Lucas Viana Knochenhauer

WANQA: UMA ABORDAGEM PARA IDENTIFICAR NOVAS QUESTÕES NÃO RESPONDÍVEIS EM COMUNIDADES DE PERGUNTAS E RESPOSTAS

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Ciência da Computação.
Orientadora: Profa. Carina Friedrich Dorneles, Dra.
Universidade Federal de Santa Catarina

Florianópolis

2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

KNOCHENHAUER, LUCAS VIANA
WANQA: UMA ABORDAGEM PARA IDENTIFICAR NOVAS QUESTÕES NÃO RESPONDÍVEIS EM COMUNIDADES DE PERGUNTAS E RESPOSTAS / LUCAS VIANA KNOCHENHAUER ; orientadora, CARINA FRIEDRICH DORNELES , 2019.
77 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Ciência da Computação, Florianópolis, 2019.

Inclui referências.

1. Ciência da Computação. 2. Comunidades de Perguntas e Respostas. 3. Classificação de Texto. 4. Recuperação da Informação. I. , CARINA FRIEDRICH DORNELES. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Lucas Viana Knochenhauer

WANQA: UMA ABORDAGEM PARA IDENTIFICAR NOVAS QUESTÕES NÃO RESPONDÍVEIS EM COMUNIDADES DE PERGUNTAS E RESPOSTAS

Esta Dissertação foi julgada adequada para a obtenção do Título de "Mestre em Ciência da Computação", e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Florianópolis, 27 de Fevereiro 2019.

Prof. José Luís Almada Güntzel, Dr.
Coordenador do Programa

Banca Examinadora:

Profa. Carina Friedrich Dorneles, Dra.
Universidade Federal de Santa Catarina
Orientadora

Prof. Roberto Willrich, Dr.
Universidade Federal de Santa Catarina

Prof. Daniel Hasan Dalip, Dr.
Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG
(videoconferência)

Profa. Renata Galante, Dra.
Universidade Federal do Rio Grande do Sul
(videoconferência)

AGRADECIMENTOS

Primeiramente, eu agradeço a Deus pelo suporte durante esta etapa da minha vida.

Agradeço à professora Carina por ter acreditado na ideia deste trabalho contribuindo com suas orientações.

Agradeço à minha esposa Carla por ter participado de todos os momentos, bons e ruins, desse período acadêmico.

Agradeço a todos os professores e revisores que leram, criticaram e sugeriram melhorias quando da publicação do artigo, qualificações e defesa.

Agradeço aos meus gestores na empresa por terem dado flexibilidade no meu horário de trabalho e assim pude frequentar às disciplinas.

Agradeço ao PPGCC-UFSC pelo auxílio financeiro para a participação e apresentação de artigo no evento SBBD 2018 no Rio de Janeiro.

Por fim, agradeço a todos os familiares e amigos que fizeram parte deste momento da minha vida.

Existem coisas melhores adiante do que qualquer outra que deixamos para trás.

Clive Staples Lewis

RESUMO

Grandes repositórios de conhecimento estão distribuídos pela Web, sendo um dos mais populares e colaborativos as comunidades de perguntas e respostas, ou as chamadas *Community Question Answering* (CQA). Nessas comunidades, os usuários perguntam e respondem questões uns dos outros, além de avaliarem o conteúdo produzido. Entretanto, devido ao grande volume de questões postadas diariamente, muitas dessas questões não recebem respostas. Esse problema vem sendo estudado para entender sua razão e evitar que novas questões permaneçam sem retorno. Nesses estudos, características específicas presentes nestas comunidades foram exploradas de forma a se criar abordagens que classifiquem questões recém postadas em *respondíveis* ou *não respondíveis*. Entretanto, as abordagens anteriores são altamente dependentes das características existentes em uma determinada comunidade, como votos que o usuário recebe ou pontos de reputação. Dessa forma não é possível sua portabilidade a comunidades que não possuem as mesmas características. Para resolver este problema, neste trabalho é proposta uma abordagem que gera um modelo capaz de classificar uma nova questão em *respondível* ou *não respondível* que seja aplicável na maioria, ou até mesmo em todas as comunidades. Para isso, diferentes comunidades foram analisadas a fim de se extrair o maior número de características comuns possíveis. Com esse conjunto pode-se treinar um modelo de classificação para uma categoria de questões em qualquer comunidade de perguntas e respostas. A finalidade desse modelo é avaliar novas questões no momento em que o usuário as submete. Dessa forma, as novas questões *não respondíveis* serão detectadas pelo modelo classificador e assim o usuário pode ser informado para ajustar a questão sendo postada. A proposta WANQA foi testada através de um conjunto de experimentos realizados em diferentes comunidades de pergunta e resposta, com diferentes algoritmos classificadores. Os resultados foram maior acurácia, precisão e revocação do que os baselines experimentados.

Palavras-chave: Comunidade de Perguntas e Respostas. Classificação de Texto. Recuperação de Informação.

ABSTRACT

There are large knowledge repositories spread over the web. A kind of these repositories is the Community Question Answering (CQA). In these communities, the people ask and answer questions one from another. Besides, the users can evaluate the posted content. However, because of the large daily volume of new questions, many of them remain unanswered. This problem has been studied to understand its reason and to avoid new questions without answers. In these projects, the question and user features were explored to classify the questions' answerability. The previous approaches are highly dependent of the existing characteristics of each community as votes and user reputation. This condition turns the approaches not portable to other communities with different features. To solve this problem, our approach generates a classification model to identify new questions as answerable or not and that it can be used in the most of the communities. For this, we analyzed several communities to find a set of common characteristics. With this set, we can train a classification model for a category of questions in any community question answering. The goal of this model is to prove a new question at the time when the user submit it. Thus, the new unanswerable questions can be detected by the classifier model and the user can be informed to adjust a submitted question. The WANQA approach was tested through a set of experiments with data from several communities and different classification algorithms. The results were better values for accuracy, precision and revocation than seen at the baseline experiments.

Keywords: Community Question Answering. Text Classification. Information Retrieval.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

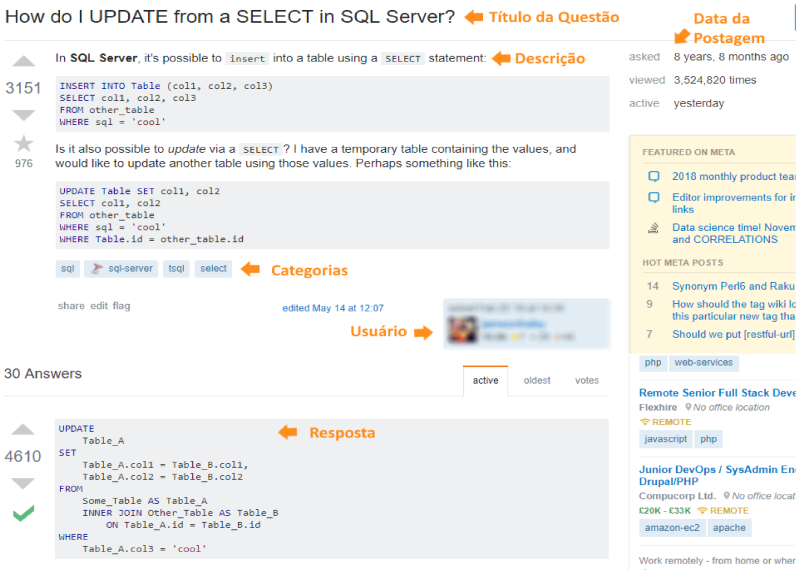
1 INTRODUÇÃO

Dados, informação e conhecimento estão distribuídos pela Web nos mais variados meios, tais como redes sociais, enciclopédias colaborativas e fóruns, dentre outros. Neste contexto, uma das formas de criação e manutenção de conteúdo é através de uma comunidade de usuários. Nas comunidades de perguntas e respostas, por exemplo, chamadas de *Community Question Answering* (CQA) (SRBA; BIELIKOVA, 2016) os usuários interagem entre si, postando questões e respostas para determinados assuntos. A interação entre os membros produz diariamente uma grande quantidade de informação e conhecimento sobre diferentes assuntos. Essas postagens são armazenadas e podem ser visualizadas por qualquer pessoa navegando na *web*. As questões contém um título resumindo o problema e uma descrição detalhada. Além disso, os usuários identificam, através de marcações, quais os assuntos envolvidos. As perguntas podem receber várias respostas dos demais usuários. Na Figura 1 apresenta-se um exemplo de questão postada, respectivas marcações de categoria, informações do usuário que a postou, data da postagem e também uma das 30 respostas recebidas.

Nestas comunidades, um dos problemas existentes é a falta de respostas para determinadas questões. Questões sem respostas não são boas para as CQA, pois acumulam conteúdo inútil para seus usuários. Para ajudar a resolver essa deficiência, algumas abordagens foram criadas na literatura (YANG et al., 2011; DROR; MAAREK; SZPEKTOR, 2013; ASADUZZAMAN et al., 2013; CHUA; BANERJEE, 2015; FONG; ZHOU; MOUTINHO, 2015) para determinar se uma nova questão está propensa a receber respostas ou não. Se uma questão pudesse ser reconhecida como não respondível no momento de sua postagem, isso poderia levar o usuário a rever o texto sendo postado assim como a marcação de categoria. Com isso, aqueles que desejam responder à dúvida apresentada terão menos dificuldades no entendimento, o que facilita a resolução da questão.

A ausência de respostas faz com que o conteúdo postado seja inútil para toda a comunidade, pois, a questão apresentada não foi resolvida. Assim, quando a busca por soluções dentro dessas comunidades depara-se somente com questões faz com que a comunidade perca seu prestígio na *Web*. Por consequência, pode ocorrer abandono dos usuários e em casos extremos o fechamento da comunidade. Por outro lado, questões de melhor qualidade contribuem para aumento da popularidade e enriquecem o conteúdo armazenado nas comunidades

Figura 1 – Exemplo de uma Questão Postada e Respectivas Respostas.



Fonte: Adaptação da página StackOverflow[1]

(BALTADZHIEVA; CHRUPAŁA, 2015). Na Tabela 1 apresenta-se a distribuição de perguntas com e sem respostas para algumas categorias no tempo de realização deste trabalho.

Tabela 1 – Distribuição de Questões nas Categorias

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%

As questões postadas podem deixar de receber respostas por diversas causas: (i) o texto da descrição pode ser curto ou longo demais (CHUA; BANERJEE, 2015); (ii) a atribuição de categorias erradas a uma

[1] https://stackoverflow.com/questions/2334712/how-do-i-update-from-a-select-in-sql-server

questão que direciona a dúvida para um grupo de usuários que não entende o assunto; (iii) a questão é muito complexa (LIU et al., 2013); e (iv) a postagem não está de acordo com os padrões seguidos pela comunidade. Nesses casos, os membros podem interferir para a remoção ou modificação da questão (ASADUZZAMAN et al., 2013). Em contrapartida, os usuários que são mais participativos conhecem a maneira como devem ser feitas as publicações para que se consiga respostas.

Neste trabalho, uma questão é considerada respondível quando possui características para receber ao menos uma resposta. Identificar se uma questão receberá respostas envolve uma grande quantidade de variáveis desde os usuários da comunidade a características do texto postado, momento da postagem, assunto abordado, dentre outros. Para realizar essa tarefa é necessária uma análise de como essas variáveis contribuem ou não para o recebimento de respostas. Essa análise demanda grande processamento sobre um conjunto de dados existentes contendo em alguns casos milhares de características.

Uma das primeiras soluções propostas para a resolução deste problema foi a análise da questão através da extração de tópicos a partir do seu conteúdo textual (YANG et al., 2011). A extração de tópicos consiste em analisar os termos no texto postado e associar um assunto ao texto, além da categoria indicada pelo usuário. Esse estudo não teve resultados muito bons pois, segundo os autores, seria necessário definir mais características das questões não respondidas para análise. Como essa foi a primeira abordagem para a solução do problema, definiu-se poucas características para uso na classificação das questões. Isso vem a ser melhorado nos trabalhos posteriores.

Motivados pelo trabalho anterior (YANG et al., 2011), Dror, Maarek e Szpektor (2013) desenvolveram uma abordagem combinando características da questão e do usuário que a postou para predizer uma questão como *respondível* ou *não respondível*, inclusive prevendo o número de respostas a ser recebido. Porém, a abordagem proposta aplica-se somente a comunidades cujas questões estão organizadas de forma hierárquica, como a comunidade Yahoo! Answers[2].

Asaduzzaman et al. (2013) fizeram um estudo para tentar identificar características específicas das questões sem respostas também com o objetivo de classificar uma dada questão em *respondível* ou *não respondível*. A partir de características observadas, tais como a presença de código de programação e número de respostas dadas pelo usuário, outros trabalhos foram propostos com o objetivo de realizar a classificação de questões. O trabalho de Chua e Banerjee (2015) fez uma análise

[2]https://answers.yahoo.com/

com regressão linear hierárquica para entender porque as postagens se mantêm sem respostas. A partir dos resultados obtidos, cada variável detectada foi analisada apontando sua relação com o fato da questão ter sido respondida ou não.

Outra abordagem, desenvolvida por Fong, Zhou e Moutinho (2015), obteve bons resultados usando seleção de características. A técnica de *feature selection*, utilizada pelos autores, aplica uma estratégia de metaheurística, que itera um certo número de vezes com diferentes combinações do conjunto de características disponíveis nas questões e entrega o conjunto que gera melhor resultado. A grande limitação dos trabalhos mencionados (ASADUZZAMAN et al., 2013; CHUA; BANERJEE, 2015; FONG; ZHOU; MOUTINHO, 2015) é que suas propostas baseiam-se em características presentes em determinadas comunidades de domínio específico, principalmente na comunidade *StackOverflow*. Exemplos incluem a reputação do usuário e votos dados para questões que podem não existir em outras comunidades.

Neste trabalho, propõe-se a classificação de novas questões em *respondível* ou *não respondível* no momento da postagem, cuja abordagem possa ser aplicada a um conjunto mais genérico de comunidades de perguntas e respostas. O principal objetivo desta dissertação é criar uma abordagem que use características mais comumente presentes nas comunidades e obtenha melhor acurácia na classificação das novas questões como *respondíveis* ou *não respondíveis*. Tais características estão disponíveis no momento da postagem e são provenientes das questões presentes nas discussões e dos usuários que criam as questões. Para aumentar a acurácia, acredita-se que o treinamento de um classificador deve ser realizado utilizando questões pertencentes a uma categoria em comum, como, por exemplo, Futebol ou Javascript. Além disso, cria-se um vetor TF-IDF dos termos usados nas postagens a fim de ressaltar os termos mais presentes em cada categoria de questão. As características e termos do vetor são pesados para identificar a importância desses valores na definição da questão ser respondida ou não. Dessa forma pode-se filtrar o número de características e diminuir o tamanho do modelo classificador a ser treinado.

Para que seja possível criar uma abordagem que use características comuns às comunidades e obtenha melhor acurácia na classificação das novas questões são definidas duas etapas, que são vistas como os objetivos específicos e principais contribuições do presente trabalho:

i Definição das características das questões e dos usuários que são comuns às comunidades de perguntas e respostas. Esse objetivo implica na análise de diferentes comunidades a fim de estabelecer

um conjunto de características que podem ser usados para a classificação de questões; e

ii Definição do processo de seleção das características que irá trazer melhores resultados na classificação das questões. Para classificar as questões é necessária a análise de milhares de variáveis relacionadas a milhares de perguntas. Para computar essas variáveis é necessário processar os textos das questões e dados dos usuários que as postam. Além disso, deve-se verificar se as características são realmente úteis para a classificação. Esse objetivo visa a entrega da abordagem que contempla a classificação das questões em respondível ou não.

Ao final destas etapas, é necessária a execução de experimentos com fontes de dados advindas de diferentes comunidades de perguntas e respostas de forma a se comparar a eficácia da proposta e a variação nos resultados obtidos. Para a validação da proposta, são utilizados conjuntos de dados das comunidades *StackOverflow*[3], *Mathematics*[4] e *Cross Validated*[5], dos quais são mensuradas as métricas de revocação, precisão e *f-value*.

O presente trabalho está organizado como segue. O Capítulo 2 apresenta a descrição dos conceitos relacionados a esta proposta. Em seguida, no Capítulo 3 são discutidos os trabalhos relacionados. No Capítulo 4, a proposta é detalhada e também se apresenta a configuração dos experimentos executados. Logo em seguida, o Capítulo 5 apresenta os resultados e respectivas análises. Finalmente, no Capítulo 6 são apresentadas as conclusões sobre a proposta e possíveis melhorias.

[3]http://www.stackoverflow.com
[4]https://math.stackexchange.com/
[5]https://stats.stackexchange.com/

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritos os principais conceitos necessários para o entendimento da proposta, indicando, dentre certas categorias, quais opções foram utilizadas e porque.

2.1 COMUNIDADES DE PERGUNTAS E RESPOSTAS

Comunidades de Perguntas e Respostas (CQA - Community Question Answering) são *sites* nos quais um grupo de usuários busca soluções para problemas de diversos temas (SRBA; BIELIKOVA, 2016). Exemplos de comunidades mais ativas atualmente são *Yahoo! Answers, StackOverflow e Quora*[1]. Nesses *sites*, há milhões de usuários registrados postando diariamente milhares de novas questões e respectivas respostas. Outro fator importante é que as informações postadas estão visíveis a todos na *web*, o que faz com que o número de visitas a estas páginas seja muito alto.

O procedimento para postar uma questão envolve geralmente 3 passos: (i) escrever um título resumindo a questão, (ii) descrever a questão em todos os detalhes e (iii) identificar as categorias em que a pergunta se encaixa. Essas categorias são pré-definidas nas comunidades e quando necessário, os próprios usuários solicitam a adição de novas categorias. A marcação das questões faz com que as informações se organizem, facilitando para os usuários consultarem os temas de seu interesse, tanto para dúvidas quanto para ajudar a solucionar. No exemplo da Figura 1 há um exemplo de categorias associadas pelo usuário a uma questão.

O conteúdo pode ser qualificado pelos próprios usuários. Porém, a forma como isso ocorre difere em cada comunidade. Uma das formas possíveis é a votação para os textos da dúvida, a qual corresponde ao total de votos positivos ou negativos. Outra forma de avaliação acontece quando uma dúvida recebe várias respostas e o questionador marca uma delas como a melhor resposta.

[1] https://www.quora.com/

2.1.1 Questão Respondida

No processo das comunidades há um passo além do ciclo de postar uma pergunta e receber respostas de outros usuários. Após o recebimento de uma ou mais respostas, fica a critério do questionador escolher qual a melhor resposta. A indicação dessa melhor resposta tem por objetivo indicar aos demais usuários que a dúvida foi sanada, tornando-se assim uma questão respondida.

No âmbito desta proposta, uma questão é considerada respondível quando há a possibilidade de receber ao menos uma resposta independente se é a solução. O fato de haver uma resposta, mesmo que não seja a solução definitiva, significa que se está mais perto de resolver o problema ou ao menos que o autor deve revisar a descrição da sua pergunta. Em algumas comunidades há a possibilidade de inserir comentários para a questão postada. Nesse caso, os comentários não são considerados uma resposta.

2.2 EXTRAÇÃO DE CARACTERÍSTICAS

No contexto deste trabalho, extração está relacionada à *Feature Construction* (MOTODA; LIU, 2002) e não à *Feature Extraction*, porque o objetivo é extrair características dos textos apresentados. O primeiro conceito significa descobrir relações entre as características que produzem informações. Essa construção pode ser alcançada pela transformação dos dados, via hipóteses ou baseando-se no conhecimento existente do domínio (MOTODA; LIU, 2002), como por exemplo verificar o número de questões postadas anteriormente pelo usuário que está postando uma nova questão. Dessa forma, expande-se o conjunto de características da base.

Por outro lado, *Feature Extraction* visa a redução do tamanho de um grupo existente de características. Essa redução pode ser alcançada através de um mapeamento funcional. Nesse mapeamento transforma-se os valores das características. Nesse momento pode ocorrer também a junção de características. Dessa forma, gera-se um novo conjunto menor do que o conjunto inicial de características.

Nas subseções 2.2.1, 2.2.2 e 2.2.3 explica-se o cálculo de 3 dos valores considerados neste trabalho, *Legibilidade*, *Subjetividade* e *Valor TF-IDF*. Esses valores são demonstrados a parte pois os cálculos são mais elaborados que os demais. As demais características são descritas na Subseção 4.1.1.

2.2.1 Legibilidade

O valor atribuído para Legibilidade indica o nível de dificuldade da leitura de um texto. Quanto maior o valor, maior é a dificuldade no entendimento, ou seja, é necessário um grau maior de escolaridade. Os níveis mais baixos são textos compreensíveis até para crianças em idade escolar. Os valores mais altos são textos para estudantes de graduação ou pós-graduação (MCCALLUM; PETERSON, 1982).

Essa característica pode ser chamada também de *Complexidade* devido à dificuldade de entendimento do texto. Asaduzzaman et al. (2013) hipotetizam que quanto mais difícil a leitura e compreensão do texto, mais tempo a questão permanecerá sem resposta. Os experimentos de Chua e Banerjee (2015) e Fong, Zhou e Moutinho (2015) relatam também que questões menos complexas atraem mais respostas.

Para se calcular o valor de legibilidade empregou-se uma série de fórmulas apresentadas na Tabela 2 (MCCALLUM; PETERSON, 1982). As fórmulas que resultam em R analisam a legibilidade de forma geral enquanto G associa um respectivo ano do ensino no sistema norte-americano. O valor final atribuído será o consenso dos valores que mais ocorrem nos resultados das fórmulas.

Tabela 2 – Fórmulas de Legibilidade

Nome	**Fórmula**
Flesch Reading Ease	$R = 206,835 - 84,6 * S/W - 1,015 * W/T$
Flesh-Kincaid	$G = -15,59 + 11,8 * S/W + 0,39 * W/T$
Fog Scale	$G = 0,4 * (W/S + 100 * (C/W))$
Automated Reliability Index	$G = -21,43 + 4,71 * L/W + 0,50 * W/T$
The Coleman Liau Index	$G = -15,8 + 5,88 * L/W - 29,59 * W/T$
Dale-Chall Readability Score	$G = 15,8 * D/W + 0,0496 * W/T$
SMOG Index	$G = 1,0430 * \sqrt{C * (30/S)} + 3,1291$
Linsear Write Formula[2][3]	$R = (3C + (W - C) + W)/S)/2$

Legenda:

- R = Legibilidade
- G = Graduação Necessária para entendimento
- C = Quantidade de Palavras com 3 sílabas ou mais
- D = Número de palavras na lista especificada por Dale Long
- L = número de letras

[2] http://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Linsear_Write.html
[3] https://en.wikipedia.org/wiki/Linsear_Write

- S = número de Sílabas
- T = número de sentenças
- W = número de palavras

Como exemplo de cálculo será usada a seguinte frase de Albert Einstein:

"Do not feel sorry for me. Despite terrible appearances, my life goes on in full harmony; I am entirely devoted to reflection. I resemble a farsighted man who is charmed by the vast horizon and who is disturbed by the foreground only when an opaque object obstructs his view."[4].

As fórmulas foram calculadas para essa citação resultando na Tabela 3. O consenso, valor que mais ocorre nos resultados, indica que o texto é adequado para alguém que estudou ao menos até o 9º ano.

Tabela 3 – Exemplo Cálculo de Legibilidade

Nome	**Resultado**	**Escolaridade Equivalente**
Flesch Reading Ease	8,5	8º/9º Ano
Flesch-Kincaid	9	9º Ano
Fog Scale	17	Graduação
Automated Reliability Index	8,5	3º Ano
The Coleman Liau Index	9	9º Ano
Dale-Chall Readability Score	9	Graduação
SMOG Index	13	7º Ano
Linsear Write Formula	11	11º Ano

2.2.2 Subjetividade

A característica subjetividade diferencia as questões entre subjetivas e objetivas. Os valores são representados entre 0 (muito objetiva) e 1 (muito subjetiva). Uma questão objetiva busca respostas para um problema específico enquanto a subjetividade busca opiniões sobre um determinado assunto (LI et al., 2008). Essa característstica vem sendo usada para a classificação de questões desde a primeira proposta de Yang et al. (YANG et al., 2011). Chua e Banerjee (CHUA; BANERJEE, 2015) caracterizam subjetividade como valores sócio-emocionais, cujas expressões podem obter respostas devido às necessidades de informação e contexto social do questionador. Essa característica também é explorada nos trabalhos de Dror, Maarek e Szpektor (DROR; MAAREK;

[4]http://assets.press.princeton.edu/chapters/s9268.pdf

SZPEKTOR, 2013) e Fong, Zhou e Moutinho (FONG; ZHOU; MOUTINHO, 2015).

Os valores calculados para essa característica resultam da comparação com um léxico de adjetivos que ocorrem em análises de produtos. Cada um dos termos presentes no léxico possui um valor para a subjetividade. A partir da presença desses termos e respectivas pontuações, atribui-se o valor para caracterizar o texto[5] [6]. O valor da subjetividade é calculado de acordo com os adjetivos presentes no texto. Alguns exemplos de pontuação são demonstrados na Tabela 4 onde quanto maior a pontuação maior o valor de subjetividade.

Considerando a frase: "The movie attempts to be surreal by incorporating various time paradoxes, but it's presented in such a ridiculous way it's seriously boring."O texto contém os seguintes termos *surreal* (1.0), *various* (0.5), *such* (0.5), *ridiculous* (1.0) e *seriously boring* (1.0). O valor atribuído para a subjetividade consiste simplesmente na média desses valores resultando em 0.8, caracterizando uma frase subjetiva. Outro exemplo seria o texto "The sky is blue and the birds are singing", contendo um único termo pontuado: *blue* (0.1). Para essa frase a subjetividade é 0.1 caracterizando a frase como objetiva.

Tabela 4 – Exemplo Pontuação Subjetividade

Termo	**Valor**
appreciated	0,1
misplaced	0,2
humble	0,4
unable	0,5
hated	0,7
loved	0,8
happy	1,0

2.2.3 Vetor TF-IDF

O vetor TF-IDF (*Term-Frequency – Inverse Document Frequency*) é uma combinação normalizada de todos os termos presentes em uma coleção de documentos. No contexto deste trabalho, uma coleção consiste de todas as questões associadas a uma categoria de uma dada

[5] https://www.clips.uantwerpen.be/pages/pattern-en#sentiment

[6] https://raw.githubusercontent.com/sloria/TextBlob/dev/textblob/en/en-sentiment.xml

comunidade. A fórmula para calcular o valor atribuído em cada termo do vetor consiste em uma multiplicação de tf (frequência do termo) com idf (inverso da frequência nos documentos)(AGGARWAL, 2015, p. 432).

O valor de tf é calculado por (n_i/n) sendo n_i o número de vezes em que um termo i aparece nos documentos dividido pelo total de documentos n. O cálculo do idf consiste em $log(n/n_i)$, onde n é o total de documentos numa coleção e n_i o total de documentos onde o termo i ocorre. O valor TF-IDF faz com que os termos menos frequentes tenham maior peso no vetor.

2.3 FEATURE SELECTION

A seleção de características (*feature selection*) traduz-se no processo de gerar um subconjunto de características que terão maior relevância ao predizer as classes das questões. Características irrelevantes geram ruído e prejudicam o treinamento dos modelos classificadores. A exclusão desses ruídos se faz através de *feature selection* (AGGARWAL, 2015, p. 287-293). Os três principais tipos de *feature selection* são:

- *Filter model*, onde se estabelece um critério matemático para avaliação da qualidade das características em relação à classe alvo. A partir dos valores calculados é gerado um subconjunto de categorias, as quais tem maior qualidade para o modelo classificador. Sua maior vantagem é a rapidez no processamento. A desvantagem deste tipo de *Feature Selection* é que se desconsidera os relacionamentos entre as variáveis.

- *Wrapper model*, esse tipo de seleção itera sobre todos os possíveis subconjuntos a fim de encontrar qual deles trará melhor resultado. Sua grande desvantagem é que quanto maior o número de características mais processamento será necessário.

- *Embedded model*, o qual tenta definir o melhor subconjunto de características durante o aprendizado do modelo classificador. Durante o treinamento são identificadas quais característiscas melhor contribuem para a acurácia do modelo classificador e elimina-se as demais. Itera-se até que haja o menor conjunto de características no modelo. A desvantagem desse tipo é o fato de a técnica geralmente estar associada a um algoritmo classificador.

Neste trabalho, optou-se pela utilização de *feature selection* baseada em filtros. O tamanho dos conjuntos de características tornam-se

muito grandes ao usar os termos das questões. Por isso, o método de *Feature Selection* escolhido é o método baseado em filtros (*Filter Model*) (AGGARWAL, 2015, p. 288) para separar um subconjunto de características mais relevantes para usar no treinamento. Além disso, o método baseado em filtro independe do algoritmo classificador. Isso evita o ajuste do classificador especificamente ao conjunto de dados usado para o treinamento.

Os critérios experimentados para filtro foram: (i) Correlação de Pearson; (ii) *Information Gain* e (iii) *Gini Gain*. A criação do vetor TF-IDF faz com que o conjunto de características tenha milhares de elementos. Dessa forma, as abordagens *wrapper* e *embedded* exigiriam maior tempo de processamento devido ao grande número de subconjuntos possíveis.

As subseções 2.3.1, 2.3.2 e 2.3.3 apresentam as 3 formas de pesos aplicadas neste trabalho para selecionar as características disponíveis. Para a compreensão dos pesos, os exemplos dos cálculos usam os dados da Tabela 5, que contém 4 questões com apenas 3 características para facilitar os cálculos.

Tabela 5 – Exemplo Simplificado de Cálculo dos Pesos

Questão	**Respondida (R)**	**Número de Dias Como Membro (A)**	**Contagem de URL (B)**	**Tam. Descrição (C)**
1	S	925	0	140
2	S	2874	0	89
3	N	315	0	94
4	N	396	0	469

2.3.1 Correlação de Pearson

O coeficiente de correlação de Pearson tem por objetivo medir o grau de associação entre um par de variáveis X e Y. Esse grau é medido resultando em valores entre -1 e 1. Se a correlação é negativa, significa que, quanto menor o valor de uma variavél X, menor será o valor para a variável Y. O contrário também é válido, quanto maior o valor de X, maior será o valor para Y (HAN; PEI; KAMBER, 2011, p. 95).

Neste trabalho, deseja-se identificar as questões que receberão respostas e também o contrário. Portanto, há somente duas classes alvo (*Questão Respondível* e *Questão Não Respondível*), ou seja, é aplicada uma classificação binária. Sendo assim, quanto maior o valor de

correlação atribuído a uma característica, maior a relação com a classe positiva, nesse caso, *Questão Respondível*. O valor de correlação é calculado conforme Eq. 2.1. Considerando a Tabela 5, n é a quantidade de questões, x_i o valor da característica na questão i, $\hat{x}$ é o valor médio da característica nas questões, y_i é o valor da classe (0 ou 1) da questão i , $\hat{y}$ é o valor médio das classes e std é o desvio padrão.

$$r = \sum_{i=1}^{n} \frac{(x_i - \hat{x}).(y_i - \hat{y})}{(n-1).std(x).std(y)} \tag{2.1}$$

Aplicando a fórmula para os valores da característica *Dias como Membro* na Tabela 5 temos o seguinte resultado:

$$\frac{(925 - 1127,5).(1 - 0,5)}{(4-1).1195,41.0,57735} = -0,049$$

$$\frac{(2874 - 1127,5).(1 - 0,5)}{(4-1).1195,41.0,57735} = 0,422$$

$$\frac{(315 - 1127,5).(0 - 0,5)}{(4-1).1195,41.0,57735} = 0,196$$

$$\frac{(396 - 1127,5).(0 - 0,5)}{(4-1).1195,41.0,57735} = 0,177$$

$$\text{Correlação} = -0,0489 + 0,42178 + 0,196219 + 0,176657 = 0,746$$

Calculando as correlações para as características *Tamanho da Descrição* e *Contagem de URLs* em relação às classes temos os respectivos valores: -0,529 e 0. Com esses valores é possível interpretar que: (i) quanto mais *Dias Como Membro* mais as questões são respondidas; (ii) quanto menor o *Tamanho da Descrição* menos respostas são dadas; e (iii) a *Contagem de URLs* não faz diferença na definição da classe.

2.3.2 Information Gain

O conceito de *Information Gain* envolve a medida de entropia. Na área de Teoria da Informação, entropia é definida como uma forma de se medir o grau de incerteza nas informações. Para se calcular o valor da entropia emprega-se a Eq. 2.2, onde p é a probabilidade do valor i ocorrer. Quanto maior o valor da entropia, maior a necessidade de se misturar características, pois os dados estão distribuídos de forma que não é possível definir uma associação entre valor e classe. Por outro

lado, quanto menor o valor, significa que a característica tem grande influência na definição da classe (HAN; PEI; KAMBER, 2011, p. 336). A Eq. 2.3 representa o cálculo da entropia da característica X em relação à classe Y. O valor n representa o total de registros, S_i é o número de ocorrências de cada valor da característica da questão i dentre o total de registros S. Um exemplo de valor para $\frac{S_i}{S}$ é apresentado na Eq. 2.6. No total há 4 registros na Tabela 5, sendo 2 deles maiores que o ponto de corte e 2 abaixo. Portanto, $\frac{S_i}{S}$ assume os respectivos valores $\frac{2}{4}$ e $\frac{2}{4}$.

$$E(Y) = -\sum_{i=1}^{n} p_i log_2 p_i \tag{2.2}$$

$$E_X(Y) = \sum_{i=1}^{n} \frac{|S_i|}{|S|} * E(X) \tag{2.3}$$

O valor *Information Gain* é calculado para definir qual a variação na entropia que determinada característica causa. Sendo assim, para se calcular o peso *Information Gain* emprega-se a fórmula apresentada na Eq. 2.4 onde E(Y) é a entropia da própria classe, que é subtraída da entropia $E_X(Y)$. A partir desse cálculo depreende-se que quanto menor o valor de entropia de X há maior ganho de informação para a definição da classe Y.

$$IG_X(Y) = E(Y) - E_X(Y) \tag{2.4}$$

Calculou-se nas Equações 2.5, 2.6, 2.7 e 2.8 os pesos de *Information Gain* para o conjunto da Tabela 5. Quando os valores da característica são contínuos, é necessário definir um ponto de corte para o valor da característica. Normalmente, esse valor é definido através de média simples. Porém, nem sempre a média simples resulta na melhor divisão. Por isso, o ponto de corte pode ser ajustado para alcançar menor entropia. Nos exemplos abaixo foi definido 900 para *Dias Como Membro* e 279 para *Tamanho da Descrição*. Os pesos calculados indicam que *Dias Como Membro* e *Tamanho da Descrição* geram ganho de informação para a classificação. Enquanto a entropia de *Contagem de URLs* é igual a entropia da classe Respondida, dessa forma há ganho nulo de informação.

$$E(R) = \frac{2}{4} * log_2 \frac{2}{4} + \frac{2}{4} * log_2 \frac{2}{4} = 1 \tag{2.5}$$

$$\begin{aligned} E_A(R) &= \frac{2}{4} * (\frac{2}{2} * log_2 \frac{2}{2}) \\ &+ \frac{2}{4} * (\frac{2}{2} * log_2 \frac{2}{2}) = 0 \\ IG &= 1 - 0 = 1 \end{aligned} \tag{2.6}$$

$$\begin{aligned} E_B(R) = \frac{4}{4} * (\frac{2}{4} * log_2 \frac{2}{4} + \frac{2}{4} * log_2 \frac{2}{4}) = 1 \\ IG = 1 - 1 = 0 \end{aligned} \tag{2.7}$$

$$\begin{aligned} E_C(R) &= \frac{3}{4} * (\frac{1}{3} * log_2 \frac{1}{3} + \frac{2}{3} * log_2 \frac{2}{3}) \\ &+ \frac{1}{4} * (\frac{1}{1} * log_2 \frac{1}{1}) = 0,689 \\ IG &= 1 - 0,689 = 0,311 \end{aligned} \tag{2.8}$$

2.3.3 Gini Gain

O cálculo do índice de GINI serve para identificar a desigualdade da variação dos valores de uma característica em relação a uma classe. A fórmula para se obter o valor é apresentada na Eq. 2.9, onde p significa a proporção das instâncias pertencentes em cada classe k. Quanto menor o valor do índice maior o poder discriminativo, ou seja, maior a identificação dos valores das características com as classes alvo (HAN; PEI; KAMBER, 2011, p. 341).

$$GI(Y) = 1 - \sum_{i=1}^{k} p_i^2 \tag{2.9}$$

$$GI_X(Y) = \sum_{i=1}^{m} \frac{|S_i|}{|S|} * GI(X) \tag{2.10}$$

$$GIgain_X(Y) = GI(Y) - GI_X(Y) \tag{2.11}$$

Semelhante ao cálculo de *Information Gain*, faz-se a subtração do índice Gini da característica, Eq. 2.10, em relação ao índice Gini da classe para identificar o valor *Gini Gain*, Eq. 2.11. Índices menores

resultam em ganhos maiores, ou seja, as características são relevantes para a classificação das classes. Considerando a Tabela 5, calculou-se os valores para a exemplificação demonstrada nas Equações 2.12, 2.13, 2.14 e 2.15. As características *Dias Como Membro*, *Tamanho da Descrição* e *Contagem de URLs* resultaram respectivamente 0.5, 0.167 e 0. A variação nos valores das características resulta em um ganho maior para a classificação. Enquanto, valores repetidos não auxiliam e podem ser descartados para a criação do modelo classificador.

$$GI(R) = 1 - ((\frac{2}{4})^2 + (\frac{2}{4})^2) = 0,5 \tag{2.12}$$

$$\begin{aligned} GI_A(R) = \frac{2}{4} * (1 - ((\frac{2}{2})^2)) + \frac{2}{4} * (1 - ((\frac{2}{2})^2)) \\ 0 + 0 = 0 \\ GIgain = 0,5 - 0 = 0,5 \end{aligned} \tag{2.13}$$

$$\begin{aligned} GI_B(R) = \frac{4}{4} * (1 - ((\frac{2}{4})^2 + (\frac{2}{4})^2)) = 0,5 \\ GIgain = 0,5 - 0,5 = 0 \end{aligned} \tag{2.14}$$

$$\begin{aligned} GI_C(R) = \frac{3}{4} * (1 - ((\frac{2}{3})^2)) + (1 - ((\frac{1}{3})^2)) \\ + \frac{1}{4} * (1 - ((\frac{1}{1})^2)) \\ 0,333 + 0 = 0,333 \\ GIgain = 0,5 - 0,333 = 0,167 \end{aligned} \tag{2.15}$$

2.4 CLASSIFICAÇÃO DE DADOS

Classificação de dados consiste em um processo para encontrar um modelo que descreve e distingue a estrutura de classes de dados ou conceitos. Os modelos são construídos com base em um conjunto de dados para treinamento. Esses modelos são usados para identificar os elementos que ainda não possuem classes associadas (HAN; PEI; KAMBER, 2011).

Os modelos gerados podem ser representados de diversas formas, tais como regras de classificação, árvores de decisão, redes neurais, den-

tre outros. A saída de um modelo classificador pode ser um rótulo ou um valor numérico. Quando há uma identificação de um valor baseada nas características, faz-se então uma predição. Quando a saída é uma identificação, por um rótulo, há uma classificação. Neste trabalho, os modelos treinados rotulam as questões com as classes *Questão Respondível* e *Questão Não Respondível*, sendo realizada portanto uma classificação.

2.4.1 Validação Cruzada

Validação cruzada é uma técnica de treinamento que 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 (AGGARWAL, 2015, p. 336). Dessa maneira, evita-se que os classificadores se ajustem demais ao conjunto de dados, porque os modelos treinados são testados em k diferentes partes.

2.4.2 Balanceamento de Classes

O problema do balanceamento de classes ocorre quando o conjunto de dados não está igualmente dividido entre as classes. Quando os classificadores treinam conjuntos de dados nessas condições geralmente resultam em uma alta acurácia. Essa acurácia é enganosa pois após o treinamento, os classificadores tendem a classificar os dados com o tipo da classe sobressalente no conjunto. Dessa forma, as demais classes, que são minoria, não são corretamente analisadas pelos classificadores.

Há duas formas de lidar com essa situação (HAN; PEI; KAMBER, 2011, p. 367; 384). A primeira é calcular métricas para a classificação de cada classe do conjunto, como por exemplo, sensitividade e especificidade, representadas nas Eq. 2.16 e 2.17. Para sensistividade, TP é o número de classificações corretas para a classe positiva e P o total de elementos pertencentes à classe positiva no conjuntos de dados. Em especificidade TN corresponde ao número de elementos da classe negativa corretamente classificados e N o número de elementos da classe negativa no conjunto de dados. Essas métricas avaliam respectivamente a classificação dos elementos de cada classe permitindo avaliar se o classificador determina cada classe corretamente.

$$sensitividade = \frac{TP}{P} \tag{2.16}$$

$$especificidade = \frac{TN}{N} \tag{2.17}$$

A segunda maneira de amenizar esse problema é balancear os dados. Para isso existem duas técnicas: (i) *oversampling* onde repete-se randômicamente os dados para a classe com menos registros e (ii) *undersampling* onde sorteia-se uma amostra da classe maior do mesmo tamanho que a classe menor.

Os conjuntos de dados nesse trabalho foram balanceados com a técnica *undersampling*. Além disso, também usou-se as medidas de sensitividade e especificidade para avaliar os resultados dos experimentos apresentados na Seção 4.2 Tabela 11, onde aparecem respectivamento como Revocação para *Questão Respondível* e Revocação para *Questão Não Respondível*.

3 TRABALHOS RELACIONADOS

Neste capítulo, são apresentados e comparados trabalhos relacionados à tarefa de classificação de questões em CQAs. 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 a classificação de que ao menos uma resposta será dada para as questões, que é também o objetivo a ser realizado nesta dissertação. Este capítulo foi dividido em duas subseções para diferenciar os trabalhos relacionados em: (i) estudos que analisaram as características das questões sem respostas e (ii) propostas de classificação para novas questões em respondíveis ou não.

3.1 ANÁLISE DAS QUESTÕES NÃO RESPONDIDAS

Asaduzzaman et al. (2013) estudaram os motivos de uma questão permanecer sem respostas na comunidade *StackOverflow*, que é voltada para tópicos de programação. Nesse trabalho, uma questão não respondida significa que não recebeu respostas até um mês depois da postagem. A partir da análise, os autores buscaram a construção de classificadores para predizer o número de dias que uma questão permanecerá não respondida. Além disso, essa abordagem usa questões publicadas ao menos há um dia. Esse tempo de espera foi definido para poder medir a popularidade da questão com a quantidade de votos recebida. Os resultados obtidos não foram muito satisfatórios, mas as características analisadas serviram como ponto de partida para elaboração de abordagens de classificação de questões.

Logo em seguida, no mesmo ano Saha, Saha e Perry (2013) analisaram toda a base de perguntas e respostas disponível no *StackOverflow* com o objetivo de identificar causas para a falta de respostas. Porém, o conjunto de questões usado nos experimentos era muito desbalanceado, com 90% de questões respondidas contra 10% não respondidas. Para amenizar essa situação, os autores aplicaram alguns filtros até que a proporção ficasse 55% e 45%. 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 usando quatro algoritmos: *Decision Tree, kNN, Naive Bayes* e *Random Forest* com dois grupos de características. No primeiro grupo estavam somente as 6 características com as medi-

das mais altas. No segundo grupo estavam 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 (2015) desenvolveram um *framework* para explicar porque algumas questões são respondíveis e outras não. Nesse trabalho, foram analisados os metadados e conteúdo da questão. Entre as características exploradas estão: popularidade do usuário, participação do usuário, data da questão e número de visualizações. O conjunto de dados utilizado foi composto por 3000 questões sobre Java postadas no *site StackOverflow*. Algumas das características, como completude e subjetividade, foram atribuídas por pessoas com conhecimento em Java. Porém, essa atribuição manual torna-se ineficiente para uma avaliação de uma base maior de questões. Os resultados encontrados contradisseram alguns trabalhos anteriores e ratificaram resultados de outros. Algumas inferências encontradas foram que as questões de usuários novos tendem a ser mais respondidas e questões com descrições breves e poucas *tags* atraem respostas.

3.2 CLASSIFICAÇÃO NO MOMENTO DA POSTAGEM

Os autores Yang et al. (2011) foram os pioneiros na classificação das 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 experimentos para classificação foram feitos com a utilização de quatro algoritmos clássicos: Naive Bayes, Decision Trees, Boosting e SVM.

A base de dados obtida para os experimentos de Yang et al. (2011) foi composta por questões postadas em um único dia. Além disso, esse conjunto de questões era desbalanceado em relação aos tipos *Respondível* e *Não Respondível*. Normalmente, o número de questões respondidas é maior, o que faz com que alguns classificadores, como *Decision Tree* e *Boosting*, não consigam diferenciar as classes corretamente. Por esse motivo, foi feito o balanceamento entre questões respondidas e não respondidas através de sorteio. Os resultados alcançados neste trabalho usando a combinação de tópicos e características teve como *baseline* um vetor TF-IDF das 1000 palavras mais frequentes no *dataset*. Apesar dos resultados serem melhores que o *baseline*, os autores declaram que são resultados ainda ruins para uso prático.

Baseando-se no trabalho de Yang et al. (2011), Dror, Maarek e

Szpektor (2013) buscaram algo ainda mais específico: predizer quantas respostas uma nova questão vai receber. Para isso trabalharam também na tarefa de indicar primeiro se a questão vai ser respondida. O trabalho executava dois passos: primeiro a classificação e então a predição para o número de possíveis respostas. A base de questões usada tinha 10 milhões de questões do *site* Yahoo! Answers e era desbalanceada contendo muito mais questões respondidas do que não respondidas. Em particular, a abordagem proposta (DROR; MAAREK; SZPEKTOR, 2013) 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 irá ser vantajosa quando as categorias estiverem organizadas em árvore.

O estudo que mais se aproxima da abordagem proposta nesta dissertação é o de Fong, Zhou e Moutinho (2015). Os autores propuseram a classificação das questões usando somente características disponíveis no momento da postagem para predizer se uma questão vai receber uma resposta. Nessa tarefa foram aplicados classificadores *data mining* tradicionais e também *data stream mining* para treinamento contínuo do modelo classificador. A entrada de dados continha 3000 questões Java divididas igualmente entre respondidas e não respondidas. O desempenho da proposta foi avaliado com a acurácia da classificação. Os *baselines* definidos foram os treinamentos com todas as características extraídas e com o uso de *CFS - correlation-based feature selection* (HALL, 1999). A proposta dos autores foi usar *feature selection* com aplicação de meta heurísticas, chamada *Accelerated Particle Swarm Optimization* (APSO) (YANG; DEB; FONG, 2011). A técnica APSO 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 em relação aos *baselines* para todos os classificadores utilizados. Porém, uma desvantagem dessa proposta é não se ter garantia dos melhores resultados, visto que, para conjuntos grandes de características, há muitas possibilidades de subconjuntos. Outro ponto fraco é o tempo de processamento, uma vez que são necessárias muitas iterações até alcançar um ponto de parada.

Os trabalhos mencionados nesta subseção são usados como *baselines* nos experimentos de avaliação da proposta.

3.3 ANÁLISE COMPARATIVA

Nas Tabelas 6 e 7 são apresentadas comparações entre as configurações das abordagens dos trabalhos citados e o que é proposto nesta dissertação. Todos os projetos mencionados trabalham com características específicas de cada comunidade, ou seja, nem todas as características existem nas demais. Essas diferenças ocorrem por causa das diferentes funcionalidades concedidas aos usuários. Alguns exemplos são os votos para questões consideradas relevantes, inserção de código-fonte na descrição da questão, reputação dos usuários entre outras. Nesta dissertação, propõe-se uma abordagem que resulta um modelo classificador treinado somente com características comuns às comunidades e que estejam disponíveis no momento do envio da pergunta.

Tabela 6 – Comparação de Trabalhos Relacionados Parte 1

Trabalho	**Seleção de Características/Tipo**	**Vetor de Palavras**	**Classificação Por Categoria**	**Comunidade**
(YANG et al., 2011)	Não	Tópicos	Não	Yahoo! Answers
(DROR; MAAREK; SZPEKTOR, 2013)	Não	Tokens	Parcial	Yahoo! Answers
(ASADUZZAMAN et al., 2013)	Não	Não	Não	StackOverflow
(SAHA; SAHA; PERRY, 2013)	Não	Não	Não	StackOverflow
(CHUA; BANERJEE, 2015)	Não	Não	Sim	StackOverflow
(FONG; ZHOU; MOUTINHO, 2015)	Sim/Wrapper	Não	Sim	StackOverflow
(KNOCHENHAUER; DORNELES; WIVES, 2018)	Sim/Filtro	TF-IDF	Sim	Genérica

A maioria dos estudos não considerou os termos usados na descrição das questões. Yang et al. (2011) usou as palavras para extração de tópicos, porém, o uso de tópicos não trouxe resultados para uso prático. Dror, Maarek e Szpektor (2013) usaram a presença ou não dos termos na classificação, mas sem especializar o treinamento em cada categoria.

Apenas uma abordagem (FONG; ZHOU; MOUTINHO, 2015) empre-

Tabela 7 – Comparação de Trabalhos Relacionados Parte 2

Trabalho	**Classificadores**	**Qtd. Características**	**Critérios**	**Balanceamento de Classes**
(YANG et al., 2011)	Naive Bayes; Árvores de Decisão; AdaBoost; SVM(SMO)	7	-	Sim
(DROR; MAAREK; SZPEKTOR, 2013)	Regressão Logística	15	-	Não
(ASADUZZAMAN et al., 2013)	Random Forest; Árvores de Decisão	9	-	Não
(SAHA; SAHA; PERRY, 2013)	Naive Bayes; KNN; Árvores de Decisão; Random Forest	6	-	Sim
(CHUA; BANERJEE, 2015)	Regressão Logística	49	-	Sim
(FONG; ZHOU; MOUTINHO, 2015)	Hyperpipes; Naive Bayes; BayesNet; Árvores de Decisão; Random Forest; SVM; Redes Neurais; Hoeffding Tree; Random Hoeffding Tree; Hoeffding Option Tree; Naive Bayes Updatable; Active; Adwin; Kstar	24	-	Sim
(KNOCHENHAUER; DORNELES; WIVES, 2018)	Naive Bayes; SVM; Regressão Logística; Hyperpipes	20	Correlação de Pearson; Information Gain; GINI Gain	Sim

gou seleção de características para diminuir o tamanho dos conjuntos treinados, o que gerou melhora na acurácia dos modelos. Nesse trabalho foi usada uma técnica *swarm* onde os subconjuntos são treinados até que se alcance um ponto de parada, como por exemplo um determinado nível de acurácia. Essa técnica para seleção de características faz com que se elimine o ruído das propriedades irrelevantes para a classificação.

Outra peculiaridade é classificar ao mesmo tempo questões de todas as categorias, ou seja, misturar todos os assuntos no treinamento dos modelos classificadores. Isso ocorre em (YANG et al., 2011; DROR; MAAREK; SZPEKTOR, 2013; ASADUZZAMAN et al., 2013; SAHA; SAHA; PERRY, 2013). Acredita-se que o treinamento de um modelo classificador para cada categoria adicionando um vetor TF-IDF traz melhor acurácia, pois os termos mais pertinentes aos assuntos das categorias serão ressaltados.

Nos trabalho citados foram testados 17 diferentes algoritmos classificadores. Dentre eles, o que mais aparece é Árvores de Decisão em 4 dos trabalhos. Além desse, os algoritmos mais usados foram Naive Bayes e Random Forest aparecendo em 3 dos 6 trabalhos. As mais variadas opções foram testadas, desde classificação em *cluster* com *KNN* a classificadores incrementais, que permitem aperfeiçoamento de um modelo já treinado, como o algoritmo *Kstar*.

Percebe-se outra diferença dos trabalhos que é o número de características usadas. Não há um crescimento no número, pois cada trabalho adaptou o conjunto de características conforme o contexto. Por exemplo, para Saha, Saha e Perry (2013) o ideal era reduzir ao máximo restando apenas 6 de 12 características experimentadas inicialmente. Já no estudo de Chua e Banerjee (2015) o número foi mais alto pois cada dia da semana e hora do dia eram considerados como característica das questões.

Quanto ao balanceamento de classes, 4 dos 6 trabalhos decidiram balancear as classes das questões, pois um grande desbalaceamento prejudicaria o treinamento dos modelos. Por fim, nenhuma das abordagens empregou *feature selection* do tipo filtro. Por isso, nenhuma das propostas possui valor associado a coluna Critérios.

O próximo capítulo detalha a proposta, quais as características foram aproveitadas dos trabalhos relacionados, como será feita a seleção de características e descreve os experimentos realizados para validação da proposta.

4 WANQA

Este capítulo apresenta a abordagem proposta, descrevendo uma visão geral do processo de classificação, apresentando as características extraídas das questões e como estas são selecionadas para a definição do modelo classificador resultante. Ainda neste capítulo é apresentada a configuração da execução e avaliação dos experimentos, assim como os resultados obtidos.

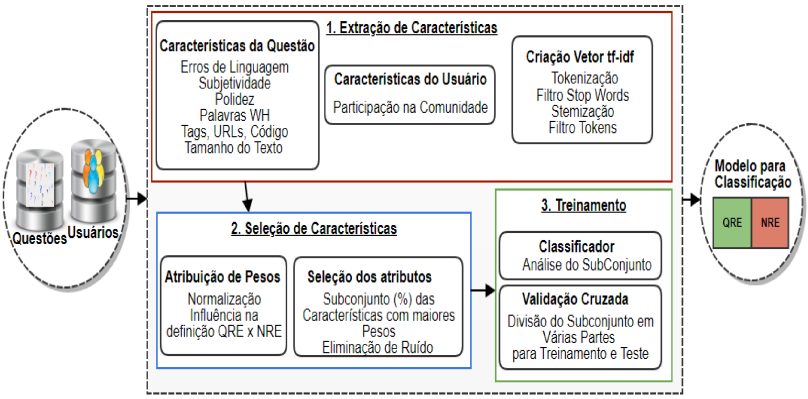
4.1 VISÃO GERAL

O nome WANQA é uma abreviação para *Weighting Analysis for New Question Answerability*, respectivamente em português para Análise de Pesos para Respondibilidade de Nova Questões. A abordagem proposta para classificar as questões no momento da postagem envolve 3 etapas 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 para o processamento, são recebidos os dados de questões e também dos usuários que as postaram. Na primeira etapa, são calculados os valores das características nas questões da entrada e cria-se um vetor TF-IDF para o conjunto de questões. 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 o algoritmo de classificação especificado, o qual gera um modelo treinado.

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 de questão possui características relevantes para o recebimento de respostas. Assim como, há também características que causam 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*. Nesta abordagem, é empregada a validação cruzada

Figura 2 – Fluxo Criação do Modelo para Classificação de Questões.



do tipo *k-fold*. Após esse treinamento são gerados o modelo treinado e os resultados contendo o número de classificações corretas e incorretas. O modelo classificador resultante é o que será efetivamente aplicado nas novas questões postadas para classificá-las em QRE ou NRE. Cada uma das etapas mostrada na Figura 2 é detalhada a seguir.

Para auxiliar o entendimento da abordagem foram adicionados os Algoritmos 1, 2 e 3 representando as 3 etapas da abordagem. O Algoritmo 4 representa o processo todo englobando as 3 etapas.

4.1.1 Extração de Características

Nesta primeira etapa, são extraídas 20 características relacionadas às questões e à descrição do perfil dos usuários. Na Tabela 8 estão descritas todas as características extraídas da nova questão. A definição de quais características devem ser extraídas foi feita com base nos trabalhos relacionados e considerando os dados que estão disponíveis no momento da postagem da questão. Além disso, para que a abordagem não seja dependente de uma comunidade específica, foram analisadas características disponíveis na grande maioria das comunidades. Alguns exemplos de características que foram desconsideradas: número de visualizações da postagem, idade da questão e reputação do usuário.

Em relação aos usuários, é obtido um total de 4 característi-

cas: número de dias como membro, número de questões postadas pelo usuário, número de respostas postadas pelo usuário e proporção de respostas em relação ao número de perguntas postadas. Para a extração do conjunto de características das questões é necessário obter Título, Descrição, Data e Categorias de cada uma delas.

A Tabela 8 apresenta as 16 características extraídas das questões. Dentre elas, está o *Tamanho do Título* de uma questão, que é relevante pois resume o questionamento em poucas frases. Outra característica extraída é *Título Inicia com Palavra WH* 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 através de uma contagem de erros gramaticais e ortográficos. Essas falhas no texto podem dificultar o entendimento do problema apresentado e por consequência serem ignorados pelos demais usuários. Além do título, também conta-se o *Tamanho da Descrição* da questão que 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 foi inserido 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.

O dia e hora da postagem, representados por *Seno Dia, Cosseno Dia, Seno Hora e Cosseno Hora*, são características que indicam comportamentos dos usuários. O período escolhido para postagem pode influenciar no alcance da questão aos demais membros. Há diferentes formas de usar esses valores, para essa proposta foi escolhida a forma Seno e Cosseno. *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 visam a separação das 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; FONG, 2016). O cálculo das características relacionadas ao Dia e Hora da postagem resultam nos valores apresentados nas Figuras 3 e 4.

Em alguns trabalhos relacionados, como em Chua e Banerjee (2015) e Fong, Zhou e Moutinho (2015), alguns valores de características mencionadas foram atribuídos manualmente. Porém, no pre-

Tabela 8 – Características Extraídas das Questões

Característica	**Descrição**	**Uso em Trabalhos Anteriores**
Tamanho do Título	Quantidade de Termos no Título	6
Tamanho da Descrição	Quantidade de Termos na Descrição	6
Título inicia com palavra WH	Valor booleano indicando se o título da questão começa com uma palavra WH (*what, why, when, who, which, how, whose, whom*)	1
Erros de Linguagem	Quantidade de erros gramaticais encontrados na descrição da questão.	2
Há Código na Descrição	Valor booleano indicando se há código de programação na questão.	3
Legibilidade	Facilidade/dificuldade de entendimento do texto apresentado. Valor calculado com base no consenso do cálculo de várias fórmulas da literatura apresentadas no item 2.2.1	3
Subjetividade	Valor numérico indicando se a postagem busca soluções ou opiniões, pode receber valor no intervalo de 0 (muito objetiva) a 1 (muito subjetiva). (Descrição no item 2.2.2)	4
Polidez	Quantidade de palavras de gentileza *(thank, thanks, please, could, would, help)* utilizadas na descrição.	3
Seno e Cosseno do Dia	Valor calculado pelas fórmulas $seno(2\pi * dia/7)$ e $cosseno(2\pi * dia/7)$ separando o valor dia em dois tipos de classes (ZHOU; FONG, 2016)	1
Seno e Cosseno da Hora	Valor calculado pelas fórmulas $seno(2\pi * hora/24)$ e $cosseno(2\pi * hora/24)$ separando o valor hora em dois tipos de classes (ZHOU; FONG, 2016)	1
Contagem de *tags*	Quantidade de categorias em que a questão foi marcada. Identifica a abrangência da questão em relação aos usuários solucionadores.	3
Contagem de URL	Quantidade de URLs informadas na descrição. Esse valor mostra se houve uma pesquisa prévia na tentativa de resolver o problema.	1
Contagem de palavras WH no título	Quantidade de Palavras WH no título	2
Contagem de palavras WH na descrição	Quantidade de Palavras WH na descrição	2

Figura 3 – Seno e Cosseno do Dia

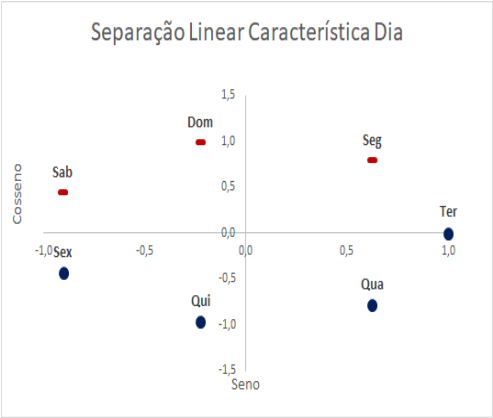
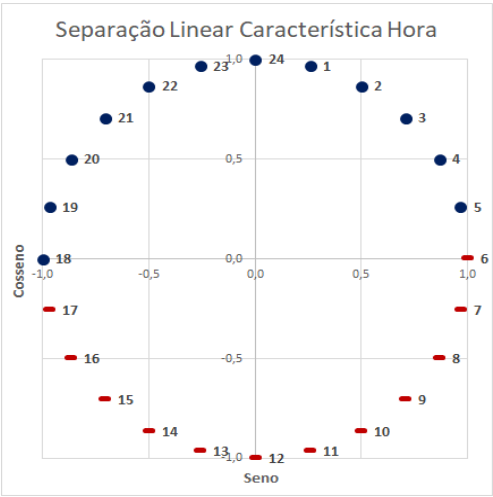


Figura 4 – Seno e Cosseno da Hora



sente trabalho, esses valores são obtidos com a utilização de *scripts*[1][2][3], que calculam *Subjetividade, Legibilidade e Erros de Linguagem.* Dessa forma, remove-se a limitação no número de questões usadas para treinar

[1] https://pypi.python.org/pypi/language-check
[2] https://pypi.python.org/pypi/textstat/0.3
[3] https://pypi.python.org/pypi/textblob

um modelo classificador, visto que, o processamento via *scripts* atribui os valores rapidamente. Os detalhes para a determinação dos valores *Subjetividade e Legibilidade* estão demonstrados nas Subseções 2.2.1, 2.2.2 do Capítulo 2. Na Tabela 9 são apresentadas características que foram usadas nos trabalhos anteriores mas que foram descartadas na abordagem proposta.

Após a extração das características, faz-se um pré-processamento no texto removendo os termos muito comuns (*stop words*), reduz-se os termos à sua raiz (*stemming*) e filtra-se os termos com no mínimo 2 caracteres. Cria-se um vetor TF-IDF (AGGARWAL, 2015, p. 432) com todos os termos encontrados nas questões e seus respectivos pesos. Esse vetor contém uma combinação de todos os termos presentes na coleção de questões recebida como entrada. Todos os valores dos termos são computados para todas as questões. 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 classificador. 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 5 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 termos.

O Algoritmo 1 representa a etapa extração de características. Como entrada recebe-se os dados das questões e dos usuários. Nesse algoritmo, nas linhas 4 a 27 são processados os valores referente as questões postadas. Nas linhas 28 a 30 calcula-se para os usuários, que postaram as questões, o único atributo que exige algum cálculo, *Proporção de Respostas em Relação ao Número de Questões Postadas*. O fim do algoritmo se dá com o retorno do conjunto completo de características para o conjunto de entrada recebido.

Tabela 9 – Características Descartadas

Característica	**Descrição**	**Motivo Descarte**
Categoria Pai	Nome da Categoria Pai onde a Categoria da questão está enquadrada. Exemplo: Esportes (Categoria Pai) e Futebol (Categoria da Questão)	Esse tipo de característica só ocorre em comunidades que tem suas categorias estruturadas hierarquicamente. Porém, essa estrutura não é comum a maioria das comunidades.
Pontuação do Usuário	Quantidade de votos que um usuário recebe ao postar na comunidade.	Não há consenso para esse tipo de funcionalidade. Em alguns casos não é possível votar negativo. Em outros não há a possibilidade de voto. Exemplos: Fluther.com[4], TheAnswerBank[5] e Microsoft Community [6]
Marcação como Favorita	Quantidade de vezes que uma questão foi salva como favorita por um usuário	Característica posterior ao momento da postagem e a funcionalidade não está disponível na maioria das comunidades. Exemplo: BlurtIt[7] e AnswerBag[8]
Número de Edições	Quantidade de vezes que um texto foi editado pelo autor ou outro membro da comunidade	Característica posterior ao momento da postagem e a funcionalidade não está disponível na grande maioria das comunidades.
Número de Visualizações	Quantidade de vezes que a página da questão foi acessada	Característica posterior ao momento da postagem
Número de Comentários	Quantidade de comentários postados para a questão	Característica posterior ao momento da postagem
Número de Palavras Complexas	Quantidade de palavras complexas no texto	Esse valor já é analisado através da característica *Legibilidade*

[4] https://www.fluther.com [5] https://www.theanswerbank.co.uk/
[6] https://answers.microsoft.com
[7] https://www.blurtit.com/ [8] http://www.answerbag.com/

Algoritmo 1: Algoritmo WANQA - Etapa Extração

```
   Entrada: Questões(Q), Usuários(U)
   Saída: Conjunto de Características C
 1 início
 2     C = Q ⋃ U;
 3     W; // W=VetorPalavras
 4     para cada questao q ∈ C faça
 5         q.tamDescricao = contaPalavras(q.descricao);
 6         q.tamTitulo = contaPalavras(q.titulo);
 7         q.tituloIniciaComWH = verificaInicioTituloWH(q.titulo);
 8         q.palavrasWHDescricao = contaPalavrasWH(q.descricao);
 9         q.palavrasWHTitulo = contaPalavrasWH(q.titulo);
10         q.polidez = contaPalavrasEducadas(q.descricao);
11         q.temCodigoFonte = verificaCodigoFonte(q.descricao);
12         q.subjetividade = calculaSubjetividade(q.descricao);
13         q.errosDeLinguagem = contaErrosDeLinguagem(q.descricao);
14         q.legibilidade = calculaLegibilidade(q.descricao);
15         q.senoDia = calculaSenoDia(q.datadapostagem);
16         q.cossenoDia = calculaCossenoDia(q.datadapostagem);
17         q.senoHora = calculaSenoHora(q.datadapostagem);
18         q.cossenoHora = calculaCossenoHora(q.datadapostagem);
19         q.qtdTags = contaTags(q.tags);
20         q.qtdURLS = contaURLs(q.descricao);
21         q.descricao = lowercase(q.descricao);
22         t = tokenize(q.descricao);
23         t = removeStopWords(t);
24         t = stem(t);
25         t = filterByLength(t);
26         W.adicionaTermos(t);
27     fim
28     para cada usuario u ∈ C faça
29         u.proporcaoRespostasPerguntas=u.qtdRespostas/u.qtdPerguntas;
30     fim
31 fim
32 retorna C ;
```

4.1.2 Seleção de Características

Esta etapa tem por objetivo selecionar o subconjunto de características que mais influenciam na definição das classes das questões. O tamanho do conjunto gerado na etapa anterior pode ser muito grande devido à criação do vetor TF-IDF. Para diminuir o tamanho do conjunto de características no modelo final, aplica-se o método de *Feature Selection* baseado em filtros.

Neste trabalho, a relevância de uma característica é dada pela atribuição de pesos que relacionam os valores das características com as classes *QRE* e *NRE*. Quanto maior o peso atribuído, maior a relevância da característica na classificação.

A seleção de características baseada em filtros inicia com a atribuição de pesos. Essa atribuição indica um valor que determina qual o

Figura 5 – Exemplo de valores TF-IDF atribuídos

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 6 – Exemplo de pesos atríbuídos

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

relacionamento da característica com as classes das questões, no caso *QRE* e *NRE*. Há diferentes formas para calcular esses pesos. Neste trabalho são testadas 3 técnicas diferentes: *Information Gain*, Correlação de Pearson e *GINI Gain*, cuja eficácia é apresentada e comparada experimentalmente. A Figura 6 apresenta um exemplo de pesos atribuídos a um conjunto de características.

Como a produção dos modelos classificadores é feita para cada categoria de questão, as características têm diferentes pesos em cada treinamento. Por exemplo, a característica *Há código na descrição* terá um peso maior para as questões que envolvam linguagens de programação, mas para dúvidas relacionadas a esportes, isso pode ser irrelevante. Sendo assim, a filtragem visa reduzir os ruídos que 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 classificador. A seleção de um

subconjunto tem por objetivo, também, aumentar a acurácia na classificação.

O Algoritmo 2 demonstra como a execução desta etapa. Recebe-se o conjunto processado na etapa anterior e também o tamanho do subconjunto a ser filtrado. Nas linhas 3 a 6 calcula-se os pesos para todas as características. Na linha 7 ordena-se o vetor e por fim faz-se o corte do subconjunto com melhores características.

Algoritmo 2: Algoritmo WANQA - Etapa Seleção

```
   Entrada: Conjunto de Características (C), QtdFiltro(F)
   Saída: Subconjunto de Características S
1  início
2  |  P; // P=VetorPesos
3  |  para cada caracteristica c ∈ C faça
4  |  |  c = normaliza(c);
5  |  |  P.calculaPeso(c);
6  |  fim
7  |  P.ordenaVetor();
8  |  S = separaMelhoresCaracteristicas(F);
9  fim
10 retorna S ;
```

4.1.3 Treinamento do Modelo Classificador

O último passo resulta na criação do modelo e consiste no treinamento do conjunto de características. Nessa etapa, são utilizados algoritmos de classificação, como, por exemplo, SVM. Neste trabalho foram testados 4 algoritmos classificadores: SVM, Naive Bayes, Regrssão Logística e *Hyperpipes*. Para evitar *overfitting*, ou seja, que o classificador não se ajuste demais ao conjunto de questões do treinamento, a técnica *k-fold* de validação cruzada foi executada com $k = 10$. Dessa forma, o algoritmo classificador executa 10 rodadas dividindo os dados em 10 partes (*folds*) sendo 9 para treinamento e 1 para validação. O valor 10 é considerado uma escolha típica, sendo recomendado para medir acurácia devido a relativa baixa variância e ajuste do classificador aos dados (HAN; PEI; KAMBER, 2011, p. 370).

Após esse treinamento são gerados o modelo treinado e os resultados contendo o número de classificações corretas e incorretas. O modelo treinado é o que será efetivamente aplicado nas novas questões postadas para classificar as questões em QRE ou NRE. A última etapa ocorre no Algoritmo 3 que recebe o subconjunto obtido na etapa anterior e também a quantidade k de validações para o treinamento do modelo. Primeiramente, na linha 2, divide-se o conjunto em k partes. Logo em seguida, nas linhas 5 a 12, executa-se o treinamento via vali-

dação cruzada separando o conjunto de treino e o conjunto para testes. Armazena-se o resultado e a iteração continua até se atingir o número de validações. Por fim, nas linhas 13 e 14, o modelo final é gerado e o resultado final é obtido a partir dos resultados das iterações. A saída dessa etapa representa também o resultado do processo todo, que é o modelo treinado.

Algoritmo 3: Algoritmo WANQA - Etapa Treinamento

```
   Entrada: Subconjunto de Características S, NúmValidações(k)
   Saída: Modelo Classificador M
 1 início
 2     CV; // CV=ValidaçãoCruzada
 3     K = particionaConjunto(k, S);
 4     x = 1;
 5     enquanto x < k faça
 6         Ktr = separaConjuntoTreino(K, k − 1, x);
 7         Kte = Ktr(x);
 8         Mx = treinaModelo(ktr);
 9         Mresx = testaModelo(M, Kte);
10         CV.adicionaResultado(Mx, Mresx);
11         x = x + 1;
12     fim
13     M = CV.geraModeloFinal(S);
14     Mres = CV.geraMakroResultado();
15 fim
16 retorna M ;
```

Algoritmo 4: Algoritmo WANQA

```
   Entrada: Questões(Q), Usuários(U), QtdFiltro(F), NúmValidações(k)
   Saída: Modelo Classificador M
 1 início
       // Etapa Extração de Características
 2     C = Etapa1(Q, U);
       // Etapa Seleção de Características
 3     S = Etapa2(C, F);
       // Etapa Treinamento
 4     M = Etapa3(S, k);
 5 fim
 6 retorna M ;
```

4.1.4 Exemplo de Execução

Esta seção tem por objetivo demonstrar um exemplo simplificado de como a proposta é executada. Para exemplificar a execução suponha-se um conjunto de entrada com 50 questões Java. Esse conjunto está balanceado entre as classes QRE e NRE contendo 25 questões cada. Na primeira etapa extrai-se os valores das 20 características listadas na Tabela 8. Para detalhar a extração das características é demonstrado abaixo os dados para uma questão e respectivos valores extraídos são

apresentados na Tabela 10.

- Título: *Refactoring away labeled loops*

- Descrição: *After I was convinced that labeled breaks/continues are a total "nono" over here, I need help to remove the label out of my code. I have a square matrix and a vector that has the same length. The vector has already some values in it an depending on the values in the matrix the vector is changed in the loop. I hope, the code-fragment is basically understandable...*

```
vectorLoop:
for( int idx = 0; idx < vectorLength; idx++) {
if( conditionAtVectorPosition( v, idx ) ) continue vectorLoop;
matrixLoop:
for( rowIdx = 0; rowIdx < n; rowIdx++ ) {
if( anotherConditionAtVector( v, rowIdx ) ) continue matrix-
Loop;
if( conditionAtMatrixRowCol( m, rowIdx, idx ) ) continue vec-
torLoop;
}
setValueInVector( v, idx );
}
```

  Please convince me, that there is a more readable/better version without the labels.

- Tags: *java, refactoring* e *label*

- Data da Postagem: 19/08/2008 07:42:02

Ainda na primeira etapa, cria-se o vetor TF-IDF associando todos os termos do conjunto para cada questão. Na Figura 5, são demonstrados alguns exemplos. O vetor criado para o conjunto do exemplo contém 390 termos. Sendo assim, o conjunto gerado na primeira etapa contém no total 410 características.

O conjunto extraído é usado na segunda etapa do processamento onde os valores são normalizados e os pesos de cada característica são calculados. Como exemplo, usou-se Correlação de Pearson para definição dos pesos, gerando como saída uma listagem como a apresentada na Figura 7. A partir dos pesos atribuídos faz-se então a seleção do subconjunto contendo um certo número de características melhores ranqueadas. Nesse momento, ainda não se sabe qual a quantidade ideal,

Tabela 10 – Exemplo Características Extraídas de uma Questão

Característica	**Valor**
Tamanho do Título	4
Tamanho da Descrição	112
Título inicia com palavra WH	0
Erros de Linguagem	6
Há Código na Descrição	1
Legibilidade	14
Subjetividade	0,458
Polidez	2
Seno e Cosseno do Dia	0,975 e -0,223
Seno e Cosseno da Hora	0,966 e -0,259
Contagem de *tags*	3
Contagem de URL	1
Contagem de palavras WH no título	0
Contagem de palavras WH na descrição	0
Dias Como Membro	0
Número de Questões Postadas	0
Respostas Postadas	0
Proporção de Respostas/Perguntas	0

esse valor varia para cada categoria de questão que será treinada. Para fins de exemplo foi filtrado 10% das características.

Após a seleção do subconjunto, acontece a terceira etapa onde se usa um algoritmo classificador, como SVM, Árvores de Decisão, dentre outros. Esse classificador treina sob o conjunto recebido para definir quais valores das características tem relação com as classes QRE e NRE. Para o exemplo foi usado o algoritmo Naive Bayes.

No final da terceira etapa, é gerado como saída o modelo classificador que possui as suas propriedades de acurácia, precisão, revocação para que seja aplicado em novas questões. Para o conjunto do exemplo, o modelo resultante possui uma acurácia de 92%, cujos detalhes são vistos na Figura 8. O modelo resultante errou na classificação de apenas 4 das 50 questões. Dessas 4, 2 são da classe positiva *Questão Respondível* e 2 da classe negativa *Questão Não Respondível*. Nos valores apresentados todas as métricas de Precisão e Revocação são 92% pois a classificação ocorreu de forma igual para as duas classes de questões.

Figura 7 – Exemplo de Pesos por Correlação de Pearson

attribute	weight ↓
cosTime	0.370
Subjectivity	0.318
http	0.311
give	0.311
android	0.295
app	0.293
r_e_q_count	0.286
read	0.281

Figura 8 – Exemplo de Resultados Modelo Classificador

Acurácia: 92.00%

	true N	true Y	class precision
pred. N	23	2	92.00%
pred. Y	2	23	92.00%
class recall	92.00%	92.00%	

4.2 EXPERIMENTOS

Esta seção descreve os experimentos realizados para comparar a eficácia da proposta. Para isso, são apresentados as fontes e dados usados, as formas de atribuição de pesos às características, os classificadores usados para treinar o modelo baseado nas características com maiores pesos e as medidas usadas para apresentação dos resultados dos modelos gerados.

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 CQA; e (iii) investigar a 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) (BAEZA-YATES; RIBEIRO-NETO, 2008) para as duas classes possíveis, QRE e NRE. O cálculo das métricas é feito com as Equações 4.1, 4.2, 4.3 e 4.4, onde TP, TN, FP e FN significam respectivamente verdadeiros positivos e negativos, e falsos positivos e negativos. As questões usadas para a criação do modelo classificador já estão identificadas se receberam ou não respostas. No contexto desse trabalho, verdadeiros positivos são as questões classe *QRE* classificadas corretamente e verdadeiros negativos são as questões classe *NRE* classificadas corretamente. Os falsos positivos e falsos negativos são as questões classificadas erroneamente, respectivamente, *QRE* e *NRE*. Assim, é possível verificar o desempenho do modelo classificador comparando as atribuições de classe nos testes com as classes reais das questões.

$$A = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

$$P = \frac{TP}{TP + FP} \tag{4.2}$$

$$R = \frac{TP}{TP + FN} \tag{4.3}$$

$$F = 2 * \frac{P * R}{P + R} \tag{4.4}$$

A abordagem proposta foi comparada com 3 *baselines*. O primeiro (B1) é a implementação da proposta de Fong, Zhou e Moutinho (2015) que combina uma técnica de *swarm* e *feature selection* para otimizar a acurácia treinando vários subconjuntos. A implementação foi necessária pois não há o código nem base de dados disponibilizados no texto publicado. Para implementar o *baseline* B1 foram consultados os artigos de Fong et al. (2014), Fong, Yang e Deb (2013) e Yang, Deb e Fong (2011). Os outros dois *baselines* são variações da proposta apresentada. No *baseline* B2 extra-se as características mas não cria um vetor TF-IDF e não se aplica a etapa Seleção de Características. No *baseline* B3 adiciona-se o vetor TF-IDF, mas também sem aplicar a etapa Seleção de Características, por isso equivale às estratégias adotadas por Yang et al. (2011) e Dror, Maarek e Szpektor (2013). Nesses trabalhos usou-se os termos usados nas questões para, respectivamente, a criação de tópicos e análise da presença desses termos nos textos.

Para avaliar estatisticamente, aplicou-se o teste t para duas amos-

tras pareadas com as seguintes hipóteses: H0: em média as abordagens WANQA e *Baseline* X tem mesma acurácia e H1 (em média, a abordagem WANQA tem maior acurácia que o *Baseline* X).

Os trabalhos relacionados operaram com dois tipos de configurações para as bases de questões. A primeira treinando questões de todas as categorias misturadas e a segunda apenas com questões de um único tópico. Neste trabalho, são experimentadas 3 bases de questões separadamente, cada uma delas referente a uma categoria de uma comunidade. O objetivo ao usar essas 3 bases de questões é analisar o comportamento dos modelos classificadores treinados com questões de diferentes comunidades.

As questões utilizadas foram obtidas nas categorias Java, Álgebra linear e Regressão, provenientes das respectivas comunidades *StackOverflow*, *Mathematics* e *Cross Validated*. As comunidades citadas pertencem ao mesmo grupo, *StackExchange*[9], porém, os assuntos tratados e os membros são distintos uma das outras. Os conjuntos de questões foram montados de forma que as classes QRE e NRE tivessem um número semelhante de registros sendo no total 3000 sobre Java, 2000 de Regressão e 2000 de Álgebra Linear. Na prática há muito mais *QRE*s do que *NRE*s. Na data de realização deste trabalho, a distribuição das questões sem respostas está como demonstrado na Tabela 1. 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 às características das questões não respondidas.

A etapa *Seleção de Características* foi testada com 3 diferentes alternativas para cálculo dos pesos: correlação de Pearson, *Information Gain* e *GINI Gain*. O objetivo ao usar 3 cálculos de pesos diferentes é tentar identificar se há algum tipo de peso que melhor identifica o relacionamento das variáveis com as classes. Para avaliar a acurácia foram testados 9 tamanhos de subconjuntos: 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80% e 90%. Assim, pode-se verificar se há melhora nos resultados quando se inclui ou retira as caraterísticas com menores pesos.

O *Treinamento dos Modelos* foi executado com 4 algoritmos de classificação já utilizados em trabalhos relacionados: SVM, Naive Bayes (NB), *Hyperpipes* (HP) e Regressão Logística (RL). A proposta deste trabalho é representada na Subseção 5 com as siglas P.PC, P.IG e P.GI, onde PC, IG e GI correspondem respectivamente a Correlação de Pearson, *Information Gain* e *GINI Gain*.

[9] https://stackexchange.com/

A combinação dos conjuntos de questões, algoritmos classificadores, pesos e tamanho de subconjuntos forma um total de 360 configurações diferentes experimentadas em busca da melhor acurácia.

As questões usadas nos experimentos foram obtidas através da página StackExchange Data Explorer[10]. A extração das características, exceto vetor TF-IDF, foi executada com *scripts Python* e banco de dados *SQLite*. A criação do vetor TF-IDF, etapas Seleção de Características e Treinamento foram executados com a ferramenta *Rapidminer Studio*[11]. Os dados, códigos, processos e resultados obtidos estão disponíveis para *download* em `https://github.com/Luckasx/NQClassificationExperiments`.

[10] https://data.stackexchange.com/stackoverflow/query/new
[11] `http://rapidminer.com/products/studio/`

5 RESULTADOS

Neste capítulos são apresentados os resultados dos experimentos e respectivas análises.

5.1 AVALIAÇÃO DOS EXPERIMENTOS

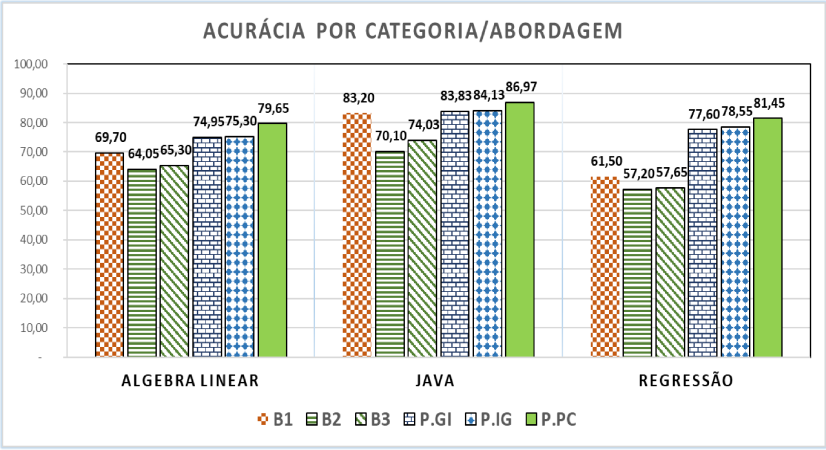
Os melhores resultados obtidos nos experimentos estão apresentados na Tabela 11 e também nos gráficos representados nas Figuras 9, 10, 11, 12, 13, 14, e 15. As linhas na tabela representam a configuração dos *baselines* (B1, B2, B3) e diferentes configurações da proposta apresentada (P.PC, P.IG e P.GI) com os três métodos de peso testados - Correlação de *Pearson* (PC), *Information Gain* (IG) e *GINI Gain* (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.

Tabela 11 – Melhores Resultados nos Experimentos

Experimento / Algoritmo	**Acurácia**	**% 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(50%)	**86,97%**	**83,14**	**91,79**	**92,73**	**81,20**	**87,67**	**86,17**
P.IG/NB(70%)	84,13%	82,49	85,96	86,67	81,60	84,53	83,72
P.GI/NB(70%)	83,83%	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(40%)	**79,65%**	**73,89**	**89,06**	**91,70**	**67,60**	**81,84**	**76,86**
P.IG/SVM(30%)	75,30%	73,73	77,09	78,60	72,00	76,09	74,46
P.GI/NB(60%)	74,95%	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(40%)	**81,45%**	**77,61**	**86,53**	**88,40**	**74,50**	**82,65**	**80,07**
P.IG/NB(60%)	78,55%	76,76	80,60	81,90	75,20	79,25	77,81
P.GI/NB(60%)	77,60%	75,79	79,68	81,10	74,10	78,36	76,79

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 na Tabela 11 que os subconjuntos de características nos experimentos P.PC, P.IG e P.GI 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 (B3). Entretanto, a simples adição do vetor TF-IDF não é suficiente para obter melhora significativa na classificação. O emprego de seleção de características no conjunto total (experimentos P.PC, P.IG e P.GI) 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 questões. 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.

Figura 9 – Melhor Acurácia Por Categoria/Abordagem



Conforme apresentado nas Figuras 10, 11, 12, 13, 14 e 15 a abordagem proposta é melhor do que as demais tanto para as QRE quanto para as NRE quanto à precisão e revocação. Exceto no conjunto de questões Java, onde o *baseline* B1 obteve melhor precisão para as QRE. Os valores de Revocação comportam-se de maneira semelhante, mas os

experimentos P.PC, P.IG e P.GI obtiveram melhor revocação para as QRE nos 3 conjuntos de questões. Por outro lado, a configuração B1 foi melhor na revocação das questões NRE bases Java e Álgebra Linear. Os valores da Medida-F ficaram todos próximos à acurácia resultante. Esses resultados mostram que a abordagem proposta está classificando corretamente a maior parte das QRE e NRE.

O fato do *baseline* B1 ter melhor precisão de QRE e revocação de NRE para as questões Java pode ser resultado do uso de características específicas da comunidade *StackOverflow*. Experimentou-se o *baseline* B1 removendo 3 características: Votos Positivos, Votos Negativos e Pontos de Reputação. Nessa configuração a Precisão QRE cai para 74,38% e a Revocação NRE cai para 74,00%, valores abaixo dos resultados obtidos para a proposta, conforme Tabela 11.

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. Ressalta-se que os classificadores foram usados com os parâmetros padrões definidos na ferramenta *Rapidminer*, sendo assim, com alguns ajustes poderia ser obtido uma melhora na acurácia. *Hyperpipes* atingiu acurácia pouco menor, pois 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 à 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 coeficientes 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 de dois classificadores, Naive Bayes e SVM, de modo a obter melhor acurácia na classificação.

A avaliação estatística foi realizada com o teste t para duas amostras pareadas. Para cálculo da média da proposta deste trabalho usou-se o melhor resultado da acurácia nos experimentos P.PC. Os valores usados para cálculo dos testes estão na Tabela 12. A comparação entre a proposta e os *baselines* resultou nos seguintes valores p:

- (WANQA/B1) = 0,070
- (WANQA/B2) = 0,009

- (WANQA/B3) = 0,018

Figura 10 – Melhor Precisão/Revocação QRE em Álgebra Linear

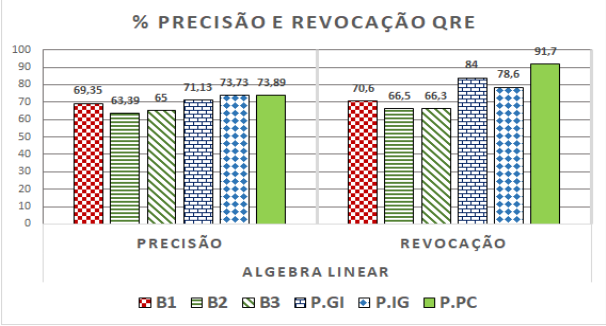


Figura 11 – Melhor Precisão/Revocação QRE em Java

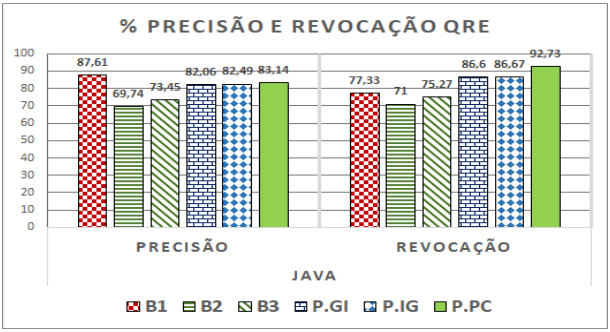


Figura 12 – Melhor Precisão/Revocação QRE em Regressão

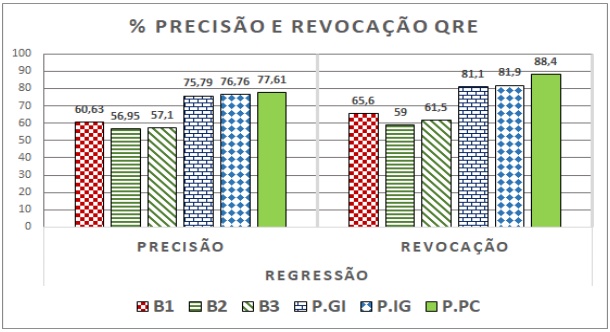


Figura 13 – Melhor Precisão/Revocação NRE em Álgebra Linear

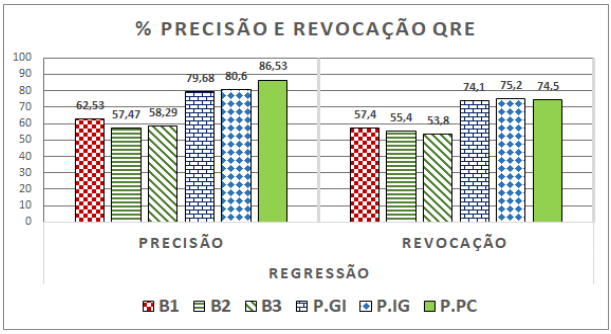


Figura 14 – Melhor Precisão/Revocação NRE em Java

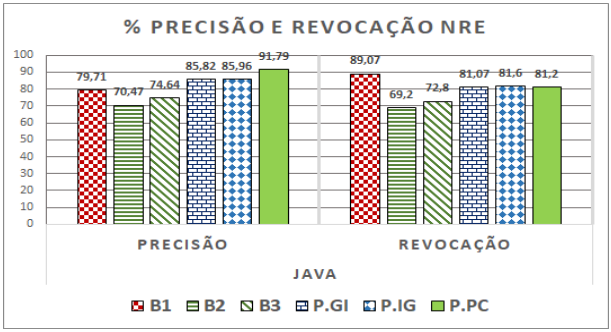
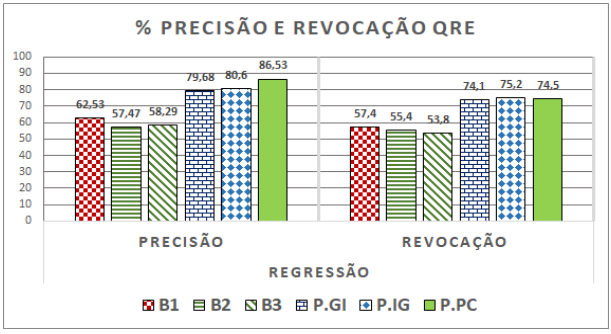


Figura 15 – Melhor Precisão/Revocação NRE em Regressão



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 rejeitar a hipótese teria que se considerar o nível de significância em 10%. Ressalta-se, no entanto, que o *baseline* 1 foi proposto para trabalhar com questões do *StackOverflow*.

Tabela 12 – Teste t para duas amostras pareadas

Conjunto Questões	**Acurácia P.PC**	**Acurácia Baseline**	**Diferença Médias**	**Média Diferenças**	**Desvio Padrão**	**t**	**p**
Baseline B1							
Java	86,97	83,2	3,77				
Álg.Linear	79,65	69,74	9,95	11,223	8,164	2,380	0,070
Regressão	81,45	73,45	19,95				
Baseline B2							
Java	86,97	70,1	16,87				
Álg.Linear	79,65	64,05	15,6	18,906	4,670	7,011	0,009
Regressão	81,45	57,2	24,25				
Baseline B3							
Java	86,97	74,03	12,94				
Álg.Linear	79,65	65,3	14,35	17,030	5,905	4,995	0,018
Regressão	81,45	57,65	23,8				

5.1.1 Comparação das Características Extraídas

Nesta seção, faz-se uma breve discussão sobre a avaliação das características extraídas dos textos e metadados associados às questões. O objetivo é comparar o que foi identificado nos trabalhos relacionados discutidos na comunidade de pesquisa com a classificação das características apresentadas na Tabela 13.

Nos conjuntos testados, o *Tamanho da Descrição* é considerado pelos pesos como tendo grande influência na definição da questão ser QRE ou NRE. A característica *Polidez* das questões também é vista como grande influência para o recebimento de respostas. De acordo com as pesagens, um valor maior para a característica *Subjetividade* não foi um dos principais fatores para o recebimento ou ausência de respostas. No artigo de Yang et al. (2011) é mostrado que as questões com textos muito curtos ou muito longos geralmente recebem respostas. Assim como maior o valor da característica *Subjetividade* da questão, mais recebe-se respostas. O contrário porém foi visto para a *Polidez*, quanto maior menos questões respondidas.

Para Saha, Saha e Perry (2013) apenas 6 características foram suficientes para identificar questões sem respostas. Com isso descartou-

se o uso de *Tamanho da Descrição, Há Código na Descrição, Contagem de URLs e Contagem de Tags*. As características mencionadas foram consideradas como relevantes nos 3 conjuntos experimentados deste trabalho, com exceção da característica *Há Código na Descrição*. Essa característica não foi relevante para o conjunto da categoria *Regressão* pois a maioria das questões não possui código em sua descrição.

Nos resultados de Chua e Banerjee (2015) inferiu-se que quanto menor o texto postado melhor para o recebimento de respostas. No entanto, ausência de *Palavras WH* e maior *Legibilidade* diminuem o envio de respostas. Também identificou-se que o uso de maior *Polidez* contribui para a questão não ser respondida. Ou seja, o fato de ser educado no texto, não trará necessariamente respostas dos demais usuários. Nos experimentos dessa proposta, *Polidez* teve pesos mais altos nos 3 conjuntos. A *Contagem de Palavras WH* teve peso maior na pesagem em 2 dos 3 conjuntos de questões, *Java* e *Álgebra Linear*. A pesagem de *Legibilidade* teve comportamento semelhante, porém com pesos maiores do que *Contagem de Palavras WH no Título*.

Sobre as características textuais, Fong, Zhou e Moutinho (2015) verificaram que a *Contagem de palavras WH* e *Subjetividade* influenciam positivamente para as respostas, quanto maior melhor. Por outro lado, *Tamanho da Questão, Complexidade, Contagem de Tags, Polidez e se Há Código na Descrição* são ruins para trazer respostas, quanto maior pior. Nos resultados dos experimentos, todas essas características foram pesadas como influentes na identificação das classes das questões nos 3 conjuntos de questões.

Segundo os resultados dos experimentos, 3 dos valores calculados para a data da postagem são influentes para a definição das classes das questões, *Cosseno Dia, Cosseno Hora* e *Seno Hora*. Porém, nos 3 conjuntos experimentados, os 3 pesos definiram a característica *Seno Dia* não tão importante para definição das classes QRE e NRE. Uma explicação para o peso atribuído a *Seno Dia* é a menor variação de valores resultantes do que a função *Cosseno*. A *Contagem de Tags* foi considerada como muito relevante nos experimentos. Quanto maior o número de *tags* associadas a uma questão, maior o alcance de usuários na comunidade.

Yang et al. (2011) mencionam que usuários com mais tempo na comunidade tendem a receber mais respostas. Por outro lado, os usuários mais novos possuem mais questões sem respostas devido a falta de familiaridade com a comunidade. Quanto à data da postagem usou-se somente os valores das horas para a classificação. A análise do conjunto de dados indicou que nas horas com mais postagens menos

respostas eram recebidas. Para Dror, Maarek e Szpektor (2013) e Chua e Banerjee (2015) verificou-se que questões postadas à noite recebem mais respostas. Enquanto cerca de 40% das questões postadas durante à tarde permanecem não respondidas. As características advindas dos usuários *Dias Como Membro, Número de Questões Postadas, Respostas Postadas e Proporção de Respostas/Perguntas* não tiveram consenso entre os pesos e categorias dos experimentos realizados no presente trabalho.

Nos experimentos de Chua e Banerjee (2015), identificou-se que o recebimento de respostas depende mais do número de postagens do que o tempo na comunidade. Sobre a contagem de *tags* os resultados dos seus experimentos identificaram que quanto menor o número de *tags* associadas mais chance de obter respostas. A mesma conclusão sobre a quantidade de *tags* foi vista no trabalho de Saha, Saha e Perry (2013). Para a proposta de Fong, Zhou e Moutinho (2015) a quantidade de postagens dos usuários foi irrelevante para a classificação, assim como *Erros de Linguagem* e o valor *Cosseno Dia*.

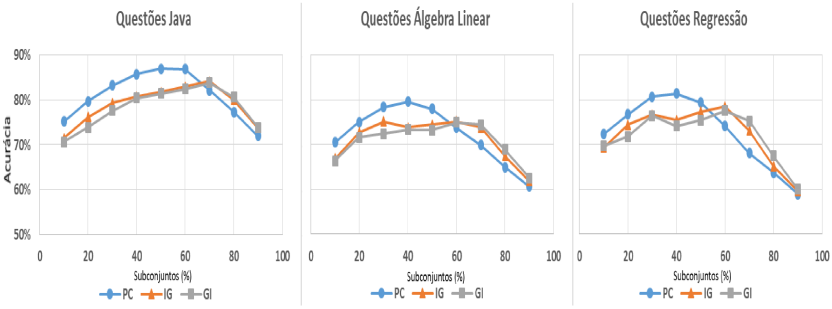
Como pode-se ver, há muitas diferenças entre os trabalhos. Esse fato reforça que cada categoria deve ser treinada individualmente. Pois cada grupo de usuários tem sua própria maneira de realizar as postagens, tanto no texto quanto ao contexto. Mesmo dentro de uma comunidade.

5.1.2 Análise da Seleção de Características

A Figura 16 apresenta os valores de acurácia obtidos na execução de P.PC, P.IG e P.GI para o classificador Naive Bayes em cada subconjunto de caracteríriscas conforme os pesos aplicados. Nos experimentos foram testados 9 tamanhos de subconjuntos de características: 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80% e 90% das características com maiores pesos. 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.

Outro ponto importante consiste em analisar como as características extraídas, exceto os termos do vetor TF-IDF, foram pesadas

Figura 16 – Acurácia X Filtro Naive Bayes



em cada grupo de questões e métodos de cálculo de peso. A Tabela 13 apresenta a posição da característica dentro do conjunto total. A primeira observação que pode ser feita é que a maioria das 20 características extraídas, exceto vetor TF-IDF, tem pesos cujo valor está na metade superior do conjunto total. 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 nos 3 conjuntos de questões, ou seja, teve peso com valor baixo demais em relação às demais características. 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 Gain*.

Uma segunda observação importante a ser feita é sobre as características em cada categoria de questões. Por exemplo, na Tabela 13 a característica *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. Isso também é válido para os termos do vetor TF-IDF. Os vetores TF-IDF são diferentes em cada categoria, mas podem conter termos em comum. Alguns exemplos são relacionados na Tabela 14. O termo *request* está bem posicionado para Java e Álgebra Linear, enquanto para Regressão está classificado na metade inferior do conjunto. Assim como o termo *book* é relevante para as questões de

Tabela 13 – 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

Regressão mas não tanto para os outros dois conjuntos. Essa observação reforça, no entanto, a necessidade de escolher o subconjunto com as características que melhor traduzem o comportamento dos usuários em relação à postagem de respostas. Esse subconjunto é usado então para a criação de um modelo classificador com melhor acurácia.

Tabela 14 – Ranking de Termos Extraídos

Termo	**# Java(7670)**			**# Álg. Linear(2865)**			**# Regressão(3530)**		
	PC	IG	GI	PC	IG	GI	PC	IG	GI
book	6183	1786	1376	1247	589	440	8	15	21
request	25	29	29	461	821	936	1802	3233	3303
analog	2552	3107	3228	463	913	664	3506	1103	1115
librari	637	570	469	2830	2418	2117	545	1013	673
meaning	7631	6792	5995	184	291	392	2609	1686	1790

5.2 LIMITAÇÕES

Nesta seção são descritos alguns cenários onde a proposta pode não funcionar adequadamente.

A primeira situação seria a ausência de questões. Caso uma comunidade queira usar o modelo classificador desde sua inauguração seria necessário um treinamento com questões de outra comunidade. Da mesma forma, quando houver somente *QRE* ou *NRE*. Os classificadores precisam de exemplos de ambos as classes para poder diferenciá-las. Na ausência de uma delas, a análise das características ignora a classe ausente.

A abordagem foi criada para o treinamento de questões de apenas uma categoria de cada vez. Sendo assim, a mistura de questões de diferentes categorias no processamento pode diminuir a acurácia. Apesar de haver termos comuns às categorias, a acurácia melhora com a análise da presença de termos específicos em cada assunto. O uso de um conjunto de questões contendo múltiplas categorias acarreta em um modelo não tão preciso como o que é treinado individualmente em para uma categoria.

Outra situação que prejudicaria o funcionamento seria a mistura de idiomas nas questões. Caso não seja usado o dicionário adequado ao idioma, a criação do vetor TF-IDF será executada incorretamente. Dessa forma, o modelo classificador seria treinado com características diferentes das adequadas ao conjunto das questões.

6 CONCLUSÃO

Nesta dissertação foi apresentada uma proposta para a classificação de novas questões postadas em CQA como respondíveis ou não respondíveis no momento da postagem. O objetivo principal foi apresentar uma abordagem para a criação de modelos classificadores que possa ser aplicada na grande maioria das comunidades de perguntas e respostas. Para alcançar esse objetivo foi necessário analisar e definir quais características estão presentes nessas comunidades.

Diferente de trabalhos anteriores, buscou-se apresentar uma abordagem aplicável à maioria das CQA, pois as propostas anteriores focaram em comunidades específicas. A abordagem proposta filtra as melhores características para cada conjunto de uma categoria de questões. Os resultados obtidos nos 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.

A abordagem foi testada com 3 conjuntos de questões, 4 algoritmos classificadores, 3 tipos de pesos e comparada à 3 *baselines* representando os trabalhos relacionados. Os experimentos mostraram que a acurácia é melhor quando comparada aos *baselines*. Com os resultados obtidos nos experimentos, é possível definir Naive Bayes e SVM como bons classificadores a serem usados na abordagem. Para cálculo dos pesos, na definição da relevância das características, ficou evidente nos resultados apresentados que a Correlação de Pearson foi melhor que os demais experimentados.

O trabalho descrito nessa dissertação gerou um artigo publicado e apresentado na XXXIII edição do SBBD - Simpósio Brasileiro de Banco de Dados (KNOCHENHAUER; DORNELES; WIVES, 2018) (Qualis B2) em 2018. Pretende-se ainda, submeter uma versão estendida, em inglês, do artigo para um periódico no estrato Qualis A1-B1.

Como trabalhos futuros, pretende-se elaborar um modo automático para definir o tamanho de subconjunto que traz melhor acurácia. Apesar dos experimentos mostrarem que o melhor tamanho de subconjunto esteja entre 40% e 70% não se pode afirmar que um desses valores será sempre o melhor para todos os conjuntos. Sendo assim, é necessário executar a abordagem com diferentes tamanhos de filtro para que se encontre a que obtém melhor acurácia.

O uso de validação cruzada neste trabalho obteve os valores das características sobre o conjunto completo de cada categoria. Por isso, pretende-se a execução dos experimentos separando as melhores carac-

terísticas apenas dos dados de treinamento sem conhecer os dados de teste. Com isso, pode-se obter uma acurácia mais compatível com o uso prático. Quanto aos testes estatísticos, pode-se melhorar comparando a acurácia de cada *fold* nos experimentos, em vez de comparar a média final.

Nos experimentos foram usadas questões mais voltadas às àreas de exatas, *Java, Álgebra Linear* e *Regressão*. Por esse motivo, é necessário testar também conjuntos de questões de categorias mais diversas às dos experimentos, como, por exemplo, política ou saúde. Para fins de comparação faltou executar as abordagens misturando as questões de diferentes categorias, assim como experimentar os conjuntos de forma desbalanceada. Além disso, verificou-se que há testes disponíveis para classificar a diversidade dos domínios sendo assim futuramente pode-se comprovar ou não se as categorias testadas são de forma geral diferentes.

Apesar desse projeto ter como objetivo classificar os conteúdos postados em comunidades de perguntas e respostas, há a possibilidade de classificação de outros conteúdos textuais que contenham título, descrição e categoria, como por exemplo notícias e postagens em redes sociais. Considerando que as características das questões são as mesmas contidas em qualquer outro tipo de texto, a proposição desta abordagem pode ser adaptada para outros objetivos em classificação de texto. Porém os dados obtidos dos usuários devem ser adaptados conforme o contexto.

REFERÊNCIAS

AGGARWAL, C. C. *Data Mining: The Textbook*. [S.l.]: Springer Publishing Company, Incorporated, 2015. ISBN 3319141414, 9783319141411.

ASADUZZAMAN, M. et al. Answering questions about unanswered questions of stack overflow. In: *Proceedings of the 10th Working Conference on Mining Software Repositories*. Piscataway, NJ, USA: IEEE Press, 2013. (MSR '13), p. 97–100. ISBN 978-1-4673-2936-1. <http://dl.acm.org/citation.cfm?id=2487085.2487109>.

BAEZA-YATES, R.; RIBEIRO-NETO, B. *Modern Information Retrieval*. 2nd. ed. USA: Addison-Wesley Publishing Company, 2008. ISBN 9780321416919.

BALTADZHIEVA, A.; CHRUPAŁA, G. Question quality in community question answering forums: a survey. *Acm Sigkdd Explorations Newsletter*, ACM, v. 17, n. 1, p. 8–13, 2015.

CHUA, A. Y.; BANERJEE, S. Answers or no answers: Studying question answerability in stackoverflow. *Journal of Information Science*, Sage Publications, Inc., Thousand Oaks, CA, USA, v. 41, n. 5, p. 720–731, out. 2015. ISSN 0165-5515. <http://dx.doi.org/10.1177/0165551515590096>.

DROR, G.; MAAREK, Y.; SZPEKTOR, I. 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*. Berlin, Heidelberg: Springer-Verlag, 2013. (ECMLPKDD'13), p. 499–514. ISBN 978-3-642-40993-6. <https://doi.org/10.1007/978-3-642-40994-3_32>.

FONG, S. et al. Feature selection in life science classification: Metaheuristic swarm search. *IT Professional*, v. 16, n. 4, p. 24–29, July-Aug. 2014. ISSN 1520-9202. <doi.ieeecomputersociety.org/10.1109/MITP.2014.50>.

FONG, S.; YANG, X.; DEB, S. Swarm search for feature selection in classification. In: *2013 IEEE 16th International Conference on Computational Science and Engineering*. [S.l.: s.n.], 2013. p. 902–909.

FONG, S.; ZHOU, S.; MOUTINHO, L. Text analytics for predicting question acceptance rates. *IT Professional*, v. 17, n. 4, p. 34–41, July 2015. ISSN 1520-9202.

HALL, M. A. *Correlation-based Feature Selection for Machine Learning*. Tese (Doutorado) — The University of Waikato, 1999.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011.

KNOCHENHAUER, L. V.; DORNELES, C. F.; WIVES, L. K. WANQA: uma abordagem para identificar novas questões não respondíveis em comunidades de perguntas e respostas. In: *XXXIII Simpósio Brasileiro de Banco de Dados, SBBD 2018, Rio de Janeiro, RJ, Brazil, August 25-26, 2018.* [s.n.], 2018. p. 1–12. <http://sbbd.org.br/2018/wp-content/uploads/sites/5/2018/08/001-sbbd_2018-fp.pdf>.

LI, B. et al. Exploring question subjectivity prediction in community qa. In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2008. (SIGIR '08), p. 735–736. ISBN 978-1-60558-164-4. <http://doi.acm.org/10.1145/1390334.1390477>.

LIU, J. et al. Question difficulty estimation in community question answering services. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. [S.l.: s.n.], 2013. p. 85–90.

MCCALLUM, D. R.; PETERSON, J. L. Computer-based readability indexes. In: *Proceedings of the ACM '82 Conference*. New York, NY, USA: ACM, 1982. (ACM '82), p. 44–48. ISBN 0-89791-085-0. <http://doi.acm.org/10.1145/800174.809754>.

MOTODA, H.; LIU, H. Feature selection, extraction and construction. *Communication of IICM (Institute of Information and Computing Machinery, Taiwan) Vol*, v. 5, p. 67–72, 2002.

SAHA, R. K.; SAHA, A. K.; PERRY, D. E. 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*. New York, NY, USA: ACM, 2013. (ESEC/FSE 2013), p. 663–666. ISBN 978-1-4503-2237-9. <http://doi.acm.org/10.1145/2491411.2494585>.

SRBA, I.; BIELIKOVA, M. A comprehensive survey and classification of approaches for community question answering. *ACM Trans. Web*, ACM, New York, NY, USA, v. 10, n. 3, p. 18:1–18:63, ago. 2016. ISSN 1559-1131. <http://doi.acm.org/10.1145/2934687>.

YANG, L. et al. Analyzing and predicting not-answered questions in community-based question answering services. In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, 2011. (AAAI'11), p. 1273–1278. <http://dl.acm.org/citation.cfm?id=2900423.2900625>.

YANG, X.-S.; DEB, S.; FONG, S. Accelerated particle swarm optimization and support vector machine for business optimization and applications. In: FONG, S. (Ed.). *Networked Digital Technologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 53–66. ISBN 978-3-642-22185-9.

ZHOU, S.; FONG, S. 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*. [S.l.]: Springer, 2016. p. 49–93.