UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

FRANCISCO DOUGLAS RODRIGUES DE OLIVEIRA

ATRIBUIÇÕES DE NOVAS FUNCIONALIDADES DE PRÉ-PROCESSAMENTO E MINERAÇÃO DE DADOS PARA FERRAMENTA UUX-POSTS

RUSSAS

2018

FRANCISCO DOUGLAS RODRIGUES DE OLIVEIRA

ATRIBUIÇÕES DE NOVAS FUNCIONALIDADES DE PRÉ-PROCESSAMENTO E MINERAÇÃO DE DADOS PARA FERRAMENTA UUX-POSTS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do CAMPUS DE RUSSAS da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientadora: Profa. Dra. Marília Soares Mendes

RUSSAS

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

O47a Oliveira, Francisco Douglas Rodrigues de.
Atribuições de novas funcionalidades de Pré-processamento e Mineração de Dados para ferramenta UUX-Posts / Francisco Douglas Rodrigues de Oliveira. – 2018.
63 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2018.
Orientação: Profa. Dra. Marília Soares Mendes.

1. Ferramenta. 2. Mineração de Dados. 3. Pré-processamento. 4. MALTU. I. Título.

CDD 005.1

FRANCISCO DOUGLAS RODRIGUES DE OLIVEIRA

ATRIBUIÇÕES DE NOVAS FUNCIONALIDADES DE PRÉ-PROCESSAMENTO E MINERAÇÃO DE DADOS PARA FERRAMENTA UUX-POSTS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do CAMPUS DE RUSSAS da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em:

BANCA EXAMINADORA

Profa. Dra. Marília Soares Mendes (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Alexandre Matos Arruda
Universidade Federal do Ceará (UFC)

Prof. Ms. Tatiane Fernandes Figueiredo
Universidade Federal do Ceará (UFC)

"Obstáculos não devem te impedir. Se você encontrar uma parede, não desista. Descubra como escalá-la."
(Michael Jordan)

AGRADECIMENTOS

Agradeço muito a Deus, por tudo que tem me proporcionado durante toda essa jornada, pela determinação para enfrentar todas as dificuldades, que de alguma forma servem de aprendizado. Enfim, por ter me iluminado durante essa caminhada.

Sou muito grato a minha família e amigos que me ajudaram bastante de todas as formas. Em especial ao meu primo Dr. Edijailson, por ter me incentivado a vir estudar na UFC em Russas, por ter me acolhido e me ajudado durante todo esse período. Aos meus pais (Beta e Dinarte), por sempre me incentivar a estudar, sempre acreditar em mim, e sobretudo a nunca medir esforços para me ajudar em todos os problemas que são postos a mim. Ao meu amigo Tatá, meu primo Allyf, e toda minha família que me ajudaram bastante também.

Não posso esquecer do corpo docente da universidade que contribuiu para minha formação profissional e pessoal. Em especial a minha orientadora Dra. Marília Mendes, que acreditou em mim, acreditou na minha capacidade e em minha vontade de aprender, esta pessoa foi sem dúvidas essencial na minha formação.

Aos meus amigos da faculdade, em especial os companheiros Carlos Eduardo, Anderson Soriano, Wallison e Gilberto. Aos colegas do projeto MALTU, do café com software, da galera do time de software, enfim a todas as pessoas que me ajudaram seja na faculdade, ou na vida a fora, me descontraindo e me fazendo feliz.

RESUMO

Os tradicionais métodos de avaliação de sistemas apresentam limitações, principalmente em relação à espontaneidade do usuário no momento da avaliação. Com isso surgem estudos e metodologias para avaliação de sistema a fim de superar essas limitações, como a metodologia de avaliação textual chamada MALTU, com o propósito de avaliar sistemas a partir de Postagens Relacionadas ao Uso (PRU), principalmente em sistemas sociais. Avaliações a partir de dados, como postagens em sistemas sociais, podem superar as limitações dos métodos tradicionais, mas avaliações a partir desses dados não são triviais, necessitando de técnicas automatizadas. Com isso a MALTU dispõe de uma ferramenta chamada UUX-Posts para dá suporte aos profissionais avaliadores de sistemas. Esta ferramenta em sua versão inicial fornece suporte as etapas de extração e classificação de PRUs, mas possui algumas limitações, como a classificação por ainda utilizar o modelo de busca booleano para classificar postagens em PRU, e por não fornecer técnicas de pré-processamento, importante para o desempenho do classificador. Em sua nova versão, a ferramenta visa dar suporte a todas as etapas da metodologia MALTU, incluindo a classificação de PRUs a partir da contribuição deste trabalho. Dessa forma, este trabalho propõe a implementação de funcionalidades de pré-processamento e mineração de dados para a ferramenta UUX-Posts, com uso de algoritmos de classificação mais sofisticados comparado a busca booleano. Para isto foram realizadas pesquisas na literatura, com o objetivo de verificar soluções confiáveis e *open source* que poderiam ser utilizadas na classificação de PRUs. A partir desse estudo foram definidas as técnicas e implementadas as funcionalidades propostas para, em outro momento ser, integradas na ferramenta. Com a implementação das funcionalidades e a partir dos resultados foi constatado que o pré-processamento contribui para o melhor resultado da classificação, apesar da sua contribuição não ter sido tão representativa, mas que pode ser melhor aproveitada com o uso de classificadores mais robustos.

Palavras-chave: Ferramenta. Mineração de Dados. Pré-processamento. MALTU.

ABSTRACT

The traditional system evaluation methods present limitations, specialy when it comes to the user's spontaneity on the evaluation moment. With that, arise studies and methodologies to system evaluation with the point of overcome these limitations, the textual evaluation methodology called MALTU, that carries the purpose of evaluating systems as stated by the Usage Related Posts (PRU), mainly in social systems. According to data such as posts in social systems, they can overcome the traditional methods limitations, but according to these data are not trivial, requiring automated techniques. With that, MALTU offers a tool called UUX-Posts to support professional systems evaluators. This tool in its initial version, provides support to the extraction and classification of PRUs steps, although presents some limitations like the classification, for still use the boolean search method to classify PRU posts, and does not provide preprocessing techniques, important things to the classifier performance. In its new version, the tool aims to provide support for all steps of the MALTU methodolody, including the PRUs classification from the contribution of this paper. Thus, this paper proposes the implementation of preprocessing functionality and data mining for the UUX-Posts tool, together with the use of more sophisticated classification algorithms compared to the boolean search. For this reason, researches were done throughout the literature, with the goal of verifying reliable and open source solutions that could be used to classify PRUs. As stated by this study, were defined techniques and proposed functionalities to, in other moment, be integrated in the tool. With the implementation and results, were found that the preprocessing contributes to a better classification result, although its contribution was not so representative, could be better improved with the use of more robust classifiers.

Keywords: Tool. Data Mining. Preprocessing. MALTU.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Segundo Mendes (2015), e Becker e Tumitam (2013), os principais métodos de avaliação do design de interação, como pesquisa de campo, entrevistas e questionários, apresentam algumas desvantagens relacionadas ao custo, restrição a um grupo definido, e muitas vezes são pouco eficazes. A primeira autora ainda relata que essas técnicas citadas não consideram a espontaneidade do usuário, o que seria importante na avaliação do sistema. Assim, as avaliações a partir de postagens de usuários em Sistemas Sociais (SS), como Twitter e Facebook, poderiam apresentar ganhos significativos, por se tratar de ambientes interativos, onde seus usuários compartilham informações, que muitas vezes se refere ao próprio sistema em uso, assim suas informações são fornecidas de forma espontânea. O que faz com que esse tipo de avaliação seja mais relevante, comparado com os métodos tradicionais, como apresenta Mendes (2015), em seu estudo sobre avaliação de SS a partir da linguagem textual de usuários.

Essas avaliações a partir das opiniões de usuários não são triviais, requerem técnicas automatizadas, devido ao grande volume de dados e a sua bipolaridade. Neste cenário, Mendes (2015) propôs uma nova metodologia de avaliação, chamado MALTU - Modelo de Avaliação da interação em SS a partir da Linguagem Textual do Usuário, para avaliar a Usabilidade e experiência do usuário (em inglês, *User eXperience* - UX) (UUX) de sistemas. A ideia da metodologia é analisar postagens de usuários em SS, a fim de fornecer um resultado de avaliação do sistema a partir dessas postagens (MENDES, 2015).

Essas postagens são denominadas pela MALTU como Postagens Relacionadas ao Uso (PRU), que são postagens no qual o usuário se refere ao próprio sistema em uso, mostrando sua satisfação ou insatisfação. Por exemplo, *"eu gosto de conversar pelo twitter porque as conversas n duram muito e tem limite de caracteres"* é uma PRU, enquanto que, *"bom dia, hj é sexta feiraaa..."* é uma postagem não-PRU. Assim, as PRUs são importantes na avaliação de sistema, como mencionado, avaliando a satisfação ou insatisfação dos usuários com o sistema. Possibilitando ainda, identificar problemas em funcionalidades, tendo em vista que, muitas vezes, os usuários relatam em suas postagens seus problemas enfrentados no sistema durante seu uso.

A metodologia MALTU dispõe de uma ferramenta para realização da extração e classificação de postagens em SS, chamada de UUX-Posts[1] (MENDES; FURTADO, 2017). Esta ferramenta é usada para auxiliar os profissionais da área de Interação Humano-Computador (IHC), com o objetivo de apoiar a avaliação textual de SS (MENDES, 2015). A UUX-Posts

[1] http://uuxposts.russas.ufc.br

apoia às etapas de extração/classificação de postagens a partir do Twitter, ou a partir de uma planilha csv, de qualquer outro sistema. No entanto esta ferramenta apresenta algumas limitações (MENDES; FURTADO, 2017), por ser usado o modelo de busca booleano, por não disponibilizar técnicas de pré-processamento de texto, e pela ausência de técnicas precisas de Mineração de Dados (MD).

Segundo Elmasri e Navathe (2011) no modelo de busca booleano os documentos são representados como um conjunto de termos, onde as consultas são formuladas como uma combinação de termos usando os operadores da lógica booleana, como AND, OR e NOT. Neste modelo não existe a noção de pontuação dos documentos resultantes, todos os resultantes são considerados igualmente importante, ou seja, não é considerada a frequência dos termos no documento. Assim, Elmasri e Navathe (2011) relatam que este modelo não possui algoritmos de pontuação sofisticados e estão entre os modelos mais antigos e mais simples.

O pré-processamento dos dados é uma etapa contida nos estudos de Processamento de Linguagem Natural (PLN). Segundo Coppin (2013), o PNL é uma coleção de técnicas usadas para permitir que computadores "entendam" a linguagem humana, com objetivo de extrair informação gramatical e significados a partir de expressões vocais humanas, e a partir desse entendimento seja possível realizar tarefas úteis como resultado. O pré-processamento, por sua vez, trata de um conjunto de diferentes estratégias e técnicas que são inter-relacionadas de formas complexas, que devem ser aplicadas para tornar os dados mais apropriados para a MD. Assim, esta etapa é de fundamental importância, principalmente com dados de SS, que consistem em textos apresentando vários ruídos que podem comprometer o resultado final, como links, palavras irrelevantes, erros ortográficos, etc.

Como define Tan, Steinbach e Kumar (2009), a MD é o processo de descoberta automática de informações úteis em grandes depósitos de dados. Tal processo é uma parte integral da descoberta de conhecimento em banco de dados, conhecido como Descoberta do Conhecimento em bases de dados (em inglês, *Knowledge Discovery in Databases*, KDD). Fayyad, Piatetsky-Shapiro e Smyth (1996) detalham o KDD em 6 etapas, sendo: 1) dados; 2) seleção; 3) pré-processamento; 4) transformação; 5) mineração de dados; e 6) interpretação/avaliação. A etapa de mineração de dados, envolve principalmente a aplicação dos algoritmos. Esta etapa assim como o pré-processamento, serão detalhadas na próxima seção.

Dessa forma, este trabalho propõe contribuir com a evolução da ferramenta UUX-Posts, fornecendo a implementação novas técnicas para novas funcionalidades a fim de superar

suas limitações quanto a MD. A proposta baseia em disponibilizar técnicas de pré-processamento dos textos e técnicas de classificações das postagens, por meio da MD, seguindo o processo KDD e a metodologia MALTU.

1.1 Motivação

O autor deste trabalho foi bolsista durante 9 meses no projeto Avaliação da interação em sistemas sociais a partir da linguagem textual do usuário, coordenado pela Profa. Dra. Marília Soares Mendes, e financiado pela Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP). Atualmente o bolsista é voluntário, em suas atividades realizadas neste projeto estudou técnicas de MD e Pré-processamento de dados, além do próprio modelo de avaliação do projeto, no qual conta com a ferramenta citada para a realização da avaliação.

A partir dos estudos realizados, foram constatadas as limitações da ferramenta UUX-Posts comparada as demais técnicas estudadas, assim como a viabilidade de atribuir essas técnicas de MD e Pré-processamento na ferramenta. O que fomentou a realização desta pesquisa e sua contribuição para o projeto, a fim de melhorar o desempenho da ferramenta com o uso de técnicas mais sofisticadas.

1.2 Objetivos

1.2.1 Objetivo geral

Implementar funcionalidades de pré-processamento e mineração de dados para uma ferramenta de avaliação textual de sistemas.

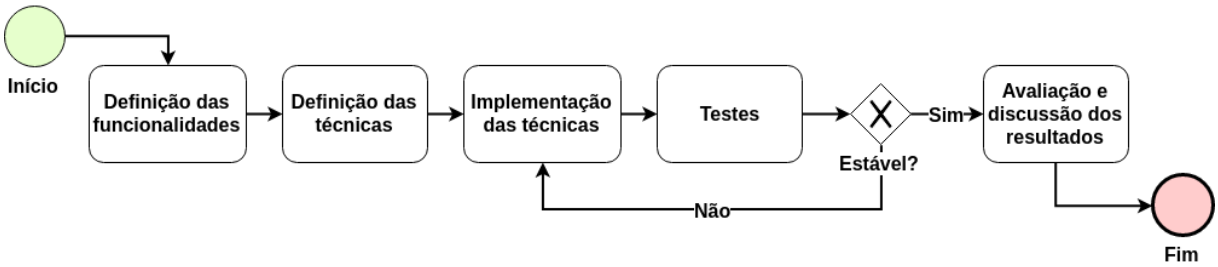
1.2.2 Objetivos específicos

- Caracterizar ferramentas similares em relação aos métodos utilizados.
- Avaliar a precisão das funcionalidades implementadas.
- Comparar a classificação sem pré-processamento com a classificação com pré-processamento.
- Comparar a classificação com diferentes algoritmos.

1.3 Procedimentos metodológicos

Os procedimentos metodológicos adotados e seguidos para atingir os objetivos da pesquisa, são representado na Figura 1, ilustrando suas etapas. A partir da etapa de implementação o processo foi em alguns momentos iterativo, onde o fluxo voltou para etapas anteriores.

Figura 1 – Etapas do procedimento metodológico



Fonte: Elaborado pelo autor

A primeira etapa consistiu na definição das funcionalidades para a ferramenta UUX-Posts. Esta etapa envolveu analisar a metodologia MALTU e a sua ferramenta, para então definir as funcionalidades que seriam importantes, sendo funcionalidades restritas ao pré-processamento e mineração de dados. Para o pré-processamento, o objetivo da escolha foi deixar esse processo o mais exaustivo possível, com intuito de melhorar a precisão da avaliação. Enquanto que para a mineração de dados, o objetivo é a classificação de PRUs (apesar da metodologia MALTU sugerir a classificação em outras categorias, este trabalho tem o foco somente na classificação de PRUs), dessa forma a funcionalidade já estava pré-definida no qual seria utilizado a tarefa de previsão, com método de classificação.

Com as funcionalidades já definidas, o próximo passo foi a escolha das técnicas a serem implementadas. Esta etapa envolveu a definição dos algoritmos a serem utilizados, assim como as tecnologias. Com as técnicas selecionadas, o passo seguinte consistiu exclusivamente na implementação das técnicas e consequentemente das funcionalidades. No primeiro momento foi realizada a implementação de todas as técnicas de pré-processamento, e em seguida as técnicas para classificação de PRUs. Durante esta etapa é necessário a iteração constante a com a etapa de testes.

O passo seguinte consistiu na realização de testes utilizando as funcionalidades propostas. Como o foco deste trabalho é somente a implementação das técnicas e não a integração com a ferramenta, os testes realizados envolvem somente testes unitários no ambiente de desenvolvimento, pois o foco desses testes é verificar individualmente o funcionamento das técnicas.

Os teste unitários "são testes de exclusividade do código e da interface disponibilizada de uma unidade. São testadas as classes, os métodos ou até pequenos trechos de código."(AMARAL *et al.*, 2010, p. 4). Quando constatado alguma inconsistência na implementação, correções foram realizadas, até se manter estável ou o mais próximo possível.

Por fim, foram realizadas avaliações dos resultados e discussão deles. Para avaliar os resultados foram realizados experimentos com as funcionalidades implementadas. Assim, a partir dos resultados as técnicas são comparadas e discutidas.

1.4 Organização do trabalho

Este trabalho está organizado em 8 capítulos, sendo eles: 1) Introdução, onde este trabalho foi contextualizado, apresentando a motivação da realização, objetivos e os procedimentos metodológicos adotados para atingir os objetivos traçados; 2) Fundamentação teórica, apresentando os conceitos e assuntos pertinentes a esta pesquisa; 3) Trabalhos relacionadas, são apresentados com intuito de mostrar e comparar pesquisas similares existentes; 4) Estudo de ferramentas similares, apresentando algumas ferramentas estudadas e suas características; 5) Implementação das funcionalidades, descrevendo os detalhes da implementação; 6) Condução do experimento, descrevendo o planejamento do experimento e como será conduzido; 7) Discussão dos resultados, apresentando os resultados obtidos para cada método e artefatos, e em seguida o estudo comparativo; e 8) Conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TÉORICA

Como embasamento teórico, foi necessária uma revisão dos principais assuntos pertinentes a pesquisa. Nesta seção apresentam-se os seguintes conteúdos: 1) avaliação textual de sistemas, detalhando as etapas de extração e classificação das postagens dos usuários e a ferramenta UUX-Posts; e 2) descoberta do conhecimento de banco de dados, detalhando as etapas de pré-processamento e mineração de dados. A explanação de conteúdos tiveram o objetivo de auxiliar no melhor entendimento da pesquisa a ser realizada.

2.1 Avaliação textual de sistemas

A avaliação textual de sistemas consiste em utilizar narrativas geradas pelo usuário a fim de obter algum resultado de avaliação (MENDES, 2015). No caso específico do contexto deste trabalho, é utilizada a metodologia MALTU, que propõe a coleta das postagens dos usuários no próprio sistema em uso a fim de considerar a espontaneidade do usuário (MENDES, 2015).

A metodologia MALTU propõe ao avaliador cinco etapas no processo de avaliação. A primeira etapa consiste na definição do contexto da avaliação, no qual envolve o contexto de uso e domínio do sistema, e os objetivos da avaliação. Definido o escopo da avaliação, é realizada a extração de PRUs, que consiste na segunda etapa. A extração de PRUs pode ser realizada de forma manual ou automática, na extração automática é usada a ferramenta UUX-Posts, apresentada mais detalhadamente na seção 2.2. Na terceira etapa é realizada a classificação de PRUs, a MALTU propõe diferentes classificações, apresentado mais detalhadamente na seção seguinte. A quarta etapa consiste na interpretação dos resultados da classificação, e por fim, a quinta e última etapa consiste no relato dos resultados.

O foco deste trabalho está nas etapas de extração e classificação, segunda e terceira respectivamente.

2.1.1 Extração e classificação de postagens de usuários

A ferramenta UUX-Posts faz a extração e classificação das postagens de usuários, primeiramente em PRUs. Isso é feito da seguinte forma: usando a API do Twitter e aplicando a busca booleana com padrões pré-definidos. Em uma avaliação do Twitter, o resultado da extração pode ser conforme apresentado na Tabela 1. Constituído de postagens PRUs e não-PRUs. Isso acontece porque a busca booleana não fornece um bom resultado.

Tabela 1 – Exemplos de PRUs e não-PRUs, obtidas no Twitter.

Postagem	PRU	não-PRU
"bicho eu to farto desse desgraçado do twitter ficar repetindo as fotos na galeria que saco"	X	
"desde ontem o twitter aqui nao abre as imagens, nem o icon do perfil de vcs"	X	
"E AI GALERA DO TWITTER, AQUI QUEM FALA É O NICK"		X
"encontrei um boy do twitter aqui na ufsc"		X

Fonte: Elaborada pelo autor

A etapa de classificação também envolve classificar as PRUs em demais categorias sugeridas pela metodologia MALTU, sendo elas (MENDES, 2015):

a) classificação por tipo (crítica, elogio, ajuda, dúvida, comparação e sugestão);
b) classificação por intenção (visceral, comportamental e reflexiva);
c) análise de sentimento (positivo, neutro ou negativo);
d) classificação por funcionalidades;
e) classificação por critério de qualidade de uso (de Usabilidade, experiência do usuário e suas metas);
f) classificação por artefato (dispositivos utilizados pelos usuários).

Mendes (2015) apresenta padrões para esses tipos de classificações, descrição destas categorias e exemplos de postagens classificadas. Como o foco deste trabalho é somente as classificações em PRUs, o detalhamento dos demais tipos de classificações não são necessários para fomentar o embasamento teórico necessário.

2.2 Ferramenta UUX-Posts

A ferramenta UUX-Posts (MENDES, 2015; MENDES; FURTADO, 2017) foi proposta para auxiliar profissionais de IHC na avaliação textual de sistemas, seguindo a metodologia MALTU. Em sua versão inicial (descrita na introdução), a ferramenta dava suporte apenas as etapas de extração e classificação, da metodologia MALTU, apresentadas anteriormente. Em sua nova versão (versão 2)[1], a ferramenta visa dar suporte a todas as etapas da metodologia MALTU, para a classificação manual, e para a classificação automática de PRUs a partir da contribuição deste trabalho, embora a contribuição seja somente a implementação, ficando fora do escopo a integração com a ferramenta.

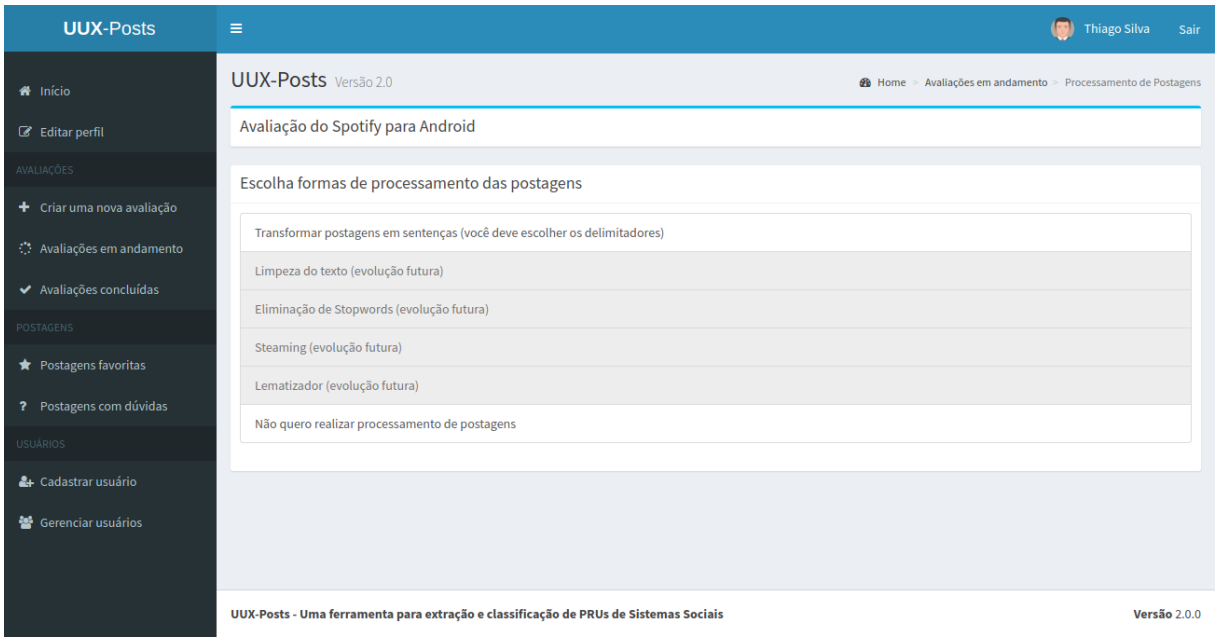
Na versão 2, é apresentada a ferramenta aos seus usuários, no qual é definida contextualizando sua proposta e seu objetivo. A definição abaixo pode ser encontrada na própria ferramenta em sua página inicial.

[1] http://uuxposts.russas.ufc.br

> A UUX-Posts é uma ferramenta de extração e classificação de postagens para avaliar a Usabilidade e Experiência do Usuário (UX) de sistemas. As postagens são coletadas de perfis públicos de Sistemas Sociais, como o Twitter ou Lojas de aplicativos. A ferramenta ainda permite o envio de um banco de postagens em formato CSV. A busca padrão é realizada utilizando um conjunto de padrões relacionados a Usabilidade e UX, no entanto, a ferramenta também possibilita uma busca avançada na qual o próprio usuário poderá alterar e definir seus padrões de busca. UUX-POSTS. *Disponível em: <http://uuxposts.russas.ufc.br>. Acesso em: 05 maio 2016.*

Com todas as etapas da MALTU na própria ferramenta, o usuário primeiro define o contexto da avaliação, inserindo algumas informações, como o sistema avaliado e a fonte de postagens. Em seguida, o usuário pode optar por extrair postagens dos SS (Twitter e Facebook) diretamente pela ferramenta, ou inserir um arquivo csv contendo as postagens. Após esta etapa serão inseridas as tarefas de pré-processamento, atribuídas a partir da implementação deste trabalho (Figura 2). Em seguida, na etapa três, o usuário pode escolher em classificar as postagens de forma automática ou a classificação por avaliadores (manual), como representado na Figura 3. Para as técnicas de classificação automática, assim como as tarefas de pré-processamento, serão utilizadas a partir da implementação deste trabalho. Vale salientar que a classificação automática está disponível nesta versão somente usando busca booleana e para classificar em: 1) PRU e não-PRU; 2) Tipos de postagens; e 3) Em facetas de Usabilidade e UX.

Figura 2 – Protótipo da página com as técnicas de pré-processamento implementadas para a ferramenta UUX-Posts.

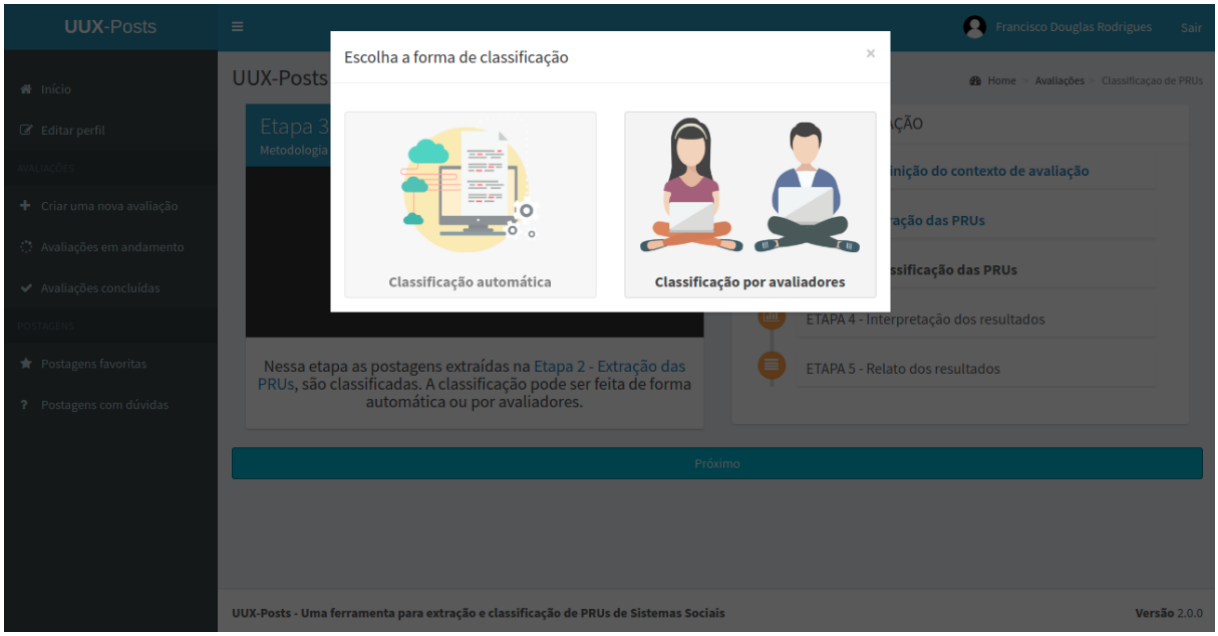


Fonte: <http://uuxposts.russas.ufc.br>

As últimas etapas, interpretação e relato dos resultados consistem na representação

dos resultados, gráficos para a representação, e um ambiente para relatar os resultados para os demais avaliadores responsáveis pela avaliação. Após essa nova versão, o avaliador poderá iniciar uma avaliação e continuar mais tarde, pois as avaliações são salvas de acordo com a etapa, para que o avaliador possa continuar em outro momento.

Figura 3 – Ferramenta UUX-Posts, etapa de classificação.



Fonte: <http://uuxposts.russas.ufc.br>

A nova versão da ferramenta foi desenvolvida utilizando principalmente tecnologias como HTML, CSS, JavaScript e PHP. Assim, as novas funcionalidades possivelmente serão integradas como um *script*, ao PHP, por se tratar de outra linguagem de programação, e pela ferramenta consistir na arquitetura WEB.

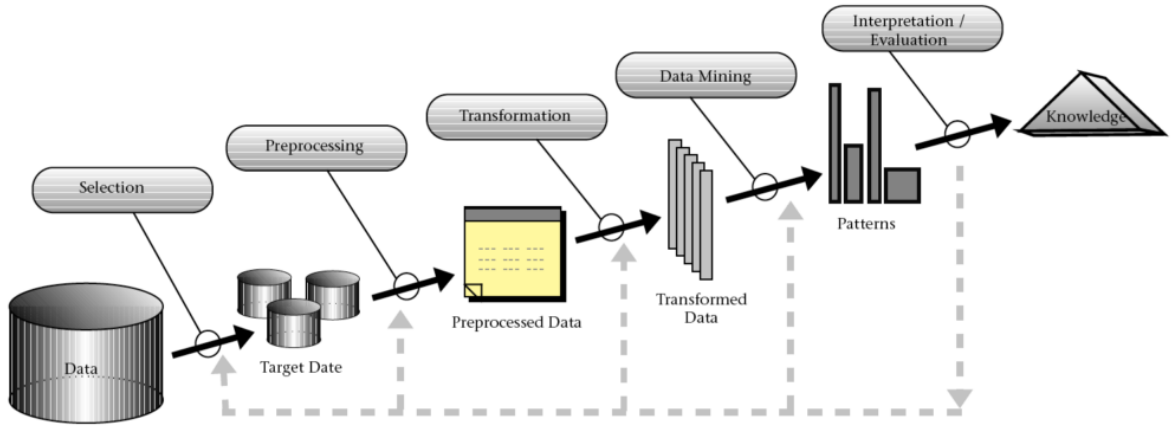
2.3 Descoberta de conhecimento em banco de dados

De acordo com Maimon e Rokach (2010) (tradução do autor) a “descoberta do conhecimento em banco de dados (KDD) é o processo organizado de identificar padrões válidos, novos, úteis e compreensíveis a partir de conjuntos de dados grandes e complexos.”. da Silva, Peres e Boscarioli (2017), também apresentam o KDD ao falar da mineração de dados, e definem seu objetivo, da seguinte forma:

> A descoberta do conhecimento em bases de dados tem como objetivo encontrar padrões intrínsecos aos dados nela contidos, apresentando-os de forma a facilitar sua assimilação com o conhecimento... tal descoberta está associada a um processo analítico, sistemático e, até onde possível, automatizado” (DA SILVA; PERES; BOSCARIOLI, 2017, p. 11).

Fayyad, Piatetsky-shapiro e Smyth (1996) apresentam as etapas do processo KDD (Figura 3). O processo é composto por seis etapas: 1) dados; 2) seleção; 3) pré-processamento; 4) transformação; 5) mineração de dados; e 6) interpretação/avaliação. Maimon e Rokach (2010) detalham a etapa de mineração de dados em três passos (descritos na seção 2.3.2), ao detalhar todas as etapas do processo KDD. A seguir serão detalhadas as duas etapas principais, foco deste trabalho: pré-processamento e mineração de dados.

Figura 4 – Etapas do processo KDD.



Fonte: Fayyad, Piatetsky-shapiro e Smyth (1996)

2.3.1 Pré-processamento de dados

Maimon e Rokach (2010) relatam que o pré-processamento de dados visa aprimorar os dados para as etapas de mineração. No qual pode incluir tarefas como: limpeza de dados, remoção de ruídos ou valores discrepantes e manipulação de valores ausentes.

Segundo Corrêa (2003) apud Mendes (2015) a limpeza de dados consiste em corrigir informações errôneas ou inconsistentes presentes nos dados. E citam outras tarefas como: *stemming*, *thesaurus* e *stopwords*. O *stemming* converte palavras para seu radical, por exemplo, livro, livrinho e livreto, ficaria somente "livr". O *thesaurus* converte palavras sinônimas para uma única palavra, por exemplo, as palavras, ruim, mal e desagradável, ficaria somente "ruim". E *stopwords* são palavras desnecessária para o contexto da avaliação, assim essas podem ser removidas. Geralmente são artigos, pronomes, conjunção e preposição. Podemos ainda acrescentar o *lemmatizer*, semelhante ao *stemming*, porém invés de converter a palavra para seu radical, ele converte ao seu *lemma*, por exemplo, fazer, fazendo e farei, ficaria somente "fazer",

em seu infinitivo verbal. A tokenização, que tem o objetivo de dividir os textos, deixando cada palavra como um *token*. Por fim, o sentenciador envolve deixar cada postagem em sentença(s), por exemplo, postagens como *"Eu adoro o twitter. Mas também amo jogar futebol"* fica duas sentenças a serem classificadas, sendo *"Eu adoro o twitter."* e *"Mas também amo jogar futebol"*.

De acordo com Han e Kamber (2006) os bancos de dados são altamente suscetíveis a dados ruidosos, ausentes e inconsistentes. Assim, os dados de baixa qualidade levarão a resultados de mineração de baixa qualidade. Dessa forma, o pré-processamento é uma importante etapa no KDD, tendo em vista que seu objetivo é preparar adequadamente os dados a serem minerados, ajudando assim a melhorar a qualidade dos dados e consequentemente os resultados da mineração.

Quando os dados constituem de postagens de usuários em SS, ou seja, textos em linguagem natural, o pré-processamento se torna mais relevante. Foster *et al.* (2011) apud Liu, Weng e Jiang (2012) relatam o resultado de um experimento da entidade Stanford (NER), no qual apresentou uma queda de desempenho de 90,8% para 45,8% nos tweets. Assim, é de grande importância normalizar o texto, provindo de redes sociais (LIU; WENG; JIANG, 2012). Devido esses textos possuírem muitos ruídos, como: *links*, palavras erradas, trechos em outras línguas, caracteres especiais (ex. # e @), excesso de vogais (ex. hooooooooje), dentre outros que são comuns nas redes sociais. Dessa forma, a limpeza e normalização de texto, envolve corrigir esses ruídos para deixar o texto mais enxuto.

2.3.2 Mineração de dados

Conforme mencionado anteriormente, Tan, Steinbach e Kumar (2009) definem a mineração de dados como uma parte integral no processo KDD, considerado um processo geral de conversão de dados brutos em informações úteis. No qual a mineração de dados envolve principalmente a aplicação dos algoritmos. Maimon e Rokach (2010) apresentam três passos na etapa da mineração de dados, do KDD.

Segundo Maimon e Rokach (2010), o primeiro passo é selecionar a tarefa ou método da mineração de dados apropriada ao conjunto de dados. As tarefas de mineração de dados são divididas em duas categorias principais, sendo: tarefas de previsão, com objetivo de prever o valor de um atributo baseado nos demais atributos; e tarefas descritivas que têm por objetivo derivar padrões que resumem os relacionamentos com os demais dados. Os métodos de previsão são: classificação (usada para valores discretos), e regressão (usada para valores contínuos). E os

métodos descritivos, são: as regras de associação (usadas para descobrir características associadas nos dados), agrupamento (usado para agrupar dados similares) e detecção de anomalias (usada para identificar observações anormais).

Segundo da Silva, Peres e Boscarioli (2017), na escolha da tarefa e técnica de mineração é importante que se tenha conhecimento do tipo da base de dados usada, que pode se manifestar de duas formas, estruturada e não estruturada. Os autores explicam que geralmente uma base de dados estruturada é organizada em estrutura tabular com linhas que armazenam ocorrência de um evento e as colunas apresentam características que descrevem uma instância de tal evento, assim esses dados geralmente são armazenados em tabelas relacionadas entre si. Enquanto que os dados em forma não estruturada, consiste em textos, imagens, vídeos e som. Para fins de mineração, os dados não estruturados precisam passar por um processo de pré-processamento, de forma que uma representação adequada seja produzida.

O segundo passo é selecionar o algoritmo específico para a tarefa selecionada anteriormente, tendo em vista a precisão e compreensibilidade. Por exemplo, Naive Bayes ou árvore de decisão para classificação (previsão), ou ainda K-means para agrupamento (descritiva). O terceiro passo da mineração de dados no KDD é implementar o algoritmo e aplicar no conjunto de dados. Em alguns casos é necessário o ajuste nos parâmetros do algoritmo, para obter o melhor resultado.

De acordo com Han e Kamber (2006), a árvore de decisão é uma estrutura em forma de árvore, onde cada nó significa um teste em um valor de atributo, cada ramificação representa um resultado do teste, e as folhas da árvore representam classes ou distribuições delas. Assim, as árvores de decisão podem ser facilmente convertidas em regras de classificação. Segundo Tan, Steinbach e Kumar (2009) o classificador Naive Bayes é baseado no princípio estatístico (teorema de Bayes) para combinar conhecimento prévio das classes com novas evidências colhidas dos dados. O teorema de bayes é dado por:

$$P(Y|X) = \frac{P(X|Y)}{P(X)}$$

Podemos também citar o algoritmo Máquina de Vetores de Suporte (em inglês, Support Vector Machines (SVM)), segundo Han e Kamber (2006) este algoritmo é utilizado para classificação linear e não linear, no qual usa o mapeamento não linear para converter os dados de treinamento em uma dimensão mais alta, para então procurar pelo hiperplano de separação ideal linear, ou seja, é feito a busca por um limite que separe os dados de uma classe da outra. Assim,

o SVM encontra este hiperplano usando vetores de suporte, tendo em vista que com a dimensão suficientemente alta, os dados de duas classes sempre podem ser separados por um hiperplano.

3 TRABALHOS RELACIONADOS

Nesta seção são apresentados alguns trabalhos que analisam textos em forma não estruturadas, ou melhor, em linguagem textual de usuários, para identificar padrões relevantes e não triviais, com uso de métodos de pré-processamento, e/ou de mineração de textos.

Magdy e Eldesouky (2017) propuseram uma ferramenta para classificação de tweets em cinco idiomas (Inglês, Francês, Alemão, Árabe e Russo), com intuito de avaliar os interesses dos usuários baseado em 14 categorias (um exemplo de categoria: futebol). Segundo os autores, este tipo de avaliação poderia ser útil para aplicações como pesquisa categorizada, análise de mídia social, perfis de usuários e sistema de recomendação. A ferramenta possui três componentes principais: a extração de recursos; modelos de classificação; e o módulo de detecção de idioma. Para o pré-processamento dos dados, foram realizadas a seguintes etapas: normalização de texto superficial (limpeza simples e normalização do texto, remoção de *links*, tokenização e remoção de pontuação); remoção de duplicatas e *retweets*; filtragem e mesclagem de categorias; e balanceamento de dados. Foi usado a abordagem *bag-of-words* por se tratar de um grande número de dados, para o treinamento foi usado o modelo *Distant-Supervision*, e para a classificação foi utilizado o SVM. Por fim, para validar os resultados da classificação, foram submetidos a profissionais, para verificar se foi perfeita, aceitável ou ruim, obtendo 56% perfeito, 28% aceitável e 16% ruim. Sobretudo, a ferramenta possui três módulos: o online, onde o usuário pode digitar a frase a ser analisada; o modo offline no qual a ferramenta pode ser baixada; e o modo em lote, no qual pode ser submetido um arquivo contendo as postagens, e em seguida é disponibilizado o resultado em um arquivo a ser baixado.

Junior e Merschmann (2016) propõem uma metodologia para mineração de texto e opiniões de usuários, para textos em português, o que, segundo os autores é uma lacuna quando comparado a outros trabalhos semelhantes, além disto, em seu trabalho também é considerado o contexto das postagens analisadas. A metodologia proposta é dividida em módulos estruturados pelo padrão *pipes-and-filters*, cada módulo com um objetivo específico, sendo: 1) detector de idiomas, para garantir que somente textos em português sejam selecionados para as próximas etapas; 2) normalização de texto (limpeza de caracteres especiais, tratamento de pontuação, tratamento de termos de vocabulário, correção ortográfica, etc.); 3) *tokenizer* (segmentação de frases); 4) extração de características (morfológico e sintático); 5) contextualizador; e 6) DTR (Transformação e Redução de Dados) que permite a aplicação de algoritmos para discretização de dados, seleção de atributos e balanceamento do conjunto de dados. Foram utilizadas as

seguintes ferramentas: Apache OpenNLP e Lucene, Language Detector, CoGroo e Smile. Além dos seguintes dicionários: VERO dictionary, Aspell, PB Unitex, e o LIWC. A base de dados consistiu de comentários extraídos no Google Play e postagens do Twitter. A metodologia utiliza na primeira etapa o SVM com a ferramenta Smile, para classificar os textos em um determinado contexto (por exemplo, saúde e economia), enquanto a segunda etapa consiste na mineração de opinião, que tem o objetivo de classificar, por exemplo, em positivo ou negativo.

Williams e Mahmoud (2017) analisaram as postagens dos usuários do Twitter, na qual se refere a algum sistema, no caso foram considerados 10 sistemas para a análise (Windows 10, Android, Apple-Support, Call of Duty, Chrome, Instagram, Minecraft, Snapchat, Visual Studio, e WhatsApp). O objetivo da análise é avaliar essas postagens para servir como suporte aos desenvolvedores, pois muitas vezes quando uma nova versão de um sistema é lançado, os usuários comentam no Twitter seu sentimento em relação a nova versão. Assim, esses comentários podem auxiliar a engenharia de software em próximas versões. Para a análise das postagens foram utilizados o SentiStrength e os classificadores supervisionados Naive Bayes e SVM, enquanto que para a extração foi utilizada a própria API do Twitter. A partir da classificação foi constatado que o desempenho para os classificadores supervisionados foi melhor comparado ao SentiStrength. De maneira geral, o objetivo do trabalho é criar um modelo com reconhecimento de emoções que possa fornecer recomendações para desenvolvedores de softwares, com base no humor do seu público.

Vu e Perez (2013) propuseram um sistema para extrair interesses de usuários em suas mensagens no Twitter. O processo proposto consiste em pré-processamento, extração de candidatos a palavras-chave e classificação delas. No pré-processamento é realizado um filtro em tweets que não estão na língua inglesa com a ferramenta *SRI Language Modeling*, eliminação de *stopwords*, normalização de texto com mapeamento morfológico e os tweets são anotados com uma tag a partir do TweetNLP Tagger. A extração de candidatos gera um conjunto de dados que podem ser palavras, frase, verbo etc. para serem classificados. Na classificação são utilizados os métodos: frequência do termo–inverso da frequência nos documentos (em inglês, *term frequency–inverse document frequency*, TFIDF), TextRank, Alocação Latente de Dirichlet (em inglês, *Latent Dirichlet Allocation*) (LDA-Rank) e Relevância-Interesse-Intervalo (RI-Rank), os dois últimos são variações do TextRank. O desempenho do sistema foi avaliado por meio de uma pesquisa online, onde os usuários selecionavam palavras-chave relevantes ou de interesse da população. Os resultados mostram que o TFIDF e o TextRank são confiáveis na classificação

de palavras-chaves, e que as duas podem sem complementares a fim de melhorar a classificação.

Li e Chen (2017) apresentam em seu trabalho, um estudo de formas para pré-processamento para textos em chinês, pois segundo o autor, o pré-processamento de textos em outras línguas não podem ser aplicados diretamente no texto chinês. Nesse trabalho é usado a remoção de urls, números, pontuações, palavras repetidas, substituição dos *emotions*, etc. É utilizado para classificação o Naive Bayes, SVM, Rede Neural Convolucional Network e *Long Short-Term Memory* (LSTM).

Janssens *et al.* (2013) propuseram a classificação de emoções de tweets em tempo real, usando técnicas como TFIDF, SVM, Bernoulli Naive Bayes, dentre outros. E Lim e Buntine (2014) analisaram tweets a fim de identificar as opiniões de usuários sobre produtos, por meio da mineração de opinião baseada em aspectos, onde os autores propuseram um novo modelo baseado em *Latent Dirichlet Allocation* (LDA), projetado para tweets, nomeado para Modelo de Tópico de Opinião do Twitter (TOTM). Vilares, Alonso e Gómez-Rodríguez (2013) propuseram um sistema híbrido que combina conhecimento lexical, sintático e semântico com técnicas de aprendizagem de máquina, para classificação de polaridade de tweets espanhóis. Como experimento neste trabalho, foram utilizados os algoritmos J48 (árvore de decisão) e Naive Bayes com a ferramenta WEKA. E Duran, Avanço e Nunes (2015) propuseram um normalizador de textos em português, com intuito de deixar o mais próximo possível do padrão esperado pelas técnicas de PNL. Esse normalizador consiste em: 1) uso de letras minúsculas em vez de maiúsculas; 2) a correção de erros de ortografia; 3) a substituição de gírias da Internet por palavras de linguagem padrão; e 4) a inserção de períodos faltantes.

A partir do estudo destes trabalhos foi possível listar algumas técnicas interessantes de pré-processamento a serem implementadas neste trabalho, dada a importância do mesmo para tratar os dados, quando estes se referem a linguagem textual de usuários, no qual envolve muitos ruídos, como gírias, abreviações, erros ortográficos, etc. que podem comprometer o resultado da classificação. No entanto nenhum dos trabalhos citados, apresentou o uso do *Stemming* e *Lemmatizer* para o pré-processamento. Além de poucos proporem um modelo com suporte a língua portuguesa.

Sobretudo, a partir do estudo de vários trabalhos, não foi encontrado algum que avaliasse a UUX do sistema, por meio da linguagem textual dos usuários, tampouco com o uso de métodos de pré-processamento e mineração de textos.

4 ESTUDO DE FERRAMENTAS SIMILARES

Este capítulo apresenta resultados do estudo realizado sobre ferramentas similares, foram consideradas ferramentas similares aquelas que de alguma forma fazem uso de técnicas de Processamento de Linguagem Natural (PLN) e/ou de Mineração de Dados (MD). O objetivo deste estudo é caracterizar essas ferramentas, tirando proveito do conhecimento prático para aplicar na ferramenta UUX-Posts. Assim, este capítulo está dividido em 4.1 Estudo de ferramentas de avaliação textual e 4.2 Estudo de ferramentas de PLN.

4.1 Estudo de ferramentas de avaliação textual

O estudo de ferramentas de avaliação textual não trouxe bons resultados, além dos já citados, como o de Magdy e Eldesouky (2017) e a UUX-Posts (MENDES, 2015; MENDES; FURTADO, 2017). A seguir, serão apresentadas três ferramentas: a MineraFórum, OncoViz e o IRAMUTEQ.

Segundo Azevedo, Behar e Reategui (2011) o MineraFórum é um software para análise qualitativa das mensagens redigidas em fóruns de discussão de ensino, capaz de calcular a relevância de cada postagem no fórum, usando técnicas de mineração de textos, com uso de grafos. O autores citam alguns recursos oferecidos pela ferramenta, dentre eles: o usuário pode informar um texto de referência sobre o tema em debate; utilizar um dicionário de sinônimos no processo de mineração; o usuário pode informar palavras que possuam equivalência semântica; calcular a relevância de cada postagens; exibir um gráfico com a média dos valores de relevância das mensagens, por cada autor; identificar se existem mensagens similares; possibilitar que os resultados do processo de mineração sejam salvo em um arquivo com extensão html; e exibir um relatório com informações sobre a análise das postagens.

Ainda de acordo com Azevedo, Behar e Reategui (2011), o MineraFórum calcula o valor de relevância de cada mensagem, a partir de três critérios: Relevância Temática da mensagem (RT), a Relevância de citações da Mensagem (RM), e a Similaridade da Mensagem (SM). Para a RT, é construído um grafo a partir do texto de referência do tema do debate, neste processo são removidos *stopwords*. As palavras mais relevantes do texto, correspondem aos vértices do grafo e as arestas são criadas a partir da proximidade entres as palavras. Após isso, é criado outro grafo para as mensagens redigidas no fórum. Por fim é calculada a relevância, a partir da análise da correspondência entre o primeiro e o segundo grafo, verificando quais

vértices são equivalentes (quando possuem conteúdos semelhantes), e calculando a quantidades de vértices dos dois grafos, a distância do vértices e o peso deles no seu grafo. Para a RM é calculada a divisão da quantidade de citação da mensagem pelo valor total no fórum, e para a SM, é analisada a correspondência entre os grafos de todas as mensagens, se a mensagem for semelhante a outra, o valor de SM será o valor de RT negativo. Assim, a relevância da mensagem é a média ponderada entre RT e RM. Assim, o MineraFórum utiliza a mineração de texto para avaliar se a mensagem está relacionado ao tema em debate e auxilia o professor a obter informações sobre os alunos (AZEVEDO; BEHAR; REATEGUI, 2011).

Ning *et al.* (2010) apresentam uma ferramenta de descoberta de informação e descoberta de conhecimento de textos centrados no usuário, chamada OncoViz, para a literatura relacionada ao câncer. Estes autores apresentam duas funcionalidades básicas da ferramenta: descobrir o conhecimento pertinente aos artigos notáveis baseado em um perfil de usuário, e entregar o conhecimento descoberto ao usuário por meio de uma ferramenta de visualização interativa.

De acordo com Ning *et al.* (2010) o processo da ferramenta de descobrir informações relevantes sobre problemas de saúde e efeitos colaterais de drogas envolve recuperar e representar documentos do domínio público e encontrar associações de entidade, para isso os autores apresentam três componentes. Primeiro, é construído um dicionário de sinônimos a partir de documentos obtidos por meio de um conjunto de palavras-chave com base no interesse do usuário. A partir desses documentos, outros termos relacionados serão gerados automaticamente usando um algoritmo de extração de termos (NING *et al.*, 2010). Segundo, os documentos são convertidos em estruturas para serem analisadas em unidades, assim, é calculada a importância dos termos a partir de sua frequência no documento. Por fim, é utilizado um método para encontrar a associação dos termos. Ning *et al.* (2010) apresentam em seu trabalho as fórmulas utilizadas para os cálculo de frequência e associação, além do sistema de visualização. Para a implementação foram utilizadas tecnologias, como PHP e MySQL.

Camargo e Justo (2013) evidenciam a ferramenta *Interface de R pour les Analyses Multidimensionnelles de Textes et de Questionnaires* (IRAMUTEQ). Uma ferramenta *open source* que usa o ambiente estatístico do software R, junto com a linguagem Python. Esta ferramenta disponibiliza diferentes tipos de análise de dados textuais, descritas a seguir.

1. Análise lexical clássica. Esta análise envolve: identificar e reformatar as unidades de texto, transformando Unidades de Contexto Iniciais (UCI) em Unidades de Contexto

Elementares (UCE); identificar a quantidade de palavras, frequência média e número de *hapax* (palavras com frequência um); pesquisar o vocabulário e reduzir as palavras com base em suas raízes (*lemmatizer*); criar dicionário de formas reduzidas, identificar formas ativas e suplementares (CAMARGO; JUSTO, 2013);

2. Análise de especificidades, no qual é possível associar diretamente os textos com variáveis descritoras e analisar a produção textual em função das variáveis de caracterização;
3. Análise para obter classes de UCE, usando o método da Classificação Hierárquica Descendente (CHD);
4. A análise de similitude, no qual se baseia na teoria dos grafos e possibilita identificar as coocorrências entre as palavras, para auxiliar na identificação da estrutura de um corpus textual;
5. Nuvem de palavras, correspondendo a uma análise lexical simples.

As análises citadas podem ser realizadas a partir de um conjunto de textos em único arquivo de texto, ou a partir de tabelas com instâncias em linha e palavras em coluna.

4.2 Estudo de ferramentas de Processamento da Linguagem Natural (PLN)

Com o estudo de ferramentas de PLN, podemos citar algumas ferramentas que de alguma forma usam o PLN. A seguir serão apresentadas as ferramentas: LX-Syllabifier[1], wReplace [2], Text Preprocessing Tool (TPT)[3], PreTexT II[4], PALAVRAS[5], WEKA[6] e a Extração de Conhecimento baseado em Aprendizagem Evolutiva (em inglês, *Knowledge Extraction based on Evolutionary Learning* - KEEL)[7]. Alguns destas são de propósito geral para mineração de dados, mas que fazem uso de PLN.

A ferramenta LX-Syllabifier é proposta no trabalho de Rodrigues *et al.* (2014). A ferramenta tem o recurso de silabação do português, por meio de duas abordagens, uma baseada em regras e outra em aprendizagem de máquina. A silabação foi implementada utilizando a divisão de base fonética. Para utilizar este recurso, é necessário informar uma palavra como entrada, para a ferramenta produzir uma nova *string* com o divisões da silabação marcadas

[1] http://lxcenter.di.fc.ul.pt/tools/pt/LXSyllabifierPT.html
[2] http://www.sharktime.com/us_wReplace.html
[3] http://sites.labic.icmc.usp.br/tpt/
[4] http://sites.labic.icmc.usp.br/pretext2/
[5] http://visl.sdu.dk/visl/pt/parsing/automatic/complex.php
[6] https://www.cs.waikato.ac.nz/m̃l/weka/
[7] http://www.keel.es/

com o caractere extra “|”. Na abordagem de aprendizagem de máquina, foi utilizado o método n-gramas para treinar os algoritmos Naive Bayes e Árvore de Decisão, que por sua vez foram implementados usando o NLTK.

O wReplace é um programa gratuito, que permite alterar/substituir múltiplas letras e palavras de cada vez (MENDES, 2015). Apesar de não ter sido encontrado trabalho sobre a ferramenta, podemos citar algumas características apresentadas em seu *site* (WREPLACE, 2013), tais como: fácil como o bloco de notas, rápido e livre (*open source*), possibilidade de exportar e importar regras de substituição, e suporte a caracteres nacionais (Unicode).

Para a ferramenta TPT, também não foi encontrado nenhum trabalho, mas na página onde seu conteúdo está hospedado (TPT, 2013), podemos encontrar algumas informações. O TPT converte os arquivos de texto com extensão *txt* em uma matriz de termo de documento, ou seja, no formato ARFF. Por meio da interface gráfica o usuário pode selecionar o diretório que contém os arquivos, assim o caminho do diretório dos arquivos será armazenado, o nome dos arquivos a e as opções de pré-processamento. As opções de pré-processamento são: língua (inglês ou português); *stem*; remoção de palavras-chave (*stopwords*); corte de frequência do documento (termos menos frequentes são removidos); e o peso (binário ou a frequência do termo).

O PreText II é uma nova versão da ferramenta PreText, apresentada no relatório técnico de Soares, Prati e Monard (2008). Segundo eles, o PreText é uma ferramenta que fornece pré-processamento utilizando funcionalidades como: n-grama, *stemming*, *stoplists*, cortes por frequência, taxonomias, normalizações, gráficos, medidas TF , TF-IDF , TF-linear, *boolean*, entre outras, com a abordagem *bag of words*. No mesmo trabalho são descritas algumas de suas características, tais como:

1. o módulo Start.pl lê o arquivo de configuração config.xml, para gerenciar os demais módulos;
2. o módulo maid.pm realiza a limpeza do conjunto de documentos inicias, remoção de *stopwords* e símbolos não relevantes, e gera os *stems* com o algoritmo de *stemming* para português, espanhol e inglês;
3. o módulo Ngram.pm gera os n-grama;
4. o módulo Report.pm realiza o processamento de taxonomias.

Segundo Soares, Prati e Monard (2008) com o pré-processamento de textos da nova versão, é possível calcular uma grande quantidade de informações dos documentos processados,

informações como arquivos limpos, *tokens, stems* e n-grama, além da tabela que representa atributo em linhas e valor em colunas. Sobretudo, esta ferramenta fornece métodos para pré-processamento de textos não estruturados.

De acordo com Fagundes e Vieira (2004) o PALAVRAS é um analisador sintático, bastante robusto por possibilitar a análise sintática de sentenças incorretas ou incompletas e por apresentar baixas taxas de erros. Esta ferramenta está disponível *online*, pelo projeto de pesquisa e desenvolvimento do Instituto de Linguagem e Comunicação (ISK) da Universidade do Sul da Dinamarca (SDU), chamado de VISL. Com ela é possível, por exemplo, extrair os *lemmas* de palavras. Segundo Bick (2000) o analisador é destinado a aplicativos como marcação de corpora, ensino de gramática e tradução automática.

Segundo Witten *et al.* (2016) o WEKA (*Waikato Environment for Knowledge Analysis*) foi desenvolvido na Universidade de Waikato na Nova Zelândia, estando disponível sob a licença GNU. A ferramenta disponibiliza métodos para os principais problemas de mineração de dados, tais como: classificação, agrupamento, regras de associação, regressão, e seleção de atributos. Além de recursos para pré-processamento (como filtros para discretização, normalização, seleção de atributos, transformação e combinação de atributos) e visualização de dados. De acordo com Witten *et al.* (2016) a principal interface do WEKA é o Explorer, onde dispõe de painéis para os métodos citados. Para cada problema de mineração selecionado, são disponibilizados vários algoritmos para minerar os dados. Assim, a ferramenta pode ser usada da seguinte forma:

> Suponha que você tenha alguns dados e queira construir uma árvore de decisão a partir dele. Primeiro, você precisa preparar os dados, depois acionar o Explorer e carregar os dados. Em seguida, você seleciona um método de construção de árvore de decisão, constrói uma árvore e interpreta a saída. É fácil fazê-lo novamente com um algoritmo de construção de árvore diferente ou um método de avaliação diferente. No Explorer, você pode alternar entre os resultados obtidos, avaliar os modelos construídos em conjuntos de dados diferentes e visualizar graficamente os modelos e os próprios conjuntos de dados, incluindo os erros de classificação que os modelos cometem. (WITTEN *et al.*, 2016, p. 16, tradução nossa).

Sobretudo, a ferramenta WEKA pode ser instalada em vários sistemas operacionais, como Windows, Linux e Mac OS. E possui um formato de arquivo próprio, o ARFF, sendo necessário que os arquivos submetidos aos métodos estejam neste formato.

Triguero *et al.* (2017) descrevem em seu trabalho a versão 3 da ferramenta KEEL, segundo eles o KEEL foi originalmente desenvolvido como uma ferramenta focada principalmente na implementação de algoritmos para problemas comuns de mineração de dados, como regras de regressão, classificação e associação, assim como técnicas de pré-processamento de dados. A ferramenta oferece uma interface simples para projetar experimentos com diferentes

conjuntos de dados e algoritmos de inteligência computacional, a fim de avaliar o comportamento dos algoritmos. (TRIGUERO *et al.*, 2017). Segundo os autores citados, na nova versão foram incluídas melhorias como: módulos para novos problemas e técnicas (como problemas semi-supervisionados, dados desequilibrados, aprendizado de múltiplas instâncias e descoberta de subgrupos) e novos algoritmos foram adicionados aos módulos originais, incluindo modelos de pré-processamento e aprendizado. Sobretudo, o KEEL está disponível sobre a licença GNU. Não necessita de bibliotecas adicionais e não precisa ser configurado ou instalado. E pode ser usado no Windows, Linux ou Mac OS.

Algumas outras ferramentas foram vistas, como o Orange[8], KNIME[9], RapidMiner[10] e TANAGRA[11]. Porém o estudo destas não foram aprofundados, dessa forma não serão apresentadas.

[8] https://orange.biolab.si/
[9] https://www.knime.com/software
[10] https://rapidminer.com/
[11] http://eric.univ-lyon2.fr/ ricco/tanagra/en/tanagra.html

5 IMPLEMENTAÇÃO DAS FUNCIONALIDADES

Este capítulo descreve o processo de implementação das funcionalidades propostas para a ferramenta UUX-Posts, apresentando como foram realizadas, tecnologias utilizadas, algoritmos, técnicas e todos os detalhes relevantes para o contexto da implementação. Ressaltando que o foco deste trabalho é contribuir para a ferramenta com a implementação das técnicas, não abrangendo assim sua integração. As técnicas definidas levaram em consideração a disponibilidade de bibliotecas ou módulos para PNL. Assim, a linguagem de programação Python foi a utilizada, por disponibilizar dessas bibliotecas como Linguagem Natural *Toolkit* (em inglês, *Natural Language Toolkit*, NLTK) e Scikit-learn, para PNL e mineração de dados.

De acordo com Pedregosa *et al.* (2011), o Python está se estabelecendo como uma das linguagens mais populares para computação científica, e é uma opção atraente para a análise exploratória de dados, devido por exemplo, ao amadurecido de bibliotecas científicas. Como o NLTK, um conjunto de módulos Python que fornece muitos tipos de dados PLN, tarefas de processamento, amostras e leitores de corpus, etc. (BIRD, 2006). E Scikit-learn que fornece implementações de última geração de muitos algoritmos de aprendizado de máquina bem conhecidos, enquanto mantém uma interface fácil de usar, e de ser integrada à linguagem Python (PEDREGOSA *et al.*, 2011).

Dessa forma, com consentimento do Python ser poderoso para análise exploratória de dados, a ideia foi implementar todas as técnicas utilizando essa linguagem, pois quando estas funcionalidades forem integradas na ferramenta, não devem envolver tanto esforço, apesar da ferramenta ter sido desenvolvida utilizando a linguagem PHP. Assim, este capítulo está dividido em: 5.1 Pré-processamento e 5.2 Classificação de PRUs, cada seção com sua implementação específica.

5.1 Pré-processamento

A implementação das funcionalidades de pré-processamento foi o primeiro passo na etapa de implementação. A seguir é descrita a implementação de cada técnica (como as representadas na Figura 2).

5.1.1 Sentenciador

Na implementação do sentenciador foi utilizada a biblioteca Python de processamento de texto chamada TextBlob, no qual dispõe de várias tarefas comuns de PLN, como extração de frase de substantivo, análise de sentimento, classificação, dentre outros (TEXTBLOB, 2018). O sentenciador implementado (Código-fonte 1) usa o TextBlob passando o texto como argumento e depois recebe as sentenças iterando em *TextBlob.sentences*. É importante que o sentenciador seja a primeira etapa no processo de pré-processamento, pois o TextBlob em muitos casos sentencia o texto a partir da identificação do ponto ("."), por isso esta etapa deve ser realizada principalmente antes da limpeza de texto.

Código-fonte 1 – Implementação do sentenciador

```
1 from textblob import TextBlob
2
3 def sentence(text):
4     blob = TextBlob(text)
5     sentences = [""".join(aux) for aux in blob.sentences]
6     return sentences
```

5.1.2 Limpeza de texto

A limpeza de texto implementada envolve conversão para minúsculas e remoção de: caracteres especiais; pontuações; dígitos decimais; acentuações; e excesso de vogais. Todas estas tarefas são integradas em uma única função. Na sua implementação foram utilizadas algumas funções simples ou módulos nativos do Python, como *lower*, *normalize* do *unicodedata* e o módulo de expressão regular (*re*), como é possível observar no Código-fonte 2.

Código-fonte 2 – Implementação da limpeza de texto

```
1 import re, string
2 from unicodedata import normalize
3
4 CASES_REMOVE_VOWELS = [("aaa","aa"), ("eee","ee"), ("iii","ii"), ("ooo","
      oo"), ("uuu","uu")]
5
6 def __remove_vowels__(text):
```

```
7       text = text.decode("utf-8")
8       for index in CASES_REMOVE_VOWELS:
9           while index[0] in text:
10              text = text.replace(index[0], index[1])
11      return text
12
13  def clear_text(text):
14      text = text.lower()
15      text = re.sub(r"[" + string.digits + "]+", " ", text)
16      text = re.sub(r"[" + string.punctuation + "]+", " ", text)
17      text = normalize("NFKD", text).encode("ASCII", "ignore")
18      text = __remove_vowels__(text)
19      return text
```

De acordo com a documentação Python Software Foundation (2001-2018), o *lower* é uma função para variáveis do tipo *string*, no qual retorna uma cópia da *string* com todos os caracteres em minúsculo, assim ele foi utilizado para todo o texto. O *re* é um módulo que fornece operações para expressões regulares, dessa forma este foi utilizado para remover dígitos decimais, pontuações e caracteres especiais. Enquanto, o módulo *unicodedata* fornece acesso ao banco de dados de caracteres Unicode, que define propriedades de caracteres e disponibiliza de várias funções, como o *normalize*, no qual retorna a forma normal para a sequência de caracteres Unicode UNISTR, o valor para remover acentos no qual foi utilizado é "NFKD".

Por fim, para remoção de excesso de vogais foi desenvolvida uma função ingênua, no qual verifica se o texto possui mais de duas vogais iguais consecutivas, em caso positivo é realizado a substituição no texto até que fique somente duas vogais. O ideal seria remover o excesso de vogais ao ponto da palavra ficar correta ou próximo disto, mas seria necessário uma função mais complexa, pois existem palavras como "álcool", "zoológico"e "coordenação"que se removido uma das vogais iguais consecutivas a palavra apresentaria erro ortográfico. Dessa forma, palavras como *"travandooooo"* e *"travandoooo"* ficaram iguais como *"travandoo"*, melhorando assim a classificação, tendo em vista a frequência das palavras que serão identificada.

5.1.3 Remoção de stopwords

Na remoção de *stopwords* foi necessário converter o texto em *tokens*, onde cada palavra do texto consiste em um *token*. Para este caso foi utilizado o método *TweetTokenizer* do

NLTK (NLTK, 2018c). A partir do texto dividido em *tokens* (palavras), é realizada a remoção de *stopwords*, para isto foi utilizada a base de dados de *stopwords* em português do corpus do NLTK (NLTK, 2018a), assim para cada *token* é verificado se ele pertence a esta base de dados, em caso positivo ele é removido. Por fim, os *tokens* são concatenados como ao texto original, mas dessa vez sem os *stopwords*, e retornado para a chamada da função.

Essa implementação pode ser observada no Código-fonte 3. Vale salientar que a base de dados usada aqui foi analisada e constatou-se que a mesma não contém verbos importantes para o contexto da metodologia MALTU, logo esta foi utilizada sem alterações.

Código-fonte 3 – Implementação da remoção de *stopwords*

```
1 from nltk.tokenize import TweetTokenizer
2 from nltk.corpus import stopwords
3
4 stopwords = stopwords.words("portuguese")
5 def remove_stopwords(text):
6     text = TweetTokenizer().tokenize(text)
7     text = [w for w in text if w not in stopwords]
8     return text
```

5.1.4 Stemming

A técnica *stemming* que converte a palavra ao seu radical, foi implementada utilizando o *stemmer* para português chamado *RSLPStemmer*, também do NLTK (NLTK, 2018b). Assim esta etapa também foi simples, pois o *RSLPStemmer* simplesmente recebe uma palavra e retorna o seu radical. Dessa forma, é necessário apenas iterar sobre o texto, chamando o *RSLPStemmer* para cada palavra, e obtendo assim um novo texto, como é possível observar no Código-fonte 4.

Código-fonte 4 – Implementação da remoção de *stopwords*

```
1 from nltk.stem import RSLPStemmer
2 st = RSLPStemmer()
3
4 def stemming(text):
5     sentence = [st.stem(word) for word in text]
6     return sentence
```

5.1.5 Lemmatizer

Para o *lemmatizer* não é utilizado bibliotecas ou módulos para fazer a sua função principal, pois a partir de pesquisas na literatura e em meios digitais não foi encontrado nenhum módulo Python que disponibilizasse esta técnica para a língua portuguesa, sendo encontrado para outras línguas, como o inglês, mas o foco deste trabalho é apenas o texto português. Dessa forma, para o *lemmatizer* foi utilizada uma base de dados contendo a representação cônica (*lemmas*) das palavras em português, disponibilizado por UNITEX-PB (2018) sobre a licença Pública Geral Menor (GNU).

A Base de Dados com *Lemmas* (BDL) citada contém mais de 9 milhões de palavras em português, disponível em um único arquivo de texto, onde cada linha do arquivo consiste na seguinte estrutura: "Palavra,*lemma*.Classe+traços:flexão". Então, para diminuir essa base de dados e consequentemente otimizar a busca do *lemma*, a BDL foi adaptada para o contexto da implementação. A adaptação consistiu em extrair somente as palavras e os *lemmas*, assim as demais informações contidas na BDL foram excluídas, também foram excluídas palavras que não são necessárias para o pré-processamento implementado, como palavras em sua conjugação nos sujeitos da língua formal, por exemplo "defamavam-lhes"(no geral palavras que contém hífen). Tendo em vista que esses tipos de palavras são pouco frequentes no contexto de sistemas sociais, e a partir da limpeza de texto do pré-processamento implementado, a palavra citada resultará em duas, sendo: "defamavam"e "lhes", pois a limpeza de texto remove o hífen, e quando utilizado a remoção de stopwords a palavra "lhes"deve ser removida, pois a mesma é considerada irrelevante, dessa forma quando for utilizado o *lemmatizer* as postagens não deve conter mais palavras como "defamavam-lhes"(apenas "defamavam"), logo não há necessidade de manter na BDL. Por fim, foram removidas palavras repetidas contidas na BDL, toda adaptação diminuiu a base consideravelmente, ficando em aproximadamente 864 mil palavras. O arquivo da BDL adaptada está disponível em: github.com/douglasrz/dataset-lemmatizer, disponibilizado pelo autor deste trabalho.

A BDL foi ainda dividida em vários arquivos, um para cada letra inicial das palavras com *lemmas* disponíveis (exemplo: a-lemmas, b-lemmas e c-lemmas). Assim o *lemmatizer* implementado percorre todo o texto e para cada palavra do texto é efetuada a busca no arquivo

específico, para isso é obtido a primeira letra da palavra e concatenado com a *string* "*-lemmas.txt*", em alguns casos quando o arquivo não é encontrado o erro é tratado e a palavra original é retornada, o mesmo caso para quando a palavra não é encontrada no arquivo. Além da implementação do *lemmatizer* propriamente, foi necessária a implementação de algoritmos para fazer a adaptação da BDL de forma automatizada. O Código-fonte 5 representa a implementação citada para converter palavra ao seu *lemma*.

Código-fonte 5 – Implementação do *lemmatizer*

```
1  from unicodedata import normalize
2
3  def lemmar(sentence):
4      clean = [__consultLemma__(word) for word in sentence.split()]
5      return " ".join(clean)
6
7  def __consultLemma__(word):
8      try:
9          path = "lemmas/" + str(word[0].lower()) + "-lemmas.txt"
10         file_new = open(path, "r")
11         lemmas = file_new.readlines()
12         for lemma in lemmas:
13             word_lemma = normalize("NFKD", lemma.split()[0]).encode("
                   ASCII", "ignore").decode("utf-8")
14             if word_lemma == word.lower():
15                 file_new.close()
16                 return normalize("NFKD", lemma.split()[1]).encode("ASCII"
                       , "ignore").decode("utf-8")
17         file_new.close()
18         return word
19     except OSError as err:
20         print("Error: " + str(err))
21         return word
```

Sobretudo, todas as técnicas implementadas são independentes entre si, com exceção da limpeza de texto, ficando assim a critério do usuário sobre quais técnicas utilizar para seu pré-processamento. Para que o pré-processamento ocorra de forma satisfatória é importante considerar a ordem das técnicas, além da escolha do *stemmig* ou *lemmatizer*. Pois o uso das

duas não terá significância, tendo em vista que uma palavra resultante do *lemmatizer* ao passar pelo *stemmig* resultará no mesmo resultado quando usada antes do *lemmatizer*, e para a palavra resultante do *stemmig* o lemmatizer não encontrará seu *lemma*, logo a mesma é retornada. Dessa forma, é recomendado que o pré-processamento seja realizado na seguinte ordem: 1ª sentenciador, 2ª limpeza de texto, 3ª remoção de stopwords, e 4ª *stemming* ou *lemmatizer*. Vale salientar ainda, que foi implementado para os experimentos o programa principal fazendo uso das técnicas citadas, e manipulando entrada/saída de cada chamada das técnicas, pois a mesma deve ser implementada na integração com a ferramenta.

5.2 Classificação de PRUs

Na mineração de dados para a classificação de PRUs os critérios de escolha das técnicas envolveram o desempenho, disponibilidade, e principalmente a compreensibilidade da técnica. Segundo Maimon e Rokach (2010) podemos considerar, por exemplo, a compreensibilidade ou precisão na escolha do algoritmo de mineração, sendo que diferentes abordagens podem ser mais apropriadas para determinados algoritmos. O mesmo cita que para compreensibilidade, a técnica da árvore de decisão seria uma boa escolha.

Dias (2002) apresenta um estudo sobre parâmetros na escolha das técnicas e ferramentas de mineração de dados, segundo a autora, basicamente os principais parâmetros na escolha da técnica são:

a) tipo de problema de descoberta de conhecimento a ser solucionado, neste caso refere-se a classificação;
b) características dos dados. A autora apresenta algumas características e as técnicas referentes, como: variáveis categóricas, com as técnicas, descoberta de regras de associação e árvores de decisão; e texto sem formatação, com a técnica raciocínio baseado em casos;
c) forma de aplicação da mineração de dados (processo de verificação ou processo de descoberta);
d) disponibilidade de ferramenta de mineração de dados.

Com isso a técnica utilizada para classificar as PRUs é definida tendo em vista a disponibilidade na linguagem Python, o fator compreensibilidade, e que o desempenho na classificação de PRUs seja superior comparada a versão 1 da ferramenta. Assim, na implementação foram utilizadas algumas bibliotecas do Python, como Pandas e Scikit-learn. Pandas é uma

biblioteca de código aberto que fornece estruturas de dados de alto desempenho e ferramentas de análise de dados fáceis de usar (Pandas, 2018). Com o Pandas, foram utilizados os módulos de leitura de arquivo CSV e o *dataframe* para estrutura de dados em tabela, com atributo-valor. Enquanto que com o Scikit-learn foram utilizadas as seguintes funções ou módulos: *Joblib*, *Pipeline*, *TfidfVectorizer*, *GridSearchCV*, *train_test_split* e os classificadores utilizados, sendo eles: *DecisionTreeClassifier*, NuSVC, e o MultinomialNB, encontrados em Pedregosa *et al.* (2011).

O módulo *Joblib* foi utilizado para salvar o modelo treinado pelo algoritmo de classificação e para carregar o modelo em outro momento. O *Pipeline* foi utilizado para aplicar a transformação do conjunto de dados para o estimador final, ou seja o pipeline recebe uma técnica de transformação de dados e o estimador para então adaptar esses dados para que a classificação seja realizada, assim com o pipeline é possível armazenar o transformador em *cache* (PEDREGOSA *et al.*, 2011).

O *TfidfVectorizer* converte o texto em uma matriz de recursos do TF-IDF (PEDREGOSA *et al.*, 2011), e assim este foi utilizado para montar o modelo *bag-of-words* dos textos. Pois o conjunto de dados a serem classificados consistem em textos, no qual devem ser adaptados ao formato esperado pelo classificador. Segundo Han e Kamber (2006), a frequência do termo – frequência do documento inverso (em inglês, *Term Frequency–Inverse Document Frequency* (TF-IDF)) é um modelo de espeço vetorial de recuperação de informação para classificação de documento, este modelo combina a Frequência do Termo (TF) com a Frequência do Documento Inverso (IDF), dado pela fórmula:

$$TF - IDF(d;t) = TF(d;t) \times IDF(t)$$

A função *"train_test_split"* foi utilizada para dividir o conjunto de dados em dois subconjuntos, sendo o conjunto de teste e o de treinamento, este caso é utilizado para treinar o modelo e obter a precisão pelo próprio classificador. O recurso *GridSearchCV* é utilizado para obter os melhores parâmetros para o classificador e para o *TfidfVectorizer*, esta função foi utilizada somente antes da consolidação da implementação, pois depois disto os melhores parâmetros já têm sido identificado por ele. Assim, foram realizadas algumas execuções com a inclusão do Código-fonte 6 do *GridSearchCV* na função *train_model* do Código-fonte 7, a partir destas execuções a função fornece os parâmetros que obtiveram o melhor desempenho, então estes foram usados na implementação final. Vale salientar que o Código-fonte 6 é utilizado para o algoritmo *DecisionTreeClassifier*, para os demais algoritmos é necessário apenas alterar os

parâmetros especifico do algoritmo, permanecendo os do *TfidfVectorizer*. Alguns parâmetros foram ignorados por não terem significância para o problema, tendo como base sua documentação em Pedregosa *et al.* (2011).

Código-fonte 6 – Implementação do GridSearchCV para o algoritmo *DecisionTreeClassifier*

```
param_grid = {
  "algorithm__criterion": ["gini", "entropy"],
  "algorithm__splitter": ["best", "random"],
  "algorithm__class_weight": [None, "balanced"],
  "vectorizer__decode_error": ["strict", "ignore", "replace"],
  "vectorizer__analyzer": ["word", "char"],
  "vectorizer__binary": [False, True],
  "vectorizer__norm": ["l1", "l2"],
  "vectorizer__use_idf": [False, True],
  "vectorizer__smooth_idf": [False, True],
  "vectorizer__sublinear_tf": [False, True]
}
grid = GridSearchCV(pipe, param_grid, cv=5, scoring="accuracy")
grid.fit(X_train, Y_train)
print(grid.best_score_)
print(grid.best_params_)
```

Nos classificadores é utilizado módulo *Tree* para utilizar o algoritmo *DecisionTreeClassifier* (algoritmo árvore de decisão para classificação). O algoritmo NuSVC é uma variação do algoritmo SVM (detalhado na seção 2.3.2), no qual usa um parâmetro para controlar o número de vetores de suporte (PEDREGOSA *et al.*, 2011). E o MultinomialNB é um algoritmo Naive Bayes multinomial, que faz uso da distribuição multinomial, de acordo com Pedregosa *et al.* (2011) ele é adequado para classificação com recursos discretos, como é o caso da contagens de palavras para a classificação de texto. Para utilizar este dois últimos algoritmos é necessário apenas utilizar o trecho do Código-fonte 7 que está comentado, referente ao algoritmo desejado, e ainda alterar os parâmetros do *TfidfVectorizer* no qual apresentaram o melhor desempenho, para o NuSVC é recomendado alterar apenas o parâmetro *norm* para *l2*, enquanto que para o MultinomialNB pode continuar com os mesmos.

Assim, o algoritmo do programa de classificação de PRUs apresentado no Código-fonte 7, recebe como entrada um arquivo CSV com as postagens já classificadas para treinar

o modelo de classificação (Parâmetro 1 - P1), um arquivo a ser salvo como o novo modelo (Parâmetro 2 - P2), e o texto a ser classificado (na função *main*). Primeiramente o programa verifica se P2 é um arquivo existente, caso positivo significa que já existe um modelo de classificação treinado, logo não necessita treiná-lo novamente, neste caso a função *train_model* não é utilizada. Caso contrário, o modelo é treinado, mas antes disto é realizado a preparação dos dados com o *dataframe* (função *prepara_data*) e em seguida o modelo é treinado e salvo para uso posterior.

Código-fonte 7 – Implementação do classificador

```
1  import os.path
2  import pandas as pd
3  from sklearn.model_selection import train_test_split
4  from sklearn.externals import joblib
5  from sklearn.pipeline import Pipeline
6  from sklearn.feature_extraction.text import TfidfVectorizer
7  from sklearn.naive_bayes import MultinomialNB
8  from sklearn.svm import NuSVC
9  from sklearn import tree
10
11 def train_model(data_frame, trained_model):
12     data_frame = data_frame.dropna()
13     taxa_train = 0.8
14     vectorizer = TfidfVectorizer(
15         lowercase=False,
16         norm=None, #norm="l2" to NuSVC
17         smooth_idf= False,
18         sublinear_tf= True,
19         ecode_error= "replace"
20     )
21     algorithm = tree.DecisionTreeClassifier(class_weight ="balanced")
22     #algorithm = MultinomialNB()
23     #algorithm = NuSVC(decision_function_shape="ovo", nu=0.8 )
24     pipe=Pipeline([("vectorizer", vectorizer), ("algorithm", algorithm)])
25     X_train, X_test, Y_train, Y_test = train_test_split(
26         data_frame.Postagem, data_frame.Categoria, train_size=taxa_train
27     )
28     pipe.fit(X_train, Y_train)
```

```
29      print("\nPrecisao com {:.0%} de dados testados: {:.1%}\n".format(
30          1 - taxa_train, pipe.score(X_test, Y_test)))
31      pipe.fit(data_frame.Postagem, data_frame.Categoria)
32      joblib.dump(pipe, trained_model)
33
34  def classification(text, trained_model):
35      pipe = joblib.load(trained_model)
36      output = pipe.predict([text])
37      print("Postagem: ", text, " Categoria: %s \n" % output[0])
38      return output[0]
39
40  def prepare_data(model_to_train):
41      if model_to_train:
42          plan_dataset = pd.read_csv(model_to_train, sep=";")
43          plan_dataset = ((r["Postagem"], r["Categoria"]) for i, r in
        plan_dataset.iterrows())
44      lines = []
45      for postagem, categoria in plan_dataset:
46          lines.append((postagem, categoria))
47      data_frame = pd.DataFrame(lines)
48      data_frame.columns = ["Postagem", "Categoria"]
49      data_frame.to_csv("../data/bag_words.ods", sep=";", encoding="utf-8")
50      return data_frame
51
52  def main(model_to_train, trained_model, text):
53      if not os.path.isfile(trained_model):
54          data_frame = prepare_data(model_to_train)
55          train_model(data_frame, trained_model)
56      if text:
57          return classification(text, trained_model)
```

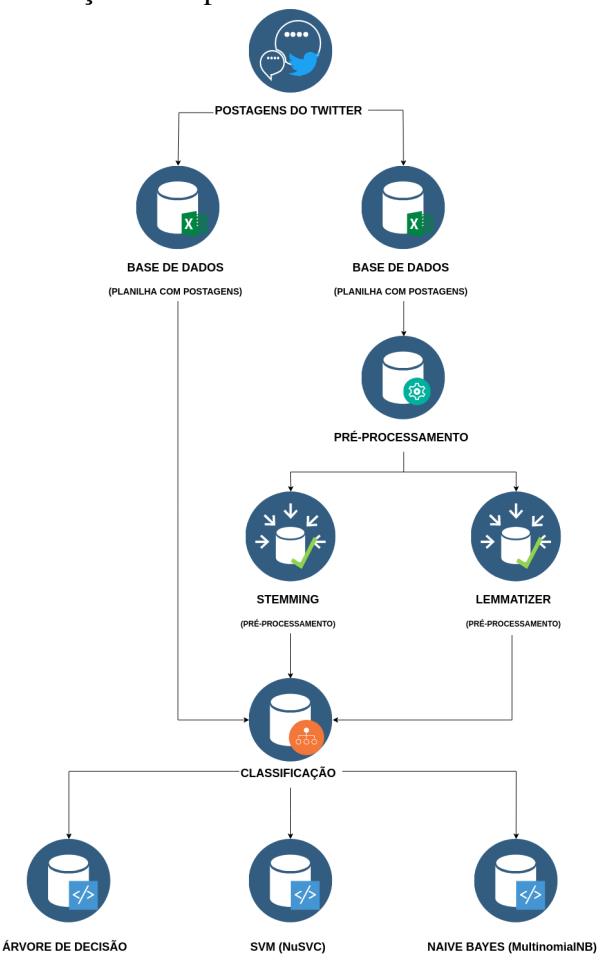
Para o treinamento do modelo (função *train_model*), a base de dados é dividida em treinamento e teste, sendo 80% e 20% dos dados respectivamente. Em seguida é obtido o valor de precisão e o modelo é treinado novamente, desta vez com toda a base de dados, no qual é salvo para uso posterior. Por fim, o programa classifica o texto a partir do modelo de classificação treinado, utilizando o *Pipeline* para fazer a classificação do texto pelo algoritmo usado. Vale salientar que o único parâmetro obrigatório esperado pelo classificador é o texto a

ser classificado, assim o usuário não necessita passar P1 e P2, pois estes são opcionais, tendo em vista que o programa já contém esses dados internamente para classificação de PRUs de postagens do Twitter, para outro contexto é necessário passar P1 e P2, ou ainda para tentar melhorar a precisão do modelo.

6 CONDUÇÃO DO EXPERIMENTO

Este capítulo descreve as diretrizes do processo de condução seguido no experimento para avaliação dos resultados das funcionalidades implementadas. A Figura 5 representa as etapas do experimento, de início são obtidas postagens do sistema social Twitter consistindo na base de dados usada durante todo o processo, este é detalhado na próxima seção. Em seguida é realizado o pré-processamento discutido na seção 6.2, e por fim a mineração de texto com três diferentes algoritmos discutidos na seção 6.3.

Figura 5 – Processo de condução do experimento



Fonte: Elaborado pelo autor

6.1 Base de dados

A base de dados utilizada para avaliar os resultados das funcionalidades implementadas consiste em uma planilha de postagens obtidas no Twitter, no qual foram classificadas manualmente em PRU e não-PRU, e a partir delas foram obtidas uma amostra contento 2680 pos-

tagens (1340 PRU e 1340 não-PRU). Sendo necessário ser dividida em um conjunto de postagens para treinamento do modelo de classificação e outro conjunto para os testes, ficando 80% dos dados para treinamento e 20% para testes, como mencionado anteriormente. Vale salientar que a base de dados foi classificada pelo autor deste trabalho, como atividade realizada no projeto de iniciação científica MALTU, e posteriormente foi validada pela orientadora professora Dra. Marília Soares Mendes.

A Tabela 2 apresenta uma pequena amostra da base de dados usada. Nela é possível observar como o texto possui muitos ruídos dificultando a classificação automática, e consequentemente destacando a importância do pré-processamento. Assim, para o experimento foi utilizada a base de dados com o pré-processamento para a classificação, e a base de dados sem o pré-processamento para outra classificação, a fim de comparar e avaliar o quanto que o pré-processamento influência no resultado final. Ressaltando que para os dois casos foi utilizada a mesma base de dados inicial.

Tabela 2 – PRUs e não-PRUs

Postagem	Categoria
Mamãe sabe que eu tenho twitter, porque somos cúmplices kkkkkkkkk	nPRU
Laari SEXTOUUU !! Bonitaa Salveeeee!!!!!!!!!!! Olha Apareceu sugestao sua do twitter aqui!	nPRU
agora vou sofrer aqui no Twitter porque sou dessas	nPRU
quero trancar meu Twitter mais esqueci aonde q é q tranca	PRU
Acho as notificações do twitter tão bonitinjas	PRU
Sr. Twitter mais uma vez obrigada pela atualização inútil. #BotãoEditarÉOQueEuTePeço	PRU

Fonte: Elaborado pelo autor com postagens do Twitter.com

6.2 Pré-processamento

A partir da base de dados no qual consiste na matéria-prima da mineração de dados, é possível aplicar o pré-processamento com o objetivo de limpar a base, removendo ruídos que podem afetar o resultado final. Neste experimento o pré-processamento é a fase intermediária, capaz de tratar adequadamente o texto da base de dados para a aplicação dos algoritmos de classificação. Assim, o algoritmo por si só não é suficiente para apresentar bons resultados, uma vez que a base de dados não é estruturada, além de apresentar os problemas citados.

No pré-processamento são utilizadas todas as técnicas implementadas, citadas na seção 5.1, com a ordem das técnicas como recomendado. Salientando que é necessário aplicar o *stemming* e *lemmatizer* separadamente, pois o uso das duas técnicas é insignificante, assim foram realizados dois pré-processamentos consistindo em duas novas base de dados como resultado,

uma com o pré-processamento incluindo o *stemming*, e outra incluindo o *lemmatizer*.

Contudo, além de avaliar a importância do pré-processamento implementado para a classificação, o experimento tem o intuito de avaliar individualmente as duas técnicas (*stemming* e *lemmatizer*) a partir dos resultados obtidos com a classificação para as duas bases de dados referentes as técnicas, pois resultados diferentes podem significar que uma técnica é superior comparada a outra.

6.3 Mineração de texto

Para a mineração de texto, foi realizada a classificação de postagens em PRUs e não-PRUs com três diferentes algoritmos, sendo eles: a Árvore de Decisão, NuSVC do modelo Máquina de Vetores de Suporte (do inglês, *Support Vector Machine* (SVM)) e MultinomialNB do modelo Naive Bayes (discutidos no capítulo 2). Esses algoritmos foram selecionados por não exigirem grandes mudanças na implementação e pela possibilidade de aplicá-los na mesma base de dados, com a mesma representação.

O objetivo em realizar o experimento com os três algoritmos é compará-los analisando qual apresenta o melhor resultado para a classificação com esses tipos de dados, e assim pode ser integrado na ferramenta aquele que apresenta o melhor desempenho, ou ainda podem ser utilizados os três, deixando a critério do usuário sobre qual algoritmo deseja utilizar. Para isto é necessário apenas adicionar um condicional sobre a entrada do usuário, para então utilizar o desejado.

Para a validação dos resultados foi utilizado o modelo de Validação Cruzada (em inglês, *Cross Validation* (CV)), com o módulo *Model Selection* do Scikit-Learn, utilizando a função *cross_validate*, usado como no trecho do Código-fonte 8. Este recurso foi utilizado para obter os resultados das métricas: precisão, revocação (do inglês *recall*), e acurácia. De acordo com Olson e Delen (2008) essas métricas são dadas por:

$$\text{precisão} = \frac{TP}{TP+FP}$$

$$\textit{recall} = \frac{TP}{TP+FN}$$

$$\text{acurácia} = \frac{TP+TN}{TP+TN+FP+FN}$$

Onde:

- TP = número de verdadeiros positivos;
- FP = número de falsos positivos;
- FN = número de falsos negativos;
- TN = número de verdadeiros negativos.

Segundo Olson e Delen (2008), os termos positivo e negativo refere-se a classe alvo da classificação, enquanto que os termos verdadeiro e falso está relacionado ao resultado da classificação, por exemplo, no contexto deste trabalho, onde o objetivo é classificar PRUs, o número de verdadeiros positivos, refere-se a classe de PRUs que foram classificadas corretamente, enquanto que o número de verdadeiros negativos refere-se a classe de nPRUs classificadas corretamente. Além das métricas obtidas com o CV, também é obtida a precisão com o próprio classificador, ou seja, os classificadores do *Scikit-learn* utilizados já disponibilizam a função de obter a precisão a partir de uma classificação, então também foi considerado esta métrica para discussão dos resultados, esta é citada adiante como precisão do próprio classificador.

Código-fonte 8 – Trecho da implementação para a validação cruzada

```
1  from sklearn.model_selection import cross_validate
2
3  scoring = ["precision_micro", "recall_micro", "accuracy"]
4  scores = cross_validate(
5      pipe,
6      data_frame.Postagem,
7      data_frame.Categoria,
8      cv=10,
9      scoring=scoring,
10     return_train_score=False
11 )
12 PRECISION_CROSS_VALIDATION = sum(scores["test_precision_micro"])/10
13 RECALL_CROSS_VALIDATION = sum(scores["test_recall_micro"])/10
14 ACCURACY_CROSS_VALIDATION = sum(scores["test_accuracy"])/10
```

Segundo Tan, Steinbach e Kumar (2009) a CV é um método para avaliar o desempenho de um classificador, onde cada registro é utilizado o mesmo número de vezes para treinamento e uma vez para teste. Este método usa k partições, sendo executado k vezes, a cada execução uma partição é utilizada para teste e as demais para treinamento, ao final das k execuções cada partição deve ter sido utilizada exatamente uma vez para teste. Assim o total de

erro é obtido pela soma dos erros de todas as k execuções.

Para este experimento o modelo de validação cruzada foi utilizada com 10 partições, assim obtemos a média aritmética das dez, que consiste no valor final de cada métrica para uma execução. Para este caso, é utilizada toda a base de dados dividida no número de partições especificadas. Enquanto que para a precisão obtida através do próprio classificador, é utilizada a base de dados dividida em treinamento e teste, com 80% e 20% respectivamente, neste caso os 20% são utilizados para fazer a classificação e obter o resultado da precisão.

Sobretudo, para manter a consistência dos resultados a implementação é executada dez vezes para cada algoritmo com a mesma base de dados, e a partir dessas execuções é obtida a média aritmética para cada métrica. Assim são obtidos os resultados dos três algoritmos para as três bases de dados diferentes. Esses resultados são apresentados e discutidos no próximo capítulo.

7 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Esta seção apresenta os resultados da implementação realizada neste trabalho. Resultados estes sobre o pré-processamento, mostrando como suas técnicas estão devolvendo o texto, que antes continha ruídos. E sobre o desempenho dos diferentes algoritmos de classificação, mostrando os resultados das métricas.

7.1 Pré-processamento

O pré-processamento foi realizado sobre a base de dados citada na seção 6.1, com todas as técnicas de pré-processamento implementadas. Esta etapa, resultou em duas novas bases, como citado anteriormente, uma com o uso do *stemming* (Tabela 3) e a outra com o *lemmatizer* (Tabela 4). Abaixo é apresentada a amostra da base de dados resultante do pré-processamento para os dois casos, com a mesma amostra da base apresentado na Tabela 2.

Tabela 3 – Pré-processamento com a técnica *stemming*

Postagem	Categoria
mama sab twitt porqu cumpl kkkkkkkkk	nPRU
laar sextouu bonita salve olh aparec sugesta twitt aqu	nPRU
agor vou sofr aqu twitt porqu dess	nPRU
quer tranc twitt esquec aond q q tranc	PRU
ach notificaco twitt tao bonitinj	PRU
sr twitt vez obrig atualizaca inutil botaoeditareoqueeutepec	PRU

Fonte: Elaborado pelo autor com postagens pré-processadas obtidas no Twitter.com

Tabela 4 – Pré-processamento com a técnica *lemmatizer*

Postagem	Categoria
mamae saber twitter porque cumplice kkkkkkkkk	nPRU
laari sextouu bonitaa salvee olhar aparecer sugestao twitter aqui	nPRU
agora ir sofrer aqui twitter porque dessar	nPRU
querer trancar twitter esquecer aonde q q tranca	PRU
achar notificacao twitter tao bonitinjas	PRU
sr twitter vez obrigado atualizacao inutil botaoeditareoqueeutepeco	PRU

Fonte: Elaborado pelo autor com postagens pré-processadas obtidas no Twitter.com

O objetivo desta implementação é propor um pré-processamento exaustivo, capaz de remover os ruídos presente no texto, além de diminuir sua dimensionalidade. De modo que este processo possa contribuir ao resultado obtidos pelo classificador, de forma positiva. Na próxima seção são apresentados os resultados da classificação, considerando o pré-processamento realizado na base, e levando conta as duas técnicas *stemming* e *lemmatizer* separadamente.

7.2 Resultados das métricas

Os resultados são obtidos para três diferentes algoritmos, assim como para três diferentes bases de dados oriundas de uma mesma. Dessa forma, esta seção está dividida em 7.2.1 Resultados do algoritmo Árvore de Decisão, 7.2.2 Resultados do algoritmo NuSV, e 7.2.3 Resultados do algoritmo MultinomialNB. Para cada seção citada, são apresentados os resultados para as três bases de dados.

7.2.1 Resultados do algoritmo Árvore de Decisão

Os resultados obtidos com o algoritmo árvore de decisão estão representados na Figura 6, as métricas são as citadas anteriormente, sendo elas: precisão, *recall* e acurácia com o modelo de Validação Cruzada (em inglês, *Cross Validation (CV)*), e a precisão obtida pelo próprio classificador. As mesmas utilizados nos demais algoritmos.

A classificação com a base de dados resultante do pré-processamento com a técnica *stemming* apresentam os melhores resultados, somente com a precisão obtida pelo próprio classificador em que a classificação com dados limpos usando o *lemmatizer* foi ligeiramente melhor. Enquanto que o pior desempenho foi a classificação com a base de dados com ruídos, para todas as métricas, apesar da diferença ser pequena comparada a classificação com dados oriundo do *lemmatizer* para as métricas do CV, ficando apenas 0,25% menor.

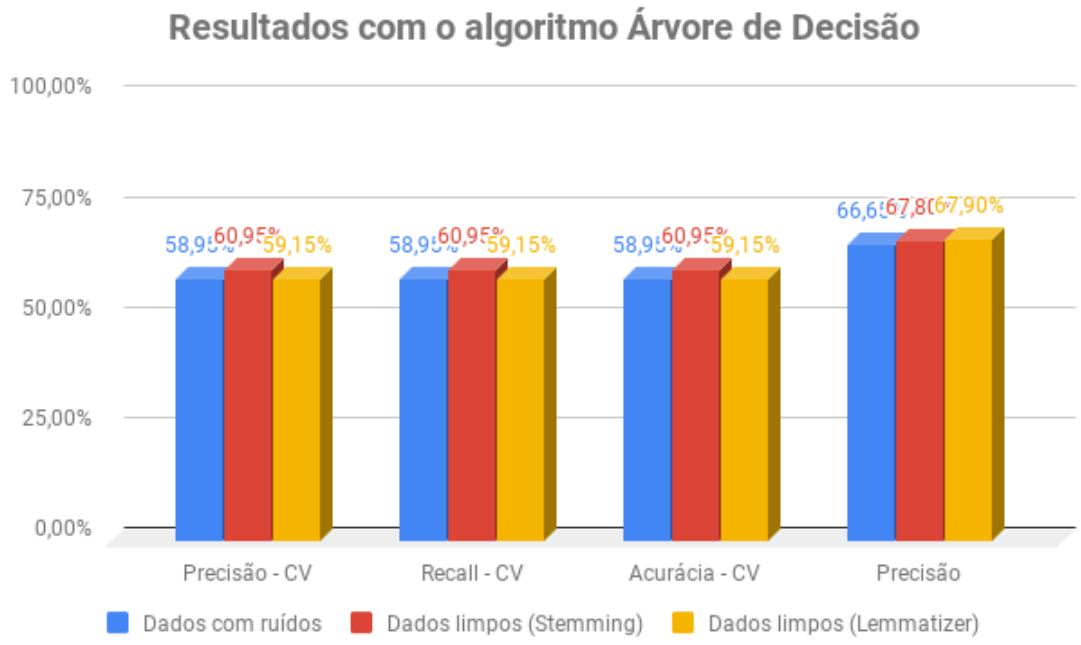
7.2.2 Resultados do algoritmo NuSVC

A Figura 7 apresenta os resultados obtidos com o classificador NuSVC (variação do SVM). Como é possível observar os resultados utilizando a base de dados com ruídos também se apresentam como os piores resultados, os demais estão iguais para as métricas do modelo CV, enquanto que para a precisão obtida pelo próprio classificador a base de dados oriunda do pré-processamento com a técnica *lemmatizer* foi um pouco superior.

7.2.3 Resultados do algoritmo MultinomialNB

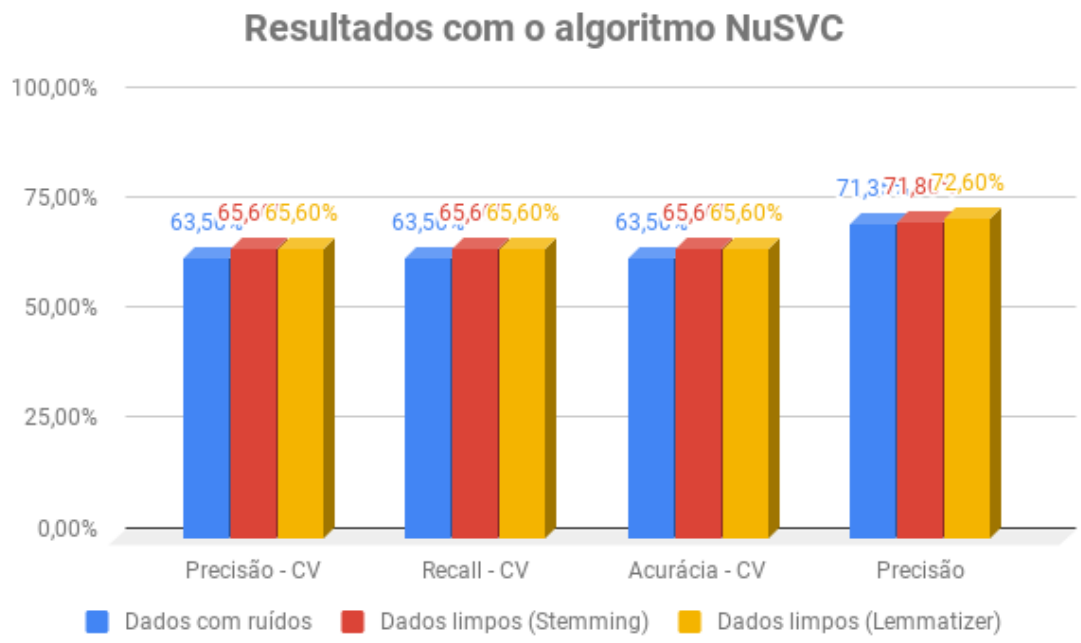
Com o algoritmo MultinomialNB (classificador Naive Bayes) os resultados são apresentados na Figura 8. Com esse algoritmo a classificação com dados limpos usando a técnica *stemming* foi superior aos demais. Mas o resultados intrigante é sobre os dados com ruídos no qual apresenta superioridade com a precisão do próprio classificador sobre a classificação

Figura 6 – Resultados com o algoritmo Árvore de Decisão



Fonte: Elaborado pelo autor

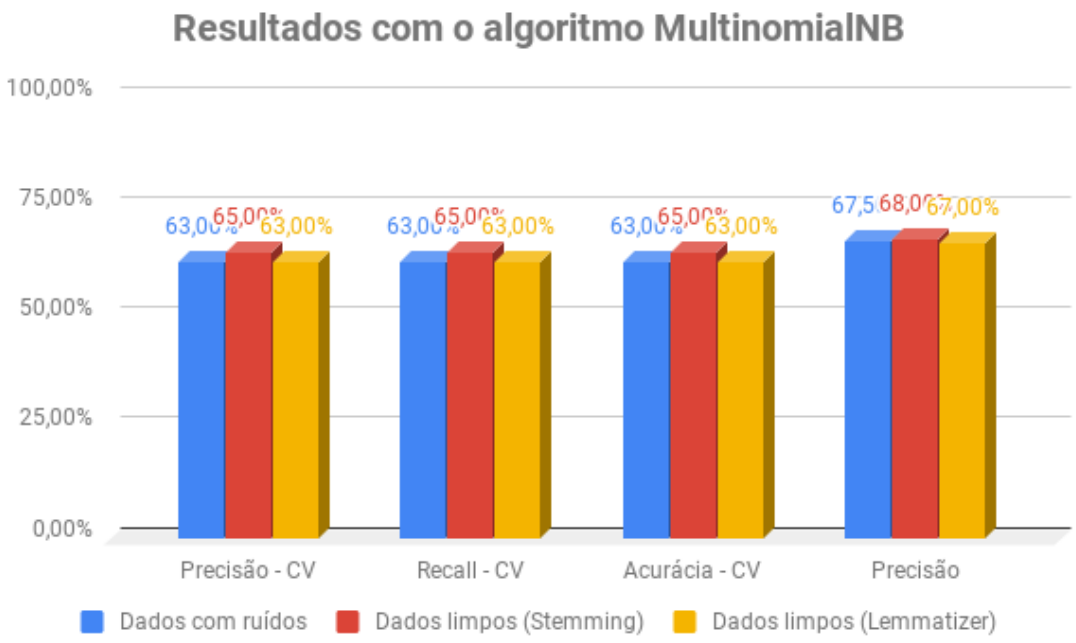
Figura 7 – Resultados com o algoritmo NuSVC



Fonte: Elaborado pelo autor

com dados limpos utilizando o *lemmatizer*, apesar da diferença ser pequena este fato não deixa de ser importante, uma vez que o objetivo do pré-processamento é melhorar o desempenho da classificação.

Figura 8 – Resultados com o algoritmo MultinomialNB



Fonte: Elaborado pelo autor

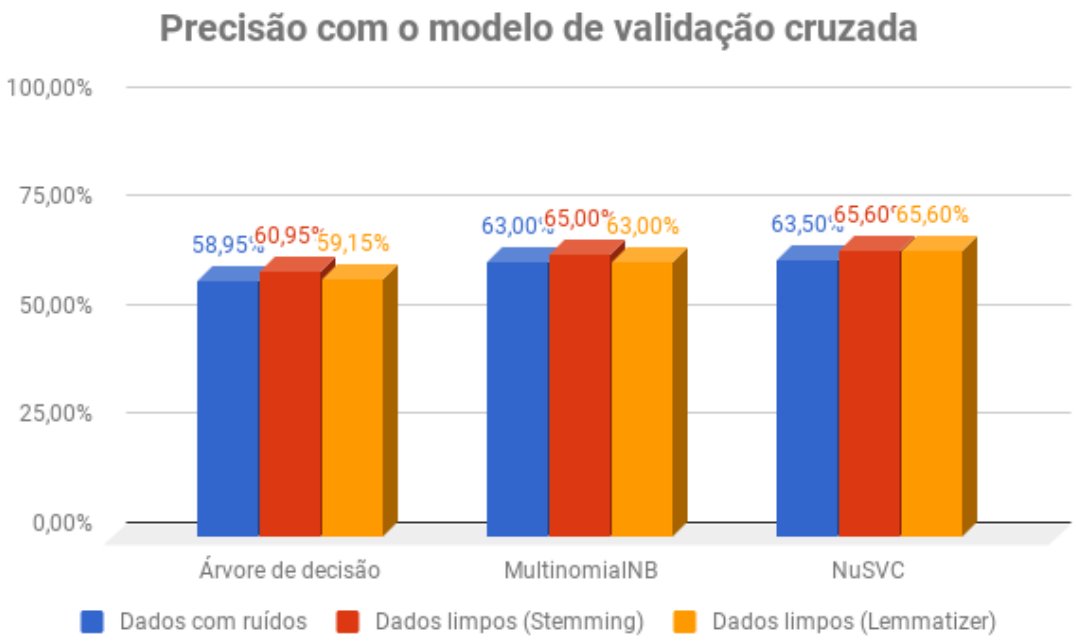
7.3 Discussão dos resultados

A partir dos resultados individuais de cada algoritmo usado é possível compará-los em relação ao seu desempenho. Para os três algoritmos os resultados das métricas obtidas com o modelo de CV foi sempre o mesmo, ou seja, o resultado para precisão foi o mesmo para o *recall* e para acurácia. Isto ocorre devido a amostra ser balanceada entre as classes alvo. Dessa forma, para discutir os resultados é necessário observar somente uma métrica do modelo CV, junto com o resultado de precisão obtido pelo próprio classificador.

A Figura 9 apresenta os resultados de precisão do modelo CV para os três algoritmos. O algoritmo com melhor desempenho segundo esta métrica é o NuSVC, seguido do algoritmo MultinomialNB. Além de apresentar o melhor desempenho, o NuSVC apresenta a maior diferença entre a classificação com dados limpos em relação aos dados com ruídos, além de apresentar o desempenho igual para as duas diferentes técnicas de pré-processamento utilizadas. Entretanto, a árvore de decisão apresenta o pior desempenho geral, mas a relação do desempenho da classificação com dados limpos sobre dados com ruídos é similar ao desempenho do MultinomialNB. Com isso, quanto melhor foi o desempenho do algoritmo, maior é a superioridade da classificação com dados limpos.

Para a precisão obtida pelo próprio classificador, os resultados são apresentados na

Figura 9 – Precisão com CV para os três algoritmos usados



Fonte: Elaborado pelo autor

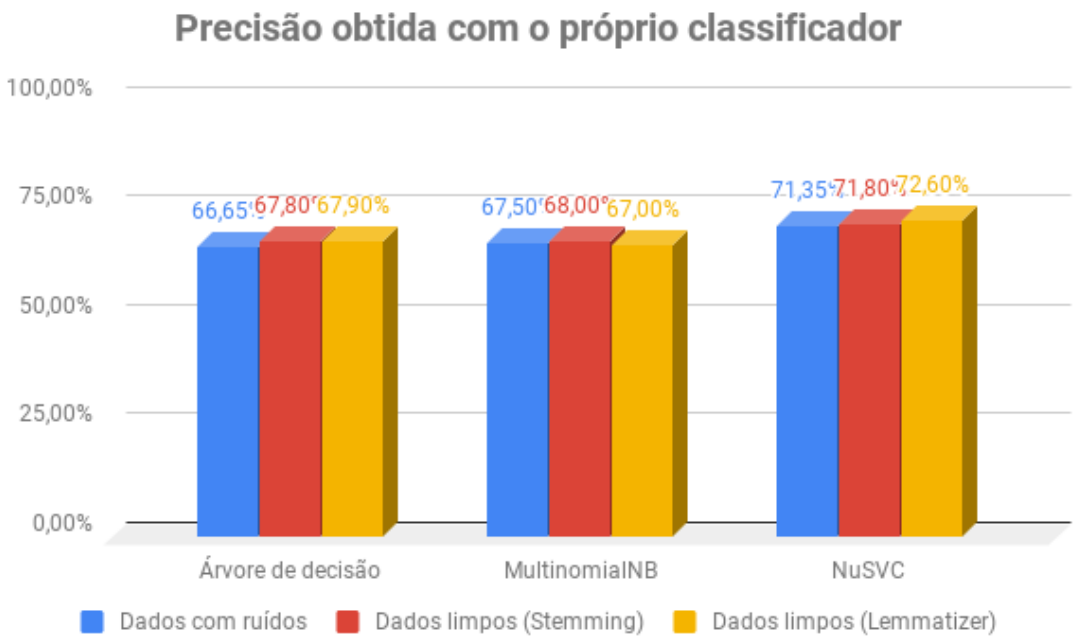
Figura 10. Para este resultado a superioridade da classificação com os dados limpos diminuiu, mais especificamente o desempenho dos dados limpos com a técnica *stemming*. O melhor algoritmo segundo esta métrica foi também o NuSVC, apresentando também a maior diferencia entre as classificações, especialmente para a superioridade da classificação com dados limpos usando *lemmatizer* sobre a classificação com dados com ruídos.

Assim, concluímos que de fato o pré-processamento contribui para o melhor desempenho do classificador, apesar desta diferença ainda ser pequena. E o melhor desempenho é utilizando o algoritmo NuSVC e que a técnica *stemming* apresenta os melhores resultados para os obtidos com o modelo CV, enquanto que para a precisão do próprio classificador o *lemmatizer* foi melhor.

Vale ressaltar que segundo Tan, Steinbach e Kumar (2009) o modelo CV cobre efetivamente o conjunto inteiro de dados, assim o resultado para essa métrica é sempre utilizado a mesma amostra de teste para todos os casos quando é utilizado a mesma base dados, com isso os resultados ficam o mais próximo do resultado real comparado a métrica do próprio classificador no qual utiliza de 20% dos dados para testes, e que na classificação seguinte a amostra de 20% pode ser diferente mesmo utilizando a mesma base de dados.

Dessa forma, de acordo com os resultados das métricas do modelo CV quanto melhor o classificador, melhor o desempenho com dados limpos, assim a utilização do pré-processamento

Figura 10 – Precisão dos três algoritmos usados



Fonte: Elaborado pelo autor

antes da classificação é uma etapa importante principalmente na utilização de métodos mais sofisticados, e esta implementação pode ser ainda mais importante quando métodos mais robustos forem implementados na ferramenta UUX-Posts.

8 CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo são apresentadas as considerações finais deste trabalho, com ênfase nas dificuldades encontradas, contribuições e trabalhos futuros. Esse trabalho foi fomentado pela necessidade da classificação automática na ferramenta UUX-Posts utilizando métodos mais sofisticados comparado a busca booleana, além de técnicas de pré-processamento para proporcionar melhores resultados. Assim, o objetivo consistiu em propor funcionalidades para a ferramenta UUX-Post, a fim de suprir essa necessidade. Apesar da integração das funcionalidades desenvolvidas não ser do escopo deste trabalho.

8.1 Dificuldades

A principal dificuldade encontrada durante a realização deste trabalho foi o apoio de trabalhos existentes para a efetivação deste em relação ao Processamento de Linguagem Natural para textos em português. Apesar da maioria das técnicas de pré-processamento serem simples, outras como *lemmatizer* custou maiores esforços para desenvolver, além de outras técnicas que não foram viáveis de implementar, como a técnica *thesaurus*. Com isso, boa parte do tempo destinado a este trabalho foi dedicado a explorar demais pesquisas, tentando colher soluções confiáveis que poderiam ser reutilizadas para a ferramenta UUX-Posts, tendo em vista que este trabalho propõe várias técnicas, inviáveis de serem todas implementadas sem uso de soluções de terceiros, para o tempo destinado. Devido isso a integração das funcionalidades implementadas na ferramenta não foi incluída neste trabalho.

8.2 Contribuições do trabalho

As contribuições deste trabalho consistiram na apresentação de um estudo de funcionalidades de pré-processamento e mineração de dados em Postagens Relacionadas ao uso de sistemas, contribuindo para o domínio de avaliação textual, bem como a possibilidade de implementar as funcionalidades desenvolvidas para a ferramenta UUX-Posts. Funcionalidades de pré-processamento e mineração de dados para classificação de PRUs, utilizando a metodologia MALTU definida em Mendes (2015).

Com essas contribuições e a partir da integração na ferramenta, será possível classificar PRUs de forma automática pela ferramenta em sua nova versão, onde atualmente só fornece classificação de forma manual. E em sua versão 1, a classificação automática não

apresentavam bons resultados, por utilizar ainda o modelo de busca booleana e por não utilizar o pré-processamento de dados. Com a funcionalidade de pré-processamento implementada, facilita a utilização de demais algoritmos para classificação de PRUs na ferramenta UUX-Posts, possibilitando assim o uso de algoritmos mais sofisticados como, por exemplo, algoritmos de rede neurais, para melhorar cada vez mais os resultados da classificação e melhorar assim a confiabilidade da ferramenta.

8.3 Trabalhos futuros

Como trabalhos futuros, pretende-se utilizar uma base de dados mais consistente e com mais postagens (de sistemas sociais) para treinar o modelo de classificação, tendo em vista que com o modelo de classificação mais consistente e com mais dados, a tendencia é que os resultados sejam melhores. Os próximos passos também envolve integrar as funcionalidades na ferramenta UUX-Posts, o que pode também necessitar de manutenção nas funcionalidades.

Para evoluir essas funcionalidades pretende-se como trabalhos futuros implementar novas técnicas e melhorar as existentes. Por exemplo: melhorar a base de dados de *stopwords*, no sentido de adicionar mais dados; melhorar o desempenho da técnica *lemmatizer*, em relação tanto a base de dados quando a otimização de busca; implementar a técnica *thesaurus*; e implementar novos algoritmos de classificação ainda mais sofisticados.

REFERÊNCIAS

AMARAL, D. d.; GOMES, R.; JESUS, R. P. d.; ARAUJO, T. M. d.; GOULART, E. E. Metodologias de teste de software. **Revista de Informática Aplicada**, Santo André, 2010.

AZEVEDO, B. F. T.; BEHAR, P. A.; REATEGUI, E. B. Análise das mensagens de fóruns de discussão atráves de um software para mineração de textos. **Anais do XXI SBIE - XVII WIE**, Aracaju, 2011.

BECKER, K.; TUMITAM, D. Introdução à mineração de opiniões: Conceitos, aplicações e desafios. In: **Anais do 28 Simpósion Brasileiro de Banco de Dados**. UFRGS, 2013. p. 27–52. Disponível em: http://inf.ufrgs.br/~kbecker/lib/exe/fetch.php?media=minicursosbbd_versaosubmetida.pdf. Acesso em: 11 maio 2018.

BICK, E. **The Parsing System "Palavras": Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework**. Aarhus University Press, 2000. ISBN 9788772889108. Disponível em: https://books.google.com.br/books?id=UJTHcr8OzaIC. Acesso em: 13 mar. 2018.

BIRD, S. Nltk: The natural language toolkit. In: **Proceedings of the COLING/ACL on Interactive Presentation Sessions**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006. (COLING-ACL '06), p. 69–72. Disponível em: https://doi.org/10.3115/1225403.1225421. Acesso em: 13 abr. 2018.

CAMARGO, B. V.; JUSTO, A. M. Iramuteq: um software gratuito para anã¡lisede dados textuais. **Temas em Psicologia**, scielopepsic, v. 21, p. 513 – 518, 12 2013. ISSN 1413-389X. Disponível em: http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S1413-389X2013000200016&nrm=iso. Acesso em: 26 maio 2018.

COPPIN, B. **Intelingência Artificial**. Rio de Janeiro: LTC, 2013. ISBN 978-85-216-1729-7.

DA SILVA, L. A.; PERES, S. M.; BOSCARIOLI, C. **Introdução a Mineração de Dados com aplicações em R**. 1. ed. Brasil: Elsevier, 2017. ISBN 9788535284478.

DIAS, M. M. Parâmetros na escolha de técnicas e ferramentas de mineração de dados. Maringá - PR, Brasil., 2002.

DURAN, M. S.; AVANçO, L.; NUNES, M. G. V. A normalizer for ugc in brazilian portuguese. In: **Workshop on Noisy User-generated Text**. BDPI, 2015. Disponível em: http://www.producao.usp.br/bitstream/handle/BDPI/49416/2712653.pdf?sequence=1&isAllowed=y. Acesso em: 21 abr. 2018.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson, 2011. Tradução Daniel Vieira; revisão técnica Enzo Seraphim e Thatyana de Faria Piola Seraphim. ISBN 978-85-7936085-5.

FAGUNDES, C.; VIEIRA, R. Grupos gramaticais e sintáticos em categorização automática com support vector machines. São Leopoldo - RS, 2004.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI Magazine**, v. 17, p. 37–54, 1996.

FOSTER, J.; ÇETINOGLU, Ö.; WAGNER, J.; ROUX, J. L.; HOGAN, S.; NIVRE, J.; HOGAN, D.; GENABITH, J. V. #hardtoparse: POS Tagging and Parsing the Twitterverse. In: **AAAI 2011 Workshop On Analyzing Microtext**. United States: HAL, 2011. p. 20–25. Disponível em: https://hal.archives-ouvertes.fr/hal-00702445. Acesso em: 14 mar. 2018.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techniques**. 2. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006. ISBN 1558609016.

JANSSENS, O.; SLEMBROUCK, M.; VERSTOCKT, S.; HOECKE, S. V.; WALLE, R. Van de. Real-time emotion classification of tweets. In: **Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining**. New York, NY, USA: ACM, 2013. (ASONAM '13), p. 1430–1431. ISBN 978-1-4503-2240-9. Disponível em: http://doi.acm.org/10.1145/2492517.2492577. Acesso em: 19 maio 2018.

JUNIOR, M. S.; MERSCHMANN, L. H. C. A methodology to handle social media posts in brazilian portuguese for text mining applications. In: **Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web**. New York, NY, USA: ACM, 2016. (Webmedia '16), p. 239–246. ISBN 978-1-4503-4512-5. Disponível em: http://doi.acm.org/10.1145/2976796.2976845. Acesso em: 18 abr. 2018.

LI, A.; CHEN, Y. Pre-processing analysis for chinese text sentiment analysis. In: **Proceedings of the 2017 2Nd International Conference on Communication and Information Systems**. New York, NY, USA: ACM, 2017. (ICCIS 2017), p. 318–323. ISBN 978-1-4503-5348-9. Disponível em: http://doi.acm.org/10.1145/3158233.3158235. Acesso em: 17 abr. 2018.

LIM, K. W.; BUNTINE, W. Twitter opinion topic model: Extracting product opinions from tweets by leveraging hashtags and sentiment lexicon. In: **Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management**. New York, NY, USA: ACM, 2014. (CIKM '14), p. 1319–1328. ISBN 978-1-4503-2598-1. Disponível em: http://doi.acm.org/10.1145/2661829.2662005. Acesso em: 24 abr. 2018.

LIU, F.; WENG, F.; JIANG, X. A broad-coverage normalization system for social media language. In: **Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012. (ACL '12), p. 1035–1044. Disponível em: http://dl.acm.org/citation.cfm?id=2390524.2390662. Acesso em: 14 mar. 2018.

MAGDY, W.; ELDESOUKY, M. Classstrength: A multilingual tool for tweets classification. I n: **P roceedings o f t he 2 017 I EEE/ACM I nternational Conference on Advances in Social Networks Analysis and Mining 2017**. New York, NY, USA: ACM, 2017. (ASONAM '17), p. 593–596. ISBN 978-1-4503-4993-2. Disponível em: http://doi.acm.org/10.1145/3110025.3110162. Acesso em: 17 abr. 2018.

MAIMON, O.; ROKACH, L. **Introduction to Knowledge Discovery and Data Mining**. New York: Springer, 2010. v. 2. ISBN 978-0-387-09822-7.

MENDES, M. S. **MALTU – Um modelo para avaliação da interação em sistemas sociais a partir da linguagem textual do usuário.** 2015. 201 f. Tese (Doutorado em Ciência da Computação) — Departaemnto de Computação, Programa de Pós-Graduação em Ciência da Computação, Universidade Federal do Ceará, Fortaleza, 2015.

MENDES, M. S.; FURTADO, E. S. Uux-posts: a tool for extracting and classifying postings related to the use of a system. **Proceedings of the 8th Latin American Conference on Human-Computer Interaction**, Antigua Guatemala, 2017.

NING, Z.; YI, J. S.; PALAKAL, M. J.; MCDANIEL, A. Oncoviz: A user-centric mining and visualization tool for cancer-related literature. In: **Proceedings of the 2010 ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2010. (SAC '10), p. 1827–1828. ISBN 978-1-60558-639-7. Disponível em: http://doi.acm.org/10.1145/1774088.1774475. Acesso em: 16 mar. 2018.

NLTK. **nltk.corpus package**. 2018. Disponível em: https://www.nltk.org/api/nltk.corpus.html. Acesso em: 13 set. 2018.

NLTK. **nltk.stem package**. 2018. Disponível em: https://www.nltk.org/api/nltk.stem.html. Acesso em: 13 set. 2018.

NLTK. **nltk.tokenize package**. 2018. Disponível em: https://www.nltk.org/api/nltk.tokenize.html. Acesso em: 13 set. 2018.

OLSON, D. L.; DELEN, D. **Advanced Data Mining Techniques**. 1. ed. Heidelberg: Springer, 2008. ISBN 978-3-540-76916-3.

Pandas. **Pandas**. 2018. Disponível em: https://pandas.pydata.org/. Acesso em: 13 set. 2018.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in python. **J. Mach. Learn. Res.**, JMLR.org, v. 12, p. 2825–2830, nov. 2011. ISSN 1532-4435. Disponível em: http://dl.acm.org/citation.cfm?id=1953048.2078195. Acesso em: 6 maio 2018.

Python Software Foundation. **Python**. 2001–2018. Disponível em: https://docs.python.org/3/library/. Acesso em: 12 set. 2018.

RODRIGUES, J.; COSTA, F.; SILVA, J.; BRANCO, A. Automatic syllabification of portuguese. XXX Encontro Anual da Associação Portuguesa de Linguística, Lisboa, 2014.

SOARES, M. V. B.; PRATI, R. C.; MONARD, M. C. **PreTexT: A Reestrutura¸c~ao da Ferramenta de Pre-Processamento de Textos**. São Carlos, 2008.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introdução ao Data Mining-Mineração de Dados**. Rio de Janeiro: Editora Ciência Moderna, 2009. ISBN 978-85-7393-761-9.

TEXTBLOB. **TextBlob: Processamento simplificado de texto**. 2018. Disponível em: https://textblob.readthedocs.io. Acesso em: 12 set. 2018.

TPT. **Text Preprocessing Tool**. 2013. Disponível em: http://sites.labic.icmc.usp.br/tpt/README.TXT. Acesso em: 27 maio 2018.

TRIGUERO, I.; GONZALEZ, S.; MOYANO, J. M.; GARCíA, S.; ALCALá-FDEZ, J.; LUENGO, J.; FERNANDEZ, A.; JESúS, M. J. del; SANCHEZ, L.; HERRERA, F. Keel 3.0: An open source software for multi-stage analysis in data mining. São Leopoldo - RS, 2017.

UNITEX-PB. **Recursos Lexicais**. 2018. Disponível em: http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/dicionarios.html. Acesso em: 23 set. 2018.

VILARES, D.; ALONSO, M. A.; GóMEZ-RODRÍGUEZ, C. Supervised polarity classification of spanish tweets based on linguistic knowledge. In: **Proceedings of the 2013 ACM Symposium on Document Engineering**. New York, NY, USA: ACM, 2013. (DocEng '13), p. 169–172. ISBN 978-1-4503-1789-4. Disponível em: http://doi.acm.org/10.1145/2494266.2494300. Acesso em: 10 abr. 2018.

VU, T.; PEREZ, V. Interest mining from user tweets. In: **Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management**. New York, NY, USA: ACM, 2013. (CIKM '13), p. 1869–1872. ISBN 978-1-4503-2263-8. Disponível em: http://doi.acm.org/10.1145/2505515.2507883. Acesso em: 24 abr. 2018.

WILLIAMS, G.; MAHMOUD, A. Analyzing, classifying, and interpreting emotions in software users' tweets. In: **Proceedings of the 2Nd International Workshop on Emotion Awareness in Software Engineering**. Piscataway, NJ, USA: IEEE Press, 2017. (SEmotion '17), p. 2–7. ISBN 978-1-5386-2793-8. Disponível em: https://doi.org/10.1109/SEmotion.2017...1. Acesso em: 12 abr. 2018.

WITTEN, I. H.; FRANK, E.; HALL, M. A.; PAL, C. J. **Data Mining: Practical Machine Learning Tools and Techniques**. 4. ed. United States: Elsevier, 2016.

WREPLACE. **wReplace**. 2013. Disponível em: http://www.sharktime.com/us_wReplace.html. Acesso em: 27 maio 2018.