? predicting "question answerability"in community question-answering sites. In *Proceedings of the 2013th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III*, ECMLPKDD'13, pages 499–514, Berlin. Springer.

[Fong et al. 2015] Fong, S., Zhou, S., and Moutinho, L. (2015). Text analytics for predicting question acceptance rates. *IT Professional*, 17(4):34–41.

[Saha et al. 2013] Saha, R. K., Saha, A. K., and Perry, D. E. (2013). Toward understanding the causes of unanswered questions in software information sites: A case study of stack overflow. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 663–666, New York, NY, USA. ACM.

[Srba and Bielikova 2016] Srba, I. and Bielikova, M. (2016). A comprehensive survey and classification of approaches for community question answering. *ACM Transactions on the Web*, 10(3):18:1–18:63.

[Yang et al. 2011a] Yang, L., Bao, S., Lin, Q., Wu, X., Han, D., Su, Z., and Yu, Y. (2011a). Analyzing and predicting not-answered questions in community-based question answering services. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 1273–1278. AAAI Press.

[Yang et al. 2011b] Yang, X.-S., Deb, S., and Fong, S. (2011b). Accelerated particle swarm optimization and support vector machine for business optimization and applications. *Networked Digital Technologies*, pages 53–66.

[Zhou and Fong 2016] Zhou, S. and Fong, S. (2016). Exploring the feature selection-based data analytics solutions for text mining online communities by investigating the influential factors: A case study of programming cqa in stack overflow. In *Big Data Applications and Use Cases*, pages 49–93. Springer.