Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Abordagem Unificada para Análise de Sentimento de Tweets com Domínio Específico

Patrícia Lustosa Ventura Ribeiro

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientador
Prof. Dr. Li Weigang

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Programa de Pós-graduação em Informática

Coordenadora: Prof.ª Dr.ª Alba C. M. A. Melo

Banca examinadora composta por:

Prof. Dr. Li Weigang (Orientador) — CIC/UnB
Prof. Dr. André Carlos Ponce de Leon F. de Carvalho — ICMC/USP
Prof.ª Dr.ª Maria Emília M. T. Walter — CIC/UnB
Prof. Dr. Camilo Chang Dórea — CIC/UnB

CIP — Catalogação Internacional na Publicação

Lustosa Ventura Ribeiro, Patrícia.

Uma Abordagem Unificada para Análise de Sentimento de Tweets com Domínio Específico / Patrícia Lustosa Ventura Ribeiro. Brasília : UnB, 2015.
110 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2015.

1. Análise de Sentimentos, 2. Léxicos de Sentimentos, 3. *Twitter*, 4. Análise de Sentimento de *Tweets*, 5. Processamento de Linguagem Natural

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Abordagem Unificada para Análise de Sentimento de Tweets com Domínio Específico

Patrícia Lustosa Ventura Ribeiro

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof. Dr. Li Weigang (Orientador)
CIC/UnB

Prof. Dr. André Carlos Ponce de Leon F. de Carvalho
ICMC/USP

Prof.ª Dr.ª Maria Emília M. T. Walter
CIC/UnB

Prof. Dr. Camilo Chang Dórea
CIC/UnB

Prof.ª Dr.ª Alba C. M. A. Melo
Coordenadora do Programa de Pós-graduação em Informática

Brasília, 24 de abril de 2015

Dedicatória

À minha família e ao meu futuro marido.

Agradecimentos

A Deus, pois com Ele tudo é possível.

A toda minha família. À minha mãe, por ser meu porto seguro e por ter me ensinado desde cedo a importância dos estudos para conquistar uma vida melhor. Ao meu pai, por sempre trabalhar e se dedicar muito para nos proporcionar uma vida confortável. À minha madrinha Duca, por sempre estar ao meu lado, como uma mãe. À minha irmã Juliana que, mesmo morando tão longe, me ajudou bastante durante esta pesquisa, corrigindo meus textos e me ajudando a rotular o sentimento dos *tweets*. Aos meus irmãos Joel e Marília, pela simples presença em minha vida. Às minhas sobrinhas Alice, Bruna e Júlia, por colorirem minha vida com risos e brincadeiras.

Ao meu noivo Thiago Galvão Cavalcanti, por estar ao meu lado durante todo o mestrado, por todo incentivo, companheirismo e cumplicidade, além da ajuda com a revisão dos textos. Sua presença tornou minha vida mais doce e completa.

Ao meu orientador Li Weigang, por acreditar na minha capacidade de pesquisa, buscando sempre as melhores oportunidades para o meu desenvolvimento, pela ajuda e pela paciência que teve comigo durante o desenvolvimento deste projeto.

Aos demais professores e funcionários do programa de pós-graduação em Informática da Universidade de Brasília e aos Drs. Viorel Milea e Tiancheng Li pela participação nesta pesquisa.

Resumo

Twitter é uma rede social *online* que permite que os usuários enviem e leiam mensagens curtas chamadas *tweets*. Em dezembro de 2014, o *Twitter* possuia mais de 500 milhões de usuários, dos quais mais de 284 milhões são usuários ativos, gerando aproximadamente 500 milhões de *tweets* todos os dias. O uso massivo de redes sociais *online* está atraindo atenção da academia e de empresas para o estudo da análise de sentimento, especialmente o *Twitter*, através da Análise de Sentimento de *Tweets* (AST). Essa análise proporciona *insights* sobre a opinião do público sobre vários tópicos, como política, notícias e produtos. Para executar AST eficientemente em um domínio específico, uma abordagem com uma ferramenta unificada é proposta. Essa abordagem possui quatro passos: coletar *tweets* relacionados ao domínio, identificar e excluir *tweets* que são *spam*, construir um léxico de sentimento específico para o domínio e analisar o sentimento dos *tweets* válidos. O léxico é um elemento chave que deve ser específico para domínio para poder incorporar expressões cujo sentimento varia de um domínio para outro. A ferramenta de AST proposta foi implementada e testada nos domínios '*iPhone 6*' e 'cigarros eletrônicos' e obteve resultados convincentes nas quatro etapas, mostrando a superioridade de uma ferramenta de AST específica para domínio em relação a uma genérica.

Palavras-chave: Análise de Sentimentos, Léxicos de Sentimentos, *Twitter*, Análise de Sentimento de *Tweets*, Processamento de Linguagem Natural

Abstract

Twitter is an online social networking (OSN) service that enables users to send and read short messages called "tweets". As of December 2014, Twitter has more than 500 million users, out of which more than 284 million are active users and about 500 million tweets are posted every day. The massive use of online social networks is attracting great attention to the study of sentiment analysis, specially Tweet Sentiment Analysis (TSA). This analysis provides insights into the opinion of the public on various topics, from political affairs, hot news to commercial products. In order to execute efficient TSA on a particular topic or domain, an approach with a unified tool is proposed. This approach consists of four steps: collecting tweets related to that topic, identifying and excluding spam tweets, building a domain-specific sentiment lexicon and analyzing the sentiment of tweets. Among them, the lexicon is a key element that is domain-specific as well as incorporates expressions whose sentiment varies from one domain to another. The proposed TSA tool is tested on the 'iPhone 6' and 'electronic cigarettes' domains which obtains convincing results in all of the four phases, showing the superiority of the domain-specific TSA tool over a generic one.

Keywords: Sentiment Analysis, Sentiment Lexicons, Twitter, Tweet Sentiment Analysis, Natural Language Processing

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Redes sociais *online* são serviços Web que permitem que os indivíduos construam perfis públicos ou semipúblicos dentro de uma comunidade virtual na Internet, articulem uma lista de outros usuários com os quais compartilham conexões e visualizem e percorram as suas listas de conexões e outras listas feitas por outros usuários no sistema [9]. Um exemplo de rede social *online* é o *Twitter*.

O *Twitter* é um serviço de microblog popular onde os usuários postam mensagens de status chamados *tweets*. Segundo o site do *Twitter* [1], essa rede social possui mais de 500 milhões de usuários, dos quais 284 milhões são usuários ativos na rede. Cerca de 500 milhões de *tweets* são publicados diariamente.

Essa quantidade enorme de dados pode ser usada para fornecer *insights* da opinião do público sobre vários tópicos diferentes, desde política até filmes e produtos. Para poder extrair essas informações dos *tweets*, as aplicações precisam realizar a análise de sentimento dessas mensagens, que consiste em categorizar o sentimento de um texto como positivo ou negativo de acordo com o seu conteúdo.

Como o volume de dados é muito grande, é inviável processar esses *tweets* manualmente para deles extrair as informações necessárias. Por isso, muitas organizações estão aderindo a formas automáticas de extração das referidas informações. Sem dúvidas, trata-se de um ambiente propício a pesquisas e, como tal, vem sendo explorado tanto pela indústria quanto pela academia, principalmente nos últimos anos.

[1] http://www.twitter.com

1.1.1 *Twitter*

Como já foi visto, *Twitter* é uma rede social e as mensagens de status postadas pelos usuários são chamadas de *tweets*. Essas mensagens são textos curtos, podendo ter no máximo 140 caracteres. Os *tweets* podem conter *emoticons*, *hashtags*, menções e URLs.

Emoticon é uma sequência de caracteres tipográficos que traduz o estado psicológico ou emotivo de quem os emprega. Como exemplos de *emoticons*, pode-se citar :(, =D e ;). *Emoticons* são bastante usados no *Twitter* e são importantes para a definição do sentimento dos *tweets*.

Hashtags são palavras-chave ou termos associados a uma informação ou tópico que se deseja indexar de forma explícita. *Hashtags* são compostas por uma palavra-chave antecedida do símbolo cerquilha (#). Elas viram hiperlinks dentro da rede, sendo indexáveis pelos mecanismos de busca. Como exemplos de *hashtags*, pode-se citar #iphone6, #happy e #brasil.

Uma menção (ou *mention*) é qualquer *tweet* que contém um nome de usuário precedido de arroba (@): @nomedeusuario. Essa função serve para referenciar outro usuário da rede social.

São funções do *Twitter* resposta, *retweet* e favorito. Resposta (ou *reply*) é uma função que permite a um usuário responder ao *tweet* de outro usuário. *Retweet* é uma função do *Twitter* que consiste em replicar uma determinada mensagem de outro usuário para a sua lista de seguidores. O usuário também tem a opção de clicar no botão de Favorito, para indicar que gostou de um *tweet*.

Usuários podem se inscrever para visualizar os *tweets* de outro usuário - isso é chamado de seguir ou *follow*. *Followers* ou seguidores de um usuário A é o conjunto de todos os usuários que seguem o usuário A. Já *followees* de um usuário A é o conjunto de todos os usuários que A segue.

Trending topics ou assuntos do momento são uma lista em tempo real das *hashtags* ou nomes de usuários mais publicados no *Twitter* pelo mundo todo. Os *trending topics* estão disponíveis na página inicial do *Twitter* e podem ser usado tanto de maneira global quanto regionalizada.

1.2 Motivação

Análise de sentimento de textos é um tópico de processamento de linguagem natural cujo objetivo é identificar o sentimento geral de um dado texto. A análise de sentimento de textos tem sido reconhecida e bastante estudada dentro do contexto de *tweets*. Através da análise de sentimento dessas mensagens é possível avaliar a reputação de uma empresa [20], monitorar a opinião pública [46] e explicar e prever o mercado de ações [7] [8].

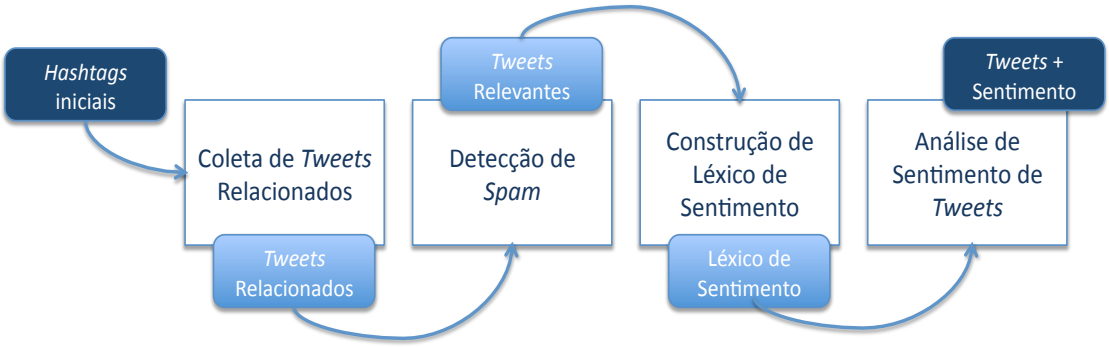

Figura 1.1: Visão Global da Arquitetura do Sistema

A maior parte dos trabalhos existentes nessa área usa um léxico de sentimentos, que é uma lista com palavras ou frases com sentimento associado [38]. O desempenho do algoritmo é intimamente relacionado à qualidade do léxico. Para alcançar bons resultados na avaliação de *tweets* de um domínio particular, é interessante criar um léxico que seja específico para essas mensagens e relacionado a esse domínio, para poder incorporar expressões cujo sentimento varia de um domínio para o outro.

Como não foi encontrado na literatura uma abordagem pra construir léxicos específicos para domínio e *tweets* de forma automática e não-supervisionada, esse trabalho propõe essa abordagem. Além disso, para facilitar o uso da ferramenta, esse trabalho não se limita à criação do léxico, englobando também as demais fases necessárias e suficientes para realizar a análise de sentimento de *tweets* de domínio específico.

1.3 Definição do Problema

O intuito desse trabalho é apresentar uma nova abordagem e construir uma ferramenta unificada para executar a análise de sentimento de *tweets* em um domínio particular. Assim, o usuário fornece como entrada um conjunto inicial de termos e/ou *hashtags* sobre o domínio que ele quer analisar e a ferramenta retorna um conjunto de *tweets* com o seu sentimento associado. Para esse trabalho, foi definido que o sentimento de um *tweet* pode ser positivo ou negativo.

Para realizar todo o trabalho necessário, essa ferramenta foi dividida em quatro etapas. A primeira etapa lida com o problema de coletar *tweets* relacionados ao domínio. A segunda etapa consiste em excluir os *tweets* que são *spam*. Já a terceira etapa é responsável por construir um léxico de sentimentos para *tweets* de domínio específico. Finalmente, a última etapa realiza a análise de sentimento dessas mensagens. Uma visão global da arquitetura do sistema pode ser vista na Figura 1.1.

1.4 Metodologia

Esta seção apresenta a metodologia do presente trabalho, explicando no que consiste cada uma das quatro etapas da ferramenta proposta.

1.4.1 Etapa 1: Coleta de *Tweets* Relacionados

A primeira etapa consiste na coleta de *tweets* relacionados. Essa atividade poderia ser feita manualmente pelo usuário. Para isso, ele deveria pesquisar sobre o domínio no *Twitter* e verificar quais *hashtags* e termos são usados para se referir a esse domínio.

Para evitar que o usuário tenha todo esse trabalho, foi proposto um algoritmo de expansão de *hashtags*. Esse algoritmo tem como objetivo a expansão de um conjunto inicial de *hashtags* e/ou termos passados pelo usuário em um conjunto maior e mais completo de *hashtags*. Com isso, pode ser feita uma busca na *API* do *Twitter* [2] por *tweets* que possuem essa *hashtag* e assim coletar os *tweets* relacionados. Com esse algoritmo, a ferramenta é capaz de coletar as mensagens necessárias sem requerer trabalho manual.

1.4.2 Etapa 2: Detecção de *Spam*

O objetivo da segunda etapa é detectar e excluir os *tweets* que são *spam* dentre os coletados na etapa anterior de forma a manter apenas os *tweets* relevantes. Dentre os *spams*, encontram-se as propagandas, os pedidos para serem seguidos, os *tweets* gerados automaticamente e os *tweets* com *links* maliciosos. A quantidade de *spam* entre os *tweets* é muito grande, principalmente nos *trending topics*. Excluir esses *spams* é importante para que não sejam usados para a construção do léxico, já que não representam uma opinião real de usuário.

1.4.3 Etapa 3: Construção de Léxicos de Sentimento

O objetivo dessa etapa é construir um léxico de sentimento. Esses léxicos são geralmente construídos tendo como base textos formais, que atendem às normas gramaticais. No entanto, os textos normalmente encontrados no *Twitter* caracterizam-se por serem textos curtos e informais e conterem muitas abreviações e gírias. Além disso, os *tweets* possuem peculiaridades no seu texto, como *emoticons*, *hashtags* e menções. Por isso, um léxico de sentimento tradicional não é adequado.

Para resolver esse problema, alguns léxicos específicos para *tweets* foram construídos. Eles aumentam a precisão da análise de sentimentos de *tweets* em relação aos léxicos de uso geral. Porém, apesar de serem específicos para *Twitter*, eles não são específicos para

[2] https://dev.twitter.com

nenhum domínio (ex.: finanças ou restaurantes). Esse fato impacta em sua precisão, pois muitas expressões positivas e negativas dependem do domínio explorado.

Por isso, a proposta é construir um léxico de sentimento para *tweets* de domínio específico. Com isso, busca-se melhorar o desempenho da ferramenta, já que esse léxico é capaz de capturar as especificidades do *Twitter*, bem como as expressões dependentes de domínio.

1.4.4 Etapa 4: Análise de Sentimento de *Tweets*

A última etapa da ferramenta tem como objetivo realizar a análise de sentimento dos *tweets* selecionados. Para isso, foi desenvolvido um algoritmo que, utlizando o léxico criado na etapa anterior, retorna o sentimento dos *tweets*. Esse algoritmo funciona de maneira não-supervisionada e não precisa de dados rotulados de entrada. Isso facilita a aplicação da ferramenta para *tweets* de outros domínios.

1.5 Objetivo Geral

Construir uma abordagem e implementar uma ferramenta unificada que lide com todas as etapas necessárias e suficientes para realizar a análise de sentimento de *tweets* de domínio específico, visando obter resultados superiores aos alcançados por ferramentas para *tweets* genéricas, o que contribuirá para facilitar o uso da análise de sentimentos por usuários finais.

1.5.1 Objetivo Específicos

São objetivos específicos do trabalho proposto:

- Desenvolver de um algoritmo de expansão de *hashtags*;
- Desenvolver de um algoritmo de detecção de *spam*;
- Desenvolver de um algoritmo de construção de léxicos de sentimento para *tweets* de domínio específico;
- Desenvolver de um algoritmo de análise de sentimentos não-supervisionado;
- Validar da ferramenta usando os domínios '*iPhone 6*' e 'cigarros eletrônicos';
- Criar de léxicos de sentimento para tweets dos domínios '*iPhone 6*' e 'cigarros eletrônicos';
- Comparar do desempenho da análise de sentimento dos *tweets* utilizando o léxico construído, léxicos para *tweets* gerais e léxicos gerais.

1.6 Estrutura do Trabalho

Este documento está dividido nos capítulos apresentados a seguir. O Capítulo 2 apresenta o estado da arte do problema de análise de sentimento de *tweets*, incluindo trabalhos relativos a todas as etapas do problema abordadas na ferramenta. O Capítulo 3 traz a fundamentação teórica necessária ao desenvolvimento da pesquisa. Os quatro capítulos seguintes referem-se às quatro etapas da ferramenta proposta. Em cada um deles, são exibidos o algoritmo proposto para resolver o problema, um estudo de caso desse algoritmo e os trabalhos futuros. O Capítulo 4 apresenta o problema da coleta de *tweets* relacionados. O Capítulo 5 mostra a detecção de *spam*. Já o Capítulo 6 traz a construção do léxico de sentimento. O Capítulo 7 é responsável pela análise de sentimento de *tweets*. O Capítulo 8 apresenta um estudo de caso completo da ferramenta para o domínio cigarros eletrônicos. Por fim, o Capítulo 9 apresenta as conclusões da pesquisa e os trabalhos futuros.

Capítulo 2

Estado da Arte

Neste capítulo, é apresentada a revisão da literatura e o estado da arte relativos às quatro etapas da ferramenta proposta. A Seção 2.1 diz respeito à coleta de *tweets* relacionados e à expansão de *hashtags*. A Seção 2.2 apresenta trabalhos sobre a identificação de *spam* no *Twitter*. A Seção 2.3 mostra diferentes maneiras de construir um léxico de sentimentos. A Seção 2.4 reúne algoritmos de análise de sentimento encontrados na literatura. Por fim, a Seção 2.5 traz comentários a respeito do apresentado no capítulo.

2.1 Coleta de *Tweets* Relacionados

Nesta seção, são discutidos trabalhos relativos ao problema de coletar *tweets* relacionados a um determinado tópico. Esse problema foi reduzido a encontrar *hashtags* relacionadas ao tópico no *Twitter*. Primeiramente, são apresentados dois artigos que têm uma abordagem similar à proposta neste trabalho e focam em expandir *hashtags* do *Twitter*. Em seguida, na Seção 2.1.2, são discutidos trabalhos a respeito do problema de agrupamento de *hashtags*. Esse problema é ligeiramente diferente, mas pode dar ideias sobre como resolver o problema original. A Seção 2.1.3 apresenta o problema de recomendação de *hashtags*. Por fim, na Seção 2.1.4 são apresentados alguns comentários e conclusões sobre os trabalhos apresentados.

2.1.1 Expansão de *Hashtags* do *Twitter*

Algoritmos de expansão de *hashtags* do *Twitter* têm o objetivo de resolver o problema de encontrar *hashtags* relacionadas a partir de um conjunto semente de *hashtags* e termos.

Em [47], os autores apresentaram um método de detecção de eventos no *Twitter* baseado em agrupamento de *hashtags* e introduziram uma técnica que usa similaridade semântica entre *hashtags*. Eles usaram um algoritmo de agrupamento aglomerativo com

pesos setados com os valores TF-IDF (*Term Frequency–Inverse Document Frequency)*). Para gerar os vetores dos *tweets*, foram usados quatro métodos: palavras em *tweets* sem expansão semântica, palavras com expansão semântica, *hashtags* sem expansão semântica e *hashtags* com expansão semântica. Os melhores resultados foram obtidos com a expansão léxico-semântica, visto que essa expansão tolera erros de digitação e consolida os *tweets* com significado parecido.

Carter *et al.* [11] propuseram um método para encontrar *hashtags* relacionadas em uma linguagem de destino, dada uma *hashtag* na linguagem de origem. O algoritmo proposto funciona da maneira explicada a seguir. Primeiro, eles retornam uma lista de *tweets* na linguagem de origem que contêm as *hashtags* semente. Em seguida, eles traduzem os *tweets* e usam TF-IDF para encontrar os termos mais encontrados nessas mensagens. Os termos são usados para fazer uma busca na *API* do *Twitter* por novos *tweets*. As *hashtags* mais usadas nesses *tweets* são retornadas.

2.1.2 Agrupamento de *Hashtags* do *Twitter*

Algoritmos de agrupamento de *hashtags* do *Twitter* têm o propósito de agrupar as *hashtags* de acordo com o tópico, dado um grande conjunto de *tweets* que contêm *hashtags*. As aplicações que usam esses algoritmos normalmente envolvem um conjunto de *tweets* de vários domínios diferentes e que devem ser separados por assunto. Por isso, esse tipo de algoritmo não foi diretamente aplicado neste trabalho.

Em [63], os autores focaram em agrupar *tweets*. A tarefa de agrupamento foi dividida em duas: agrupamento em lote e agrupamento *online* de um fluxo de *tweets*. Na primeira tarefa, o algoritmo explora dados rotulados, permitindo a conversão do problema em agrupamento de *hashtags*. Para resolver esse problema, foi criada uma coleção de documentos não-esparsos agregando todos os *tweets* que compartilham uma mesma *hashtag*. Depois foi usado o algoritmo k-média para agrupar as *hashtags* de acordo com o conteúdo dos documentos.

Muntean *et al.* [44] agruparam *hashtags* em grupos semanticamente interconectados para decifrar o seu significado. O algoritmo de agrupamento utilizado foi o k-média e foram testados diversos valores de k para chegar ao melhor resultado.

O artigo de Jan Kalmeijer [26] analisou *tweets* enviados por 144 membros do parlamento holandês durante um período de três meses. As *hashtags* foram agrupadas com base na co-ocorrência para identificar os tópicos. *Hashtags* que são usadas no mesmo *tweet* foram consideradas mais semanticamente similares do que *hashtags* que não o são. Por essa razão, eles agruparam *hashtags* baseados na relação de co-ocorrência.

Em [55], os autores apresentaram um estudo sobre o agrupamento automático e a classificação de *tweets* em diferentes categorias, inspirado na abordagem feita por serviços

de agregamento de notícias como o Google News. Eles compararam o classificador de Rocchio como método supervisionado e LDA (*Latent Dirichlet Allocation*) e k-média como métodos não-supervisionados. Os resultados alcançados sugerem que agrupamentos produzidos por métodos não-supervisionados tradicionais podem ser incoerentes enquanto utilizar uma abordagem supervisionada pode trazer bons resultados.

Rangrej *et al.* [49] focaram no agrupamento de *tweets* baseado no seu conteúdo. Eles compararam várias técnicas de agrupamento de documentos, incluindo k-média, métodos baseados em decomposição em valores singulares (*SVD - Singular Value Decomposition*) e uma abordagem baseada em grafos, e observaram o desempenho em dados coletados do *Twitter*. Os resultados mostraram que uma abordagem baseada em grafos usando propagação por afinidade apresenta o melhor desempenho.

2.1.3 Recomendação de *Hashtags* do *Twitter*

Algoritmos de recomendação de *hashtags* têm o objetivo de, dado um *tweet* escrito pelo usuário, recomendar *hashtags* relacionadas ao conteúdo do *tweet*.

Em [34], os autores propuseram uma ferramenta de sugestão automática de *hashtags* que retorna uma lista de *hashtags* relevantes a um determinado *tweet*. Para medir a similaridade entre dois *tweets*, eles usaram similaridade de cossenos sobre a representação vetorial *bag-of-words* do conteúdo dos *tweets*. Para determinar a relevância das *hashtags* para um *tweet* individual, eles usaram o modelo *Naive Bayes*.

Kywe *et al.* [28] abordaram o problema de fazer recomendações personalizadas de *hashtags* para o *Twitter*. O objetivo é recomendar uma lista de *hashtags* apropriadas para um determinado usuário que está escrevendo um novo *tweet*. O método proposto considera tanto as preferências do usuário quanto o conteúdo do *tweet*. Dado um usuário e um *tweet*, o algoritmo seleciona os usuários e os *tweets* mais similares e usam essas informações para extrair as *hashtags*. Os experimentos demonstraram que usar recomendação personalizada alcança melhores resultados do que os algoritmos tradicionais, que levam em consideração apenas o conteúdo do *tweet*.

2.1.4 Comentários

Na Tabela 2.1, é apresentada a comparação dos algoritmos apresentados na Seção 2.1. A coluna foco determina se o foco do algoritmo é agrupamento, expansão ou recomendação de *hashtags* do *Twitter*.

Examinando os artigos apresentados, pode-se constatar que a maior parte dos trabalhos focam em agrupamento de *tweets* ou *hashtags*. Dos nove artigos, seis utilizam algum algoritmo de agrupamento, tais como k-média e agrupamento aglomerativo. Como o pro-

blema que se quer resolver no presente trabalho é diferente dos encontrados na literatura, não serão usados algoritmos de agrupamento.

Tabela 2.1: Comparação dos Algoritmos Relacionados a *Hashtags*

Ano	Referência	Foco	Método
2014	[26]	Agrupamento	Co-ocorrência
2013	[63]	Agrupamento	k-média
2012	[47]	Expansão	Similaridade semântica e agrupamento aglomerativo
2012	[44]	Agrupamento	k-média
2012	[28]	Recomendação	Similaridade semântica
2011	[11]	Expansão	Tradução e *TF-IDF*
2011	[55]	Agrupamento	Classificador de Rocchio, *LDA* e k-média
2011	[49]	Agrupamento	k-média, *SVD* e abordagem baseada em grafos
2009	[34]	Recomendação	Similaridade de cossenos e *Naive Bayes*

2.2 Detecção de *Spam*

Nesta seção, é explorado o problema de detectar *spam* no *Twitter*. Há dois tipos de algoritmos relacionados a esse problema: os que focam em detectar os usuários que postam *spam* (*spammers*) e os que focam em detectar *tweets* que são *spam*. Existem também os artigos cujo foco principal não é a detecção de *spam*, mas que tratam *spam* como uma etapa de pré-processamento. Essas três abordagens são detalhadas nas próximas seções.

2.2.1 Detecção de *Spammers*

O objetivo desta seção é explorar os algoritmos que identificam *spammers* no *Twitter*, como Benevenuto *et al.* [3]. Eles coletaram um grande conjunto de dados do *Twitter*, que inclui mais de 54 milhões de perfis de usuários. Usando *tweets* relacionados a três *trending topics* famosos de 2009, eles construiram uma grande coleção de usuários rotulados, manualmente classificados como *spammers* ou não-*spammers*. Com isso, eles identificaram um conjunto de características relacionadas ao conteúdo do *tweet* e ao comportamento do usuário que podem ser usadas para detectar *spammers*. Eles usaram essas características como atributos para um algoritmo de aprendizagem de máquina com o objetivo de classificar usuários como *spammers* ou não. Os cinco atributos com maior desempenho para detectar *spammers* foram fração dos *tweets* com URLs, idade da conta do usuário,

número médio de URLs por *tweet*, fração de *followers* por *followees* e fração de *tweets* que o usuário respondeu.

Stringhini *et al.* [60] lidaram com a detecção de *spammers* no *Facebook*, *Twitter* e *MySpace*. Eles criaram perfis *honeypot* nas três redes sociais e analisaram as requisições de amizade e as mensagens enviadas para eles. Com base na análise desse comportamento, eles desenvolveram técnicas para detectar *spammers* em redes sociais e agruparam essas mensagens em grandes campanhas de *spam*.

Klassen [27] analisou o problema de identificar *spammers* e explorou a redução de atributos e o pré-processamento de dados do ponto de vista dos algoritmos de detecção de *spam*. Vários algoritmos de aprendizagem de máquina foram usados, como máquinas de vetores de suporte, redes neurais, J4.8 e florestas aleatórias. Os atributos classificados entre os dez melhores em todos os método de seleção foram fração de *followers* por *followees*, fração de *tweets* respondidos e número de vezes que o usuário respondeu.

Chu *et al.* [13] exploraram uma abordagem de detecção coletiva para capturar campanhas de *spam* com múltiplas contas de usuário. Eles usaram um sistema de classificação baseado em aprendizagem de máquina e aplicaram várias características combinando conteúdo e comportamento para classificar as campanhas de *spam*.

Em [67], os autores examinaram *spams* que usavam a *hashtag* #robotpickuplines e mostraram a existência de diferenças na estrutura da rede entre contas de *spammers* e usuários legítimos. Eles concluiram que as contas de *spammers* não são significativamente mais novas, exibem um número um pouco maior de *retweets* e respostas e têm um número maior de *followers*. Esse resultado contradiz a maioria dos estudos anteriores.

2.2.2 Detecção de *Tweets* que são *Spam*

Há alguns estudos que focam em detectar *tweets* que são *spam*. Nesta seção, são apresentados três artigos que seguem essa linha. Miler *et al.* [36] propuseram um método de identificar *tweets* que são *spam* a partir de um *stream* de *tweets*. Eles trataram *spam* como um problema de detecção de anomalias, enquanto a maioria dos estudos trata como um problema de classificação. A solução proposta inclui dois algoritmos de agrupamento que usam informações do usuário juntamente com algumas características dos *tweets*.

Martinez *et al.* [33] apresentaram uma metodologia baseada em dois aspectos: a detecção de *tweets* que são *spam* de forma isolada e sem informações prévias sobre o usuário e uma análise estatística da linguagem para detectar *spam* em *trending topics*. Eles propuseram um conjunto reduzido de atributos dificilmente manipulados por *spammers* e desenvolveram um sistema de aprendizagem de máquina para analisar as características emergentes de *spam* nas redes sociais.

Um sistema de filtragem *online* de *spam* foi desenvolvido [17] como um componente nas redes sociais *Twitter* e *Facebook* para inspecionar mensagens geradas pelos usuários em tempo real. Foi proposta a reconstrução de *spam* em campanhas ao invés de examinar essas mensagens individualmente. A identificação de campanhas de *spam* já tinha sido feita de modo *offline*, mas a proposta deles foi aplicar essa técnica para detectar *spam online* com pouca sobrecarga.

2.2.3 *Spam* como Pré-processamento

Alguns artigos mencionam a detecção de *spam* apenas como uma etapa de pré-processamento dos seus algoritmos. Em [4], Benhardus afirmou que os *tweets* foram pré-processados para remover URLs, caracteres Unicode, nomes de usuário e pontuação. Um dicionário foi utilizado para filtrar os *tweets* que não continham pelo menos 60% das palavras em inglês. Em seguida, *tweets* foram classificados como *spam* e descartados se pelo menos metade das palavras da mensagem fossem iguais.

O *Twitter* identifica várias classes diferentes de comportamento que é considerado *spam*, como seguir e deixar de seguir usuários de forma agressiva, vender *followers* e fazer menções e respostas não-solicitadas. Focando nesses comportamentos, Almuhimedia *et al.* [1] foram capazes de identificar um limite inferior para a porcentagem de *spam* dentre os *tweets* apagados. Essa abordagem considera apenas uma porção de *tweets*, já que nem todos os *spams* se enquadram nesses comportamentos.

Em [45], os autores usaram o projeto Apache Nutch [1] em combinação com a informação de linguagem reportada pelo *Twitter* para detectar a linguagem do *tweet* e remover os que não estão em inglês. Eles também filtraram *spam* usando um dicionário de termos que são normalmente associados a *spam*. Outras heurísticas incluem remover *tweets* que são respostas e *tweets* que contêm URLs.

2.2.4 Comparação dos Métodos

Nas Tabelas 2.2 e 2.3, são exibidas as características dos *tweets* e das contas de usuário que são usadas por cada artigo apresentado na Seção 2.2. A última coluna de cada tabela (#) significa uma contagem de quantos artigos utilizaram aquela característica. Essas tabelas foram levadas em consideração para construir o algoritmo de detecção de *spam*, detalhado no Capítulo 5.

[1] http://nutch.apache.org/

Tabela 2.2: Comparação dos Artigos Apresentados - Características dos *Tweets*

Característica	[3]	[4]	[13]	[17]	[27]	[33]	[45]	[60]	[67]	#
Número de *Hashtags*	x		x		x			x	x	5
Hashtags com conteúdo não-relacionado					x					1
Hashtags em tópicos diferentes									x	1
Três ou mais *hashtags* de *trending topics*	x									1
Menções de usuário	x		x					x		3
Número de URLs	x			x	x		x	x	x	6
Lista negra de URLs			x			x				2
URL para uma página sobre outro tópico	x					x				2
Número de redirecionamentos de URL			x							1
Entropia		x	x							2
Lista de *spam words*	x		x				x		x	4

2.3 Construção de Léxicos de Sentimento

Nesta seção, é apresentada uma lista dos léxicos existentes na literatura e como eles foram construídos. Esses léxicos estão divididos em genéricos, específicos para domínio e específicos para *tweets*. As próximas seções detalham esses três tipos de léxicos.

2.3.1 Léxicos Genéricos

Léxicos genéricos ou gerais são construídos independentemente de domínio e usam o sentido mais comumente empregado das palavras. Esse léxicos são bastante usados, pois eles podem ser aplicados para vários domínios diferentes. A desvantagem é que eles normalmente não capturam especificidades de cada domínio, o que geralmente leva a baixos valores de *recall*.

Web Lexicon

Em [64], Velikovich *et al.* examinaram a viabilidade de construir grandes léxicos de sentimento a partir da web de forma semi-automática. Eles propuseram um algoritmo de

Tabela 2.3: Comparação dos Artigos Apresentados - Características da Conta de Usuário

Característica	[3]	[13]	[27]	[33]	[36]	[60]	[67]	#
Idade da conta de usuário	x	x	x		x			4
Contagem de *retweets*					x			1
Fração de *followers* por *followees*					x	x		2
Contagem de *followers*			x					1
Média de URLs por *tweet*	x		x			x		3
Média de *tweets* que são resposta			x					1
Número de vezes que o usuário enviou uma resposta			x					1
Número de *tweets*		x						1

propagação de grafo e usaram uma abordagem baseada em corpus.

O corpus utilizado foi constituído de *n-grams* extraídos de quatro bilhões de páginas da web. Apenas 20 milhões de frases candidatas foram selecionadas usando diversas heurísticas, como frequência e informação mútua dos limites das palavras. Um vetor de contexto para cada frase candidata foi construído com base em uma janela de palavras de tamanho seis utilizando todas as menções de cada frase nos 4 bilhões de documentos. Um grafo foi criado com essas frases e o peso das arestas corresponde ao valor da similaridade de cossenos entre as frases.

Um método de propagação de grafos foi usado para calcular o sentimento de cada frase agregando o valor dos melhores caminhos a partir das palavras semente. Com o imenso conjunto de dados, eles conseguiram gerar um léxico com mais de 178.000 entradas. Esse léxico foi comparado com outros três léxicos existentes na literatura e obteve os melhores resultados.

Label Propagation

Rao e Ravichandran [50] propuseram um método de construir um léxico de sentimentos genérico através do uso de um algoritmo de *label propagation* semi-supervisionado. Eles utilizaram dados do WordNet e complementaram com o tesauro do OpenOffice quando WordNet não estava disponível.

Os autores mostraram que o método proposto é independente de linguagem e testaram para as linguagens francês e hindu. Além de *label propagation*, eles também usaram cortes mínimos e cortes mínimos randômicos.

Bing Liu Lexicon

Em [25], o *Bing Liu Lexicon* foi apresentado. Esse léxico utiliza a abordagem baseada em dicionário. Uma lista inicial de 30 palavras relacionadas a sentimentos positivos e negativos foi criada e usada como semente. Usou-se a ideia de que palavras que são sinônimas têm o mesmo sentimento e palavras que são antônimas possuem sentimentos contrários. Assim, se *fast* é uma palavra relacionada a um sentimento positivo, o seu antônimo, *slow*, será uma palavra relacionada a um sentimento negativo.

O dicionário WordNet foi usado para prover uma lista de sinônimos e antônimos. As novas palavras encontradas foram adicionadas à lista de palavras e o processo continuou até que novas palavras não foram mais encontradas. A lista final é o léxico de sentimentos com palavras positivas e negativas.

O léxico resultante possui 6.800 palavras (*unigrams*). Esse léxico vem sendo atualizado por Bing Liu e é disponibilizado na web [2].

MPQA Lexicon

Esse léxico também foi construído usando a abordagem de dicionário e com um trabalho altamente manual. Para construir o léxico, Wilson *et al.* [65] usaram uma lista de sugestões de subjetividade derivada do trabalho de Rillof *et al.* [53]. Sugestões de subjetividade são palavras ou frases que podem ser usadas para expressar subjetividade. Essa lista foi expandida usando um dicionário e um tesauro, coletando os sinônimos e antônimos das palavras que já existiam na lista, de forma iterativa, até o léxico estar completo. Palavras do General Inquirer [58] de Stanford também foram adicionadas à lista.

Todas as palavras foram manualmente marcadas como fortemente ou fracamente subjetivas. O próximo passo foi marcar as palavras como tendo um sentimento positivo ou negativo. As que já possuiam essa classificação, a mesma foi mantida. As que não possuiam, foram marcadas manualmente com uma das seguintes *tags*: positiva, negativa, ambos ou neutra. Ambos significam que em algum casos ela tem um sentimento positivo e em outros casos negativo. Neutro significa que aquela palavra não representa sentimento algum.

O léxico resultante possui 8.000 palavras (*unigrams*) com classificação de subjetividade e classificação de polaridade (ou seja, do sentimento).

[2] http://www.cs.uic.edu/ liub/FBS/sentiment-analysis.html

SentiWordNet

Em [16], Esuli e Sebastiani explicaram a contrução do SentiWordNet, que é um léxico de propósito geral baseado em *synsets*. *Synsets* são conjuntos de sinônimos. Assim, o SentiWordNet não apresenta suas categorizações para termos (*n-grams*), como visto nos outros exemplos. Para o termo *estimable*, por exemplo, existem pelo menos 3 *synsets* no WordNet e cada um deles possui categorizações diferentes no SentiWordNet.

Cada *synset* possui três notas: Positiva, Negativa e Objetiva. A soma das três notas é sempre 1 e cada uma das notas pode assumir um valor entre 0 e 1. Com isso, pode-se detectar tanto SO-polaridade (classificação entre subjetivo e objetivo) quanto PN-polaridade (classificação entre positivo e negativo).

Para a criação do léxico, foi usada a abordagem baseada em dicionário. Primeiro foi escolhido um conjunto semente de *synsets* positivos e negativos que foi expandido usando relações léxicas do WordNet para formar o conjunto de treinamento. Oito classificadores ternários foram usados e diferem um do outro no conjunto de treinamento usado e no tipo de treinamento empregado. Ou seja, os classificadores ternários usaram a abordagem baseada em dicionário de maneira um pouco diferente um do outro para criar conjuntos de treinamento diferentes.

As notas finais de um *synset* foram determinadas pela proporção dos classificadores ternários que atribuíram aquele rótulo ao *synset*. Ou seja, se metade dos classificadores atribuiu Positivo a um *synset* e a outra metada atribuiu Objetivo, então a nota desse *synset* será 0.5 Positivo, 0 Negativo e 0.5 Objetivo.

O léxico final contém 115.000 *synsets*. O SentiWordNet continua sendo atualizado e é distribuído *online* [3].

NRC EmoLex

Nos trabalhos [39] e [40], foi apresentado o EmoLex, um léxico de emoções construído manualmente. A principal diferença entre esse e outros léxicos é que o EmoLex categoriza os termos em relação à emoção que eles expressam, ao contrário dos demais, que caracterizam em positivos ou negativos. As emoções usadas são *joy, sadness, anger, fear, trust, disgust, surprise* e *anticipation*.

Os termos usados foram extraídos dos Macquerie Thesaurus [5], General Inquierer [58] e WordNet Affect Lexicon [59]. A categorização dos termos com relação às emoções foi feita manualmente, através de uma abordagem de *crowdsourcing*, usando a ferramenta *Amazon's Mechanical Turk* [4]. Com isso, várias pessoas responderam questionários construídos para possibilitar a extração das informações necessárias para a categorização dos

[3] http://sentiwordnet.isti.cnr.it/
[4] https://www.mturk.com

termos em relação às emoções. Um pós-processamento foi feito para minimizar os erros das respostas e melhorar a precisão dos resultados.

O léxico final apresenta mais de 24 mil termos categorizados e está disponível *online* [5] .

2.3.2 Léxicos para Domínio Específico

Para capturar as especificidades de cada domínios, vários artigos focam em construir léxicos para domínio específico. Nesta seção, são apresentados alguns deles.

Lu Lexicon

Em [32], os autores propuseram um *framework* para criar um léxico de sentimento dependente de contexto a partir da combinação de diferentes fontes de informação.

Esse problema foi tratado como um problema de otimização. Basicamente, foi usada uma função objetiva a ser otimizada pelos candidatos a entradas do léxico. Para designar essa função objetiva, foram definidas múltiplas fontes de informação, tais como o sentimento da entrada em um léxico de sentimento genérico (2.1), sentimento geral da frase de acordo com um léxico de sentimento genérico (2.2), entradas com um sentimento similar (2.3) e entradas com sentimento oposto (2.4). Todas essas restrições foram consolidadas em uma única função objetiva, apresentada na equação abaixo.

$$\Omega = \frac{\lambda_{prior}}{||I^G||_1} \sum_{j=1}^{n} I_j^G |S_j - G_j| \tag{2.1}$$

$$+ \frac{\lambda_{rating}}{||I^O||_1} \sum_{i=1}^{m} I_i^O \left| \sum_{j=1}^{n} X_{ij} S_j - O_i \right| \tag{2.2}$$

$$+ \frac{\lambda_{sim}}{||A||_1} \sum_{j=1}^{n} \sum_{k=1}^{n} A_{jk} \left| S_j - S_k \right| \tag{2.3}$$

$$+ \frac{\lambda_{oppo}}{||B||_1} \sum_{j=1}^{n} \sum_{k=1}^{n} B_{jk} \left(\left| S_j^+ - S_k^- \right| + \left| S_j^- - S_k^+ \right| \right) \tag{2.4}$$

$$+ \frac{\delta}{n} \sum_{j=1}^{n} \left(S_j^+ - S_j^- \right) \tag{2.5}$$

λ_{prior}, λ_{rating}, λ_{sim}, λ_{oppo} são parâmetros de peso que devem ser setados de acordo com o grau de confiança em cada fonte de informação e δ pode ser setado como um valor pequeno, como 0.01.

[5] http://www.purl.org/net/emolex

O problema de otimização é

$$S = \text{argmin}\Omega \tag{2.6}$$

sujeito a

$$S_j = S_j^+ - S_j^-, \forall j = 1...n \tag{2.7}$$

$$S_j^+, S_j^- \geq 0, \forall j = 1...n \tag{2.8}$$

$$-1 \geq S_j \geq 1, \forall j = 1...n \tag{2.9}$$

Esse problema de otimização foi transformado em uma problema de programação linear inteira. Experimentos realizados com dois conjuntos de dados (opiniões sobre hotéis e impressoras) mostraram que essa abordagem pode identificar tanto palavras específicas para domínio, quanto palavras cujo sentimento varia de acordo com o aspecto do domínio sendo analisado. Além disso, o uso desse léxico melhorou a precisão da tarefa de classificação de sentimento no nível de aspecto.

Tan and Wu Lexicon

Tan e Wu [61] propuseram uma abordagem para construir um léxico de sentimentos orientado a domínio usando dados rotulados de outro domínio. O algoritmo foi baseado no modelo de *random walk* utilizando várias relações entre documentos e palavras dos domínios origem e destino.

Random walk é a formalização matemática de um caminho que consiste em uma sucessão de passos randômicos. Ou seja, um modelo de *random walk* assume que, de um período de tempo para o próximo, apenas um passo randômico é dado a partir da última posição. Esse modelo é aplicado para ciência da computação, física e outros campos.

Quatro grafos foram gerados, um para cada tipo de relação, levando em consideração os dados classificados do domínio antigo e os dados não classificados do domínio novo. Para os dados classificados, 1 representa positivo e -1 representa negativo. Para os dados não classificados, o valor inicial é 0.

O algoritmo de *random walk* foi aplicado a esses grafos até convergir e chegar no resultado. Se a palavra foi classificada com uma nota entre 0 e 1, essa palavra tem sentimento positivo. Se a palavra foi classificada com nota entre -1 e 0, tem sentimento negativo. Quanto mais próximo de 1 ou -1, mais forte é o sentimento.

Os léxicos criados são relacionados aos domínios de eletrônicos, mercado de ações e hotéis. Os corpora utilizados são de *reviews* sobre esses assuntos advindos de diferentes fontes e a abordagem utilizada é baseada em corpus.

Abordagem usando programação linear inteira

Choi e Cardie [12] propuseram um método baseado em programação linear inteira para adaptar um léxico existente em um novo léxico que reflete as características dos dados utilizados. Esse método considera as relações entre palavras e expressões com opiniões para derivar a polaridade mais provável de cada item do léxico para aquele domínio.

O algoritmo de programação linear inteira foi baseado em um problema de otimização. A função objetiva desse problema pode ser vizualidada na Equação 2.10, onde x^+, x^-, $x^=$ e $x^\neg$ representam, respectivamente, as polaridades positiva, negativa, neutra e negativa de x.

$$\begin{aligned} \text{maximize} \quad & \sum_i \left(w_i^+ x_i^+ + w_i^= x_i^= + w_i^- x_i^- + w_i^\neg x_i^\neg - w_a \alpha_i \right) \\ - \quad & \sum_l w_\beta \rho_l \beta_l \\ - \quad & \sum_{l,k} w_\beta \rho_{(l,k)} \beta_{(l,k)} \end{aligned} \tag{2.10}$$

A avaliação do método proposto foi feita utilizando uma aplicação de classificação da polaridade no nível de expressão. Os resultados mostraram que o léxico adaptado possui melhores resultados do que o léxico original.

Abordagem usando *Deep learning*

Em [19], Glorot *et al.* estudaram o problema de adaptação de classificadores de sentimento em relação ao domínio. O objetivo é treinar o sistema em *reviews* rotuladas do domínio de origem e usar o sistema para um domínio de destino. Para isso, os autores propuseram uma abordagem baseada em *deep learning*, que aprende a extrair representações significativas de cada *review* de maneira não-supervisionada.

Classificadores de sentimento treinados usando essa representação de características de alto nível obteve resultados melhores que os métodos encontrados no estado da arte. Os testes foram realizados usando *reviews* de quatro tipos de produtos da Amazon. Além disso, o método proposto foi aplicado para fazer a adaptação para 22 domínios diferentes.

Esse trabalho não cria um léxico como os outros apresentados nesta seção, mas faz a adaptação do domínio para obter melhores resultados na ferramenta de análise de sentimento.

2.3.3 Léxicos para *Twitter*

Para capturar as particularidades do *Twitter*, vários léxicos específicos para *tweets* foram propostos.

NRC Hashtag Sentiment Lexicon

Em [37], foi demonstrado que *hashtags* de sentimento são bons indicadores do sentimento geral do *tweet*. Em [38], Mohammad *et al.* construíram um léxico a partir de 78 *hashtags* semente positivas e negativas. Essas *hashtags* foram usadas para classificar os *tweets* e ter um grande corpus dessas mensagens marcadas com sentimentos. Esses *tweets* foram obtidos através da *API* do *Twitter* e as consultas foram feitas de 4 em 4 horas durante 8 meses de 2012.

Com esse corpus, foi criado um grande léxico de sentimentos. O *score* de associação de um termo w é calculado a partir desses *tweets* pseudo-tageados como positivos ou negativos. Um *score* positivo significa um sentimento positivo enquanto um *score* negativo significa um sentimento negativo. A magnitude indica o grau de associação. O *score* é calculado usando a Fórmula 2.11, onde PMI se refere a informação mútua pontual.

$$score(w) = PMI(w, positive) - PMI(w, negative) \tag{2.11}$$

Informação mútua é uma medida da sobreposição de informações entre duas variáveis aleatórias. Informação mútua pontual, mais conhecida como *PMI* (*Pontual Mutual Information*), é uma medida do quanto a probabilidade de uma co-ocorrência de eventos particulares $p(x, y)$ difere do que seria esperado com base na probabilidade dos eventos individuais e na presunção da independência de x e y.

Esse léxico utiliza a abordagem baseada em corpus. Ele foi criado em 2012 como uma parte do trabalho que foi submetido ao SemEval 2013 - Tarefa: Análise de Sentimentos de *Tweets*. Com os bons resultados alcançados com esse léxico, esse trabalho ficou em primeiro lugar nessa categoria.

O léxico final contém 54.129 *unigrams* e 316.531 *bigrams*. Entradas para pares de *n-grams* que não são necessariamente contíguos nos *tweets* também foram gerados. O léxico final contém 308.808 pares não-contíguos de *unigram-unigram, unigram-bigram, bigram-unigram, bigram-bigram*.

Sentiment 140 Lexicon

Esse léxico foi criado pela mesma equipe do anterior e com o mesmo propósito. A abordagem para criação do léxico foi a mesma. A diferença é que foi usado o corpus

Sentiment 140 [20] como semente, que consiste em *emoticons* que contêm sentimentos claramente positivos ou negativos.

O léxico final contém 62.468 *unigrams*, 677.689 *bigrams* e 480.010 pares não contíguos.

Justin Bieber Lexicon

Esse léxico foi construído usando a abordagem baseada em corpus [18]. O conjunto de dados utilizado consiste de mais de dez milhões de *tweets* sobre Justin Bieber extraídos em 2012. Esse conjunto foi dividido entre desenvolvimento, treinamento e teste. Dessas mensagens, três mil foram classificadas manualmente por avaliadores humanos em Fortemente Positivo, Fracamente Positivo, Neutro, Fracamente Negativo ou Fortemente Negativo. Os avaliadores humanos também marcaram os *bigrams* e *trigrams* mais importantes de cada *tweet*. Cada mensagem foi avaliada duas vezes e apenas os *tweets* que tiveram a mesma classificação pelos dois avaliadores foram mantidos.

No estágio seguinte, os *bigrams* e *trigrams* que obtiveram mais marcações foram decompostos em *unigrams*. A frequência desses *unigrams* foi recalculada no corpus de três mil *tweets* avaliados manualmente. O conjunto resultante foi um conjunto de aproximadamente 700 *unigrams*. Nesse conjunto, foram avaliados os termos que têm uma forte afinidade com um dos sentimentos (positivo ou negativo) e os que ficaram no meio termo foram descartados. Foram adicionados sinônimos dos termos restantes ao conjunto.

Uma vez que os *bigrams* e *trigrams* originais foram decompostos e o conjunto de *unigrams* foi selecionado baseado na frequência dos termos, os *bigrams* e *trigrams* que ocorrem mais frequentemente puderam ser introduzidos no modelo de maneira segura, usando o método de afinidade.

O resultado é um léxico com 755 *n-grams* com $n = 1, 2, 3$. Esses *n-grams* foram divididos em 187 grupos de atributos para serem usadas nos algoritmos de aprendizagem de máquina.

Em seguida, os autores aumentaram esse léxico reduzido de forma manual com termos específicos utilizados em *tweets* sobre Justin Bieber. Esse léxico é específico para *tweets* e relacionado ao domínio Justin Bieber. Pelos testes, obteve resultados melhores do que usando um léxico genérico. Esse trabalho é específico para domínio e para *tweets* mas, por ser bastante manual, não pode ser estendido para outros domínios com facilidade.

2.3.4 Comparação dos Métodos

Na Tabela 2.4, é apresentada a comparação dos léxicos descritos na Seção 2.3. A coluna manual se refere à quantidade de trabalho manual necessário para construir o léxico. O ideal é que seja exigido um baixo trabalho manual, por isso esses léxicos estão

em destaque. A coluna *tweets* representa se o léxico é específico para *tweets*. No presente trabalho, pretende-se desenvolver um léxico específico para essas mensagens. A coluna domínio representa se o léxico é específico para algum domínio.

Pode-se constatar, pelos destaques feitos na tabela, que não foi encontrado na literatura um léxico que é, ao mesmo tempo, específico para *tweets*, específico para domínio e com pouco trabalho manual. Por isso, um algoritmo para construir tal léxico foi proposto no presente trabalho.

2.4 Análise de Sentimento

Nesta seção, são apresentados trabalhos que dizem respeito à análise de sentimentos. A Seção 2.3.1 explora as ferramentas de análise de sentimento genéricas, ou seja, para qualquer tipo de texto. A Seção 2.4.2 se refere a trabalhos que focam na análise de sentimentos específica para *tweets*. Em seguida, a Seção 2.4.3 aborda *surveys* sobre análise de sentimentos. Por fim, a Seção 2.4.4 apresenta comentários sobre os artigos estudados e a sua aplicabilidade à presente pesquisa.

2.4.1 Análise de Sentimento Genérica

A pesquisa realizada por Gonçalves *et al.* [21] comparou oito algoritmos de análise de sentimento existentes na literatura. Pela comparação, o que obteve o melhor resultado foi o uso de *emoticons* para prever o sentimento. Em seguida, foi desenvolvido um novo método que combina as abordagens existentes em uma. Esse método foi implementado e conseguiu superar os resultados dos oito algoritmos comparados.

Heerschop *et al.* [24] propuseram um *framework* chamado Pathos que desempenha a análise de sentimento baseado na estrutura de discurso do documento. A hipótese explorada é que dividir um texto em suas partes constituintes mais e menos importantes e levar essa informação em consideração ao dar pesos diferentes a cada parte do texto pode levar a uma melhoria no desempenho do classificador de sentimentos. A estrutura de discurso do documento é obtida aplicando a teoria da estrutura retórica de textos (*RST - Rethorical Structure Theory*) no nível de sentenças. A utilização de um algoritmo genético para otimizar os pesos levou a resultados melhores comparados ao *baseline*.

Em [48], os autores apresentaram um novo paradigma para análise de sentimentos no nível de conceitos que mistura linguística, algoritmos tradicionais e aprendizagem de máquina para melhorar o desempenho da detecção de polaridade. O algoritmo construído segundo a abordagem proposta obteve resultados melhores do que os existentes na literatura.

2.4.2 Análise de Sentimento de *Tweets*

Mohammad *et al.* [38] criaram um classificador de sentimentos baseado em máquinas de vetores de suporte. Os *tweets* foram representados como vetores de características levando em consideração vários grupos de características. Dentre eles, pode-se citar: *n-grams*, *part-of-speech*, *hashtags*, léxicos, pontuação, *emoticons*, palavras elongadas, negação e clusters. O classificador foi treinado usando quase 1.000 *tweets* e obteve um *f-measure* de 69%.

Em [2], os autores testaram duas abordagens para prover a classificação do sentimento de *tweets*: máquinas de vetores de suporte e uma função de pontuação de sentimentos. O melhor resultado foi obtido usando máquinas de vetores de suporte. Assim como o trabalho de Mohammad [38], esse também usa aprendizagem supervisionada e necessita de dados rotulados com sentimentos para treinar o algoritmo.

Saif *et al.* [56] propuseram adicionar semântica como uma característica adicional para o treinamento de uma ferramenta para classificação de sentimento de *tweets*. Para cada entidade extraída do texto, o conceito semântico relacionado é considerado no conjunto de características do *tweet*. Foram testadas três maneiras de considerar o conceito semântico: substituição, aumento e interpolação. Foi utilizado *Naive Bayes* para o treinamento da ferramenta de classificação.

Em [14], os autores usaram uma estratégia baseada nos k-vizinhos mais próximos (*kNN - k-nearest neighbors*) para fazer a classificação do sentimento de *tweets*. Foram usadas quatro tipos de características: baseadas em palavras e *n-grams*, baseadas em padrões, baseadas em pontuação e eficiência da seleção de características. Eles usaram 50 *hashtags* e 15 *emoticons* para construir um corpus de treinamento para o algoritmo.

2.4.3 *Surveys* sobre Análise de Sentimento

Há uma enorme gama de estudos acerca da análise de sentimento, que variam no tipo de aplicação, na definição da análise de sentimento, na abordagem para solução, entre outros. Por isso, foram feitos alguns estudos que focam em comparar e analisar vários artigos sobre análise de sentimentos e seus algoritmos e aplicações. Entre eles, podem ser citados [35], [42] e [41].

2.4.4 Comentários

Na Tabela 2.5, é apresentada a comparação dos algoritmos de análise de sentimento discutidos na Seção 2.4. A coluna *tweets* indica se o algoritmo é específico para *tweets* ou não.

Os estudos apresentados nesta seção são baseados, em sua maioria, em abordagens supervisionadas para a análise de sentimentos. Com isso, bons resultados são obtidos, mas ao custo de precisar de dados rotulados com sentimento. Isso não é fácil de conseguir para vários domínios, pois demanda muito trabalho manual. Por isso, o presente trabalho foca em uma abordagem não-supervisionada, eliminando a necessidade de dados rotulados e facilitando a aplicação do algoritmo a diferentes domínios.

2.5 Comentários Finais

Neste capítulo, foi apresentada a revisão da literatura e o estado da arte relativos às quatro etapas da ferramenta proposta. A Seção 2.1 apresentou coleta de *tweets* relacionados. Foram discutidos estudos que focam na expansão, no agrupamento e na recomendação de *hashtags* do *Twitter*.

A Seção 2.2 apresentou trabalhos sobre a identificação de *spam* no *Twitter*. Foram relatados trabalhos que focam em detectar *spammers*, detectar *tweets* que são *spam* e que tratam *spam* como uma fase de pré-processamento.

A Seção 2.3 mostrou diferentes maneiras de construir um léxico de sentimentos. Foram apresentados seis algoritmos para criar um léxico genérico e quatro algoritmos para criar um léxico para domínio específico. Foram também apresentados três métodos para criar léxicos de sentimento para *tweets*.

Por fim, a Seção 2.4 reuniu algoritmos de análise de sentimento encontrados na literatura. Análise de sentimento genérica e de *tweets* foram discutidas e *surveys* sobre análise de sentimentos foram apresentados.

Tabela 2.4: Comparação dos Léxicos Apresentados

Ano	**Nome**	**Refer.**	**Manual**	**Abordagem**	***Tweets***	**Domínio**	**Tamanho**
2013	NRC Hashtag Sentiment Lexicon	[38]	**Baixo**	Baseada em corpus	**Sim**	Não	54.129 *unigrams* + 316.531 *bigrams*
2013	Sentiment 140 Lexicon	[38]	**Baixo**	Baseada em corpus	**Sim**	Não	62.468 *unigrams* + 677.689 *bigrams* + 480.010 pairs
2013	Justin Bieber Lexicon	[18]	Alto	Baseada em corpus	**Sim**	**Sim**	187 *feature groups*
2011	Lu Lexicon	[32]	**Baixo**	Baseada em corpus	Não	**Sim**	4.627 pares aspecto-sentimento
2011	Tan e Wu Lexicon	[61]	Médio	Baseada em corpus	Não	**Sim**	-
2011	NRC EmoLex	[39]	Alto	Baseada em Dicionário	Não	Não	24.000 termos
2011	Deep Learning	[19]	**Baixo**	Baseada em corpus	Não	**Sim**	-
2010	Web Lexicon	[64]	**Baixo**	Baseada em corpus	Não	Não	178.000 *n-grams*
2009	*Label Propagation*	[50]	**Baixo**	Baseada em dicionário	Não	Não	-
2009	Programação Inteira	[12]	**Baixo**	Baseada em corpus	Não	**Sim**	-
2006	SentiWordNet	[16]	Médio	Baseada em Dicionário	Não	Não	115.000 *synsets*
2005	MPQA Lexicon	[65]	Alto	Baseada em Dicionário	Não	Não	8.000 *unigrams*
2004	Bing Liu Lexicon	[25]	Médio	Baseada em Dicionário	Não	Não	6.800 *unigrams*

Tabela 2.5: Comparação dos Algoritmos De Análise de Sentimento Apresentados

Ano	**Referência**	***Tweets***	**Método**
2014	[48]	Não	Linguística e Aprendizagem de Máquina
2013	[21]	Não	Combinação de 8 métodos existentes
2013	[38]	Sim	Máquinas de vetores de suporte
2012	[2]	Sim	Máquinas de vetores de suporte e função de pontuação de sentimentos
2012	[56]	Sim	*Naive Bayes* e informação semântica
2011	[24]	Não	Estrutura retórica de textos
2010	[14]	Sim	k-vizinhos mais próximos

Capítulo 3

Fundamentação Teórica

Neste capítulo, é apresentada a fundamentação teórica necessária para a realização do presente trabalho. A Seção 3.1 foca em léxicos de sentimento. A Seção 3.2 apresenta a análise de sentimentos. Já a Seção 3.3 explica a propagação de grafos. A Seção 3.4 reúne as métricas utilizadas no presente trabalho. Por fim, a Seção 3.5 traz comentários a respeito do apresentado no capítulo.

3.1 Léxicos de Sentimento

Há palavras que são comumente usadas para expressar sentimentos positivos ou negativos. Por exemplo, '*good*', '*beautiful*', '*impressive*' são palavras positivas enquanto '*bad*', '*ugly*' e '*disappointment*' são palavras negativas. Além disso, há expressões que designam sentimentos positivos ou negativos. Palavras e expressões que exprimem sentimentos são fundamentais para a análise de sentimento. Uma lista dessas palavras e expressões é chamada de léxico de sentimentos [31].

Léxicos de sentimentos são importantes para a análise de sentimentos mas possuem suas limitações. Primeiro, algumas palavras podem ser positivas ou negativas dependendo do domínio. Por exemplo, a palavra '*loud*' (em português, alto ou barulhento) é considerada positiva quando se está falando de um aparelho de som, mas é negativa quando se refere a uma geladeira.

Segundo, frases com negação de uma expressão positiva, se não for tomado o devido cuidado, pode acabar sendo considerada como positiva. A frase '*The new iPhone 6 is not pretty.*', por exemplo, é uma frase com sentimento negativo, mas se for usado apenas um léxico, a palavra '*pretty*' seria considerada como tendo um sentimento positivo e classificar a frase como positiva. Por isso, léxicos de sentimento devem ser usados juntamente com outras técnicas de processamento de linguagem natural (mais conhecido como *NLP - Natural Language Processing*) para se ter um bom resultado.

3.1.1 Específicos para *Tweets*

As ferramentas de NLP tradicionais são baseadas nas normas corretas das linguagens. Na internet, em especial no *Twitter*, as pessoas nem sempre usam uma linguagem formal. Os *tweets* são textos limitados a 140 caracteres e, por isso, muitos usuários usam abreviações, gírias, acrônimos e *emoticons* ao escreverem seus textos. Por exemplo, é comum escrever '*LOL*', que é o acrônimo de '*Laughing Out Loud*' em inglês.

Além disso, o *Twitter* tem outras características específicas da sua linguagem, como *#hashtags*, *@nomesdeusuario* e *www.links.com*, que podem ser usados pelos usuários. Para aplicar uma ferramenta de NLP tradicional a *tweets*, normalmente essas informações são extraídas e o que restou do texto é processado. Porém, algumas informações essenciais para a classificação do *tweet* podem ser perdidas nessa extração.

Por essas peculiaridades, é possível afirmar que um tratamento mais direcionado às características dos *tweets* melhora o desempenho dos léxicos de sentimento e, consequentemente, dos algoritmos de análise de sentimentos dessas mensagens.

3.1.2 Relacionados a Domínios

Como foi visto anteriormente, os léxicos de sentimento específicos para *tweets* melhoram o desempenho de algoritmos de análise de sentimento dessas mensagens. Porém, eles ainda apresentam algumas falhas, pois algumas palavras podem ter um significado positivo em um domínio e negativo em outro domínio. Uma solução para esse problema é construir léxicos de sentimento relacionados a domínios, para aumentar a precisão da análise de sentimento naqueles domínios.

Em [18], foi demonstrado que, quando se limita o léxico a um determinado domínio, aparecem alguns termos no léxicos que são gerais e outros termos que são específicos daquele domínio, que não seriam identificados num léxico de propósito geral. Neste trabalho, foi construído um léxico específico para o domínio Justin Bieber e eles encontraram termos que normalmente são identificados como negativos (ex: *fever* (febre em inglês), mas que no contexto de Justin Bieber é considerado um termo positivo (*Bieber fever*)).

Com o exposto acima, faz sentido construir léxicos de sentimento específicos para *tweets* e relacionados a domínio, pois agrega os benefícios das duas abordagens para construção de um léxico e obtém maior precisão dos resultados.

3.1.3 Construção de Léxicos de Sentimento

Existem duas abordagens principais na literatura para a construção de léxicos de sentimentos: baseadas em dicionário e baseadas em corpus [31]. A abordagem baseada

em dicionário é mais usada em léxicos de propósito geral. Já a abordagem baseada em corpus é mais usada em léxicos para domínio específico.

Abordagens Baseadas em Dicionário

Usar um dicionário para gerar o léxico é uma abordagem óbvia, porque dicionários contêm a maior parte das palavras e listam sinônimos e antônimos para cada palavra [31]. Então, uma técnica simples é usar algumas palavras que são claramente positivas ou negativas como semente e expandir isso usando os sinônimos e os antônimos do dicionário.

Em detelhes, o método funciona da seguinte maneira: um conjunto de palavras inicial com orientação positiva ou negativa é criado. A cada rodada, os sinônimos e os antônimos das palavras do conjunto são buscadas no dicionário e as novas palavras adicionadas ao léxico de acordo com a sua orientação. O processo termina quando não puderem ser encontradas palavras novas.

A principal vantagem da abordagem baseada em dicionário é que se encontra um número grande de palavras e suas orientações de maneira fácil e rápida. Porém, o resultado normalmente apresenta muitos erros. Assim, um processo manual pode ser usado para verificar se o resultado está correto. Essa verificação manual é trabalhosa. Porém, ela só precisa ser feita uma vez e o léxico criado pode ser usado diversas vezes.

A principal desvantagem da abordagem baseada em dicionário é que as palavras encontradas são independentes de domínio. É difícil usar essa abordagem para encontrar as orientações de palavras que são dependentes de contexto ou domínio. Com isso, para o presente trabalho, outro tipo de abordagem é mais adequada.

Outra desvantagem dessa abordagem é que só considera palavras e não expressões, visto que os dicionários trabalham com palavras.

Abordagens Baseadas em Corpus

A abordagem baseada em corpus faz uso de um corpus de textos para construir o léxico de sentimentos. Existem dois cenários principais:

1. A partir de uma lista inicial de palavras (normalmente de propósito geral) com as respectivas orientações de sentimento, descobrir novas palavras e suas orientações usando um corpus de um domínio específico.

2. Adaptar um léxico de sentimento de propósito-geral para um novo léxico relacionado a domínio usando um corpus do domínio.

No presente trabalho, foi escolhido seguir o primeiro cenário. O algoritmo proposto é apresentado no Capítulo 6.

3.1.4 Descrição Formal do Problema

O problema de criar um léxico de sentimento segundo a abordagem baseada em corpus pode ser definido formalmente da seguinte maneira. Dado um conjunto de textos T do domínio escolhido, onde $t \in T$ é uma *string* de caracteres, o léxico de sentimentos associado a esse domínio é um conjunto de tuplas $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, ...\}$. Cada tupla γ_x pode ser definida como $\gamma_x = (w, s)$, onde $w \in TT$ é um *token* e $s \in \mathbb{R}$ é a nota associada a w.

O conjunto TT pode ser definido como o conjunto de todos os *tokens* extraídos dos textos de T, ou seja, $TT = \cup tok(t), \forall t \in T$, onde $tok(x)$ é o conjunto de *tokens* resultado da *tokenização* de x. A *tokenização* de uma frase refere-se ao processo de dividir essa frase nas suas palavras constituintes, eliminando pontuação e outros elementos desnecessários ao processamento da frase. Ou seja, $tok(t)$ é uma função que recebe uma *string* de caracteres e retorna um conjunto de *strings* de caracteres referentes às suas partes constituintes.

A nota s associada a um *token* w é um indicativo do sentimento desse *token*. Se $s \geq 0$, então w possui uma conotação positiva. Por outro lado, se $s < 0$, então w possui uma conotação negativa.

Já o problema de criar um léxico de sentimento segundo a abordagem baseada em dicionário possui a mesma saída esperada, qual seja, um conjunto de tuplas $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, ...\}$. Porém, nesse caso, a entrada do problema é diferente, pois é constituída de um dicionário formal constituído de sinônimos e antônimos, além de informações de um etiquetador morfológico (*part-of-speech*).

3.2 Análise de Sentimento

Linguagem Natural é definida como uma linguagem que é usada para a comunicação diária de humanos, ou seja, linguagens como inglês e português [6]. Ao contrário de linguagens artificiais, como linguagens de programação e notações matemáticas, linguagens naturais evoluíram através das gerações, sendo difícil muitas vezes explicar as suas regras de construção.

Processamento de linguagem natural (NLP) é definido como qualquer tipo de manipulação por computador de uma linguagem natural. Por um lado, pode ser simples como contar a frequência de palavras para comparar diferentes estilos de escrita. Por outro lado, NLP envolve entender a comunicação humana e conseguir dar respostas úteis para perguntas feitas por humanos. Um dos problemas existentes no NLP é a análise de sentimentos. Segundo [31], análise de sentimento é a área de estudo responsável por analisar as opiniões, sentimentos, avaliações, atitudes e emoções das pessoas sobre entidades

como produtos, serviços, organizações, indivíduos, assuntos, eventos, tópicos, e os seus atributos.

Há também vários nomes e subtarefas relacionadas à análise de sentimento, como mineração de opiniões, extração de opiniões, mineração de sentimentos, análise de emoções e mineração de *reviews*.

NLP tem sido estudado por cientistas da computação há bastante tempo. Porém, o interesse pela análise de sentimentos é mais recente. A grande quantidade de dados disponíveis nas redes sociais e em blogs, que muitas vezes expressam opiniões ou sentimentos, impulsionou o interesse por essa área. Na análise do sentimento de um texto, há diferentes níveis de análise:

1. Nível de documento: o objetivo é saber o sentimento geral do texto sobre o assunto a que ele se refere. O mesmo texto pode conter frases positivas e negativas sobre o assunto e é preciso consolidar o sentimento predominante do texto.

2. Nível de sentença: o objetivo é saber o sentimento de uma sentença do texto. A mesma sentença pode ter ter sentimentos conflitantes e, da mesma forma, a intenção é saber o sentimento predominante da sentença.

3. Nível de palavra: o objetivo é saber se uma determinada palavra no texto tem conotação positiva ou negativa.

Em alguns casos, o documento ou sentença não apresenta nenhum sentimento ou opinião. Por isso, para algumas aplicações, é necessário saber se um determinado documento é objetivo ou subjetivo. Isso é chamado de polaridade-SO, onde SO significa subjetivo e objetivo. O sentimento do texto é analisado apenas se ele for subjetivo. O sentimento pode ser positivo ou negativo. Isso é chamado de polaridade-PN. Por exemplo, a frase '*I bought an iPhone*' é objetiva e não contém nenhuma opinião sobre o objeto, enquanto a frase '*I loved my new iPhone*' é subjetiva e implica em uma opinião positiva.

Outra abordagem para esse problema é categorizar o sentimento de um texto como positivo ou negativo. Por exemplo, na análise de sentimento de *tweets* sobre um determinado assunto é suficiente saber a quantidade de *tweets* negativos e positivos. Com isso, é possível extrair a opinião geral das pessoas no *Twitter* sobre o assunto.

Em outras aplicações isso não é suficiente. Por exemplo, se um usuário deseja comprar um celular novo, pode não ser suficiente saber se a opinião geral dos usuários sobre o celular é positiva ou negativa. Pode ser mais interessante saber sobre alguns aspectos específicos do produto, como a bateria ou a resolução da tela.

Para isso, foi criado o conceito de quíntuplas de opiniões [31]. Segundo o autor, uma opinião pode ser definida como uma quíntupla de informações que contêm o nome da

entidade, o aspecto da entidade, o sentimento a respeito daquele aspecto, a pessoa que deu a opinião e quando a opinião foi emitida. Essa definição é importante quando se deseja extrair diversas opiniões diferentes e consolidar isso em relatórios detalhados.

O nível de análise do sentimento extraída de um texto depende da aplicação. Para o propósito deste estudo, define-se análise de sentimento como sendo a subtarefa de categorizar o sentimento de um texto como positivo ou negativo.

3.2.1 Análise de Sentimento de *Tweets*

A análise de sentimentos de *tweets* é uma tarefa relacionada à análise de sentimentos tradicional, com algumas peculiaridades. Por um lado, a análise de *tweets* pode ser considerada mais fácil do que a análise de textos tradicionais pelo tamanho curto dessas mensagens, o que faz com que normalmente elas sejam mais objetivas na opinião que querem expressar. Por outro lado, o uso de gírias, abreviações, sarcasmo, *emoticons*, *hashtags*, menções e *links* podem dificultar o trabalho de processamento desses textos, tornando-os mais difíceis do que textos tradicionais.

A análise de sentimentos de *tweets* normalmente é usada para analisar a opinião geral dos usuários sobre um determinado tópico, normalmente através de filtros de *hashtags*, usuários ou termos. Então, na maioria dos casos, é suficiente extrair se o *tweet* apresenta uma opinião positiva ou negativa, sem precisar quebrar em quíntuplas como foi visto na seção anterior.

Muitas abordagens para esse problema podem ser encontradas na literatura. Uma boa parte delas faz uso de algoritmos de aprendizagem de máquina ([38], [2], [56], [15]). Os algoritmos mais comuns usados com essa finalidade são máquinas de vetores de suporte, entropia máxima e classificador *Naive Bayes*.

3.2.2 Descrição Formal do Problema

Para o presente trabalho, a análise de sentimento de textos pode ser definido formalmente da seguinte maneira. Dado um conjunto de textos T, onde $t \in T$ é uma *string* de caracteres, a análise de sentimento desse conjunto corresponde a um conjunto de tuplas $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, ...\}$. Cada tupla γ_x pode ser definida como $\gamma_x = (t, n, s)$, onde $t \in T$, $n \in \mathbb{R}$ é a nota associada a t e $s \in \{Positivo, Negativo\}$ é o sentimento de t.

O sentimento s é derivado de n através da Equação 3.1.

$$s = \begin{cases} Positivo & \text{se } n \geq 0 \\ Negativo & \text{caso contrário} \end{cases} \tag{3.1}$$

O sentimento geral do conjunto de textos pode ser extraído computando a porcentagem deles que são positivos e negativos.

3.3 Propagação de Grafos

Nesta seção, são estudados os algoritmos de propagação de grafos utilizados na construção de léxicos.

3.3.1 *Label Propagation*

Em [69], Zhu e Ghahramani propuseram um algoritmo de *label propagation* para propagar informações de dados rolutados para dados não-rotulados, tornando possível o uso de um grande número de dados não-rotulados para realizar treinamento com uma abordagem semi-supervisionada.

Esse algoritmo é um *framework* de aprendizagem de máquina que usa poucos exemplos rotulados, também chamados de sementes, para categorizar um grande número de exemplos não-rotulados. Além dos exemplos semente, esse algoritmo também usa uma relação entre os exemplos. Essa relação tem dois requisitos: deve ser transitiva e deve codificar alguma noção de similaridade entre os exemplos.

A relação entre os exemplos pode ser facilmente representada como um grafo. Cada nó do grafo é um exemplo e as arestas representam a similaridade entre os nós. Associada a cada nó, também há uma nota relacionada. No caso de léxicos de sentimento, essa nota representa a polaridade do nó. Para os nós semente, essa nota é fixa. O objetivo do algoritmo é derivar essa nota para os demais nós.

Considere um grafo $G = (V, E)$ onde V são os nós e E as arestas. O algoritmo de propagação de grafo tem como objetivo minimizar a função quadrática de energia apresentada na Equação 3.2, onde y_i e y_j são as notas associadas aos nós i e j, respectivamente.

$$\varepsilon = \frac{1}{2} \sum_{(i,j) \in E} w_{ij} (y_i - y_j)^2 \tag{3.2}$$

Assim, para derivar as notas de y_i, é setado $\frac{\varrho \varepsilon}{\varrho_{y_i}} = 0$ para obter a equação de atualização 3.3.

$$y_i = \frac{\sum\limits_{(i,j) \in E} w_{ij} y_j}{\sum\limits_{(i,j) \in E} w_{ij}} \tag{3.3}$$

Uma matriz estocástica de transição T é derivada da normalização das linhas de W, onde W é uma matriz $n \times n$, $n = |V|$ e $W = [w_{ij}]$. A matriz T é obtida através da Equação 3.4 e pode ser vista como a probabilidade de transição do nó j ao nó i.

$$T_{ij} = P(j \to i) = \frac{w_{ij}}{\sum_{k=1}^{n} w_{kj}} \tag{3.4}$$

O algoritmo funciona da seguinte forma:

1. Crie uma matriz Y $(n \times C)$ com a nota inicial dos nós, onde C é o número de classes do problema.
2. Propague as notas para todos os nós computando $Y = TY$.
3. Normalize Y linha a linha tal que a soma de cada linha totalize 1.
4. Volte os exemplos semente em Y à sua nota original.
5. Repita os passos 2-5 até Y convergir.

Esse algoritmo pode ser usado para aprendizagem não-supervisionada em diversas aplicações diferentes. Em [50], esse algoritmo foi usado para construir um léxico de sentimentos, como apresentado na Seção 2.3.1.

3.3.2 *Graph Propagation*

Em [64], os autores propuseram uma adaptação ao algoritmo de *label propagation* chamado de *graph propagation*. *Label propagation* é um algoritmo iterativo onde cada nó recebe como nota a média ponderada dos valores dos seus vizinhos na iteração anterior. O resultado é que nós com vários caminhos a partir dos nós semente recebem notas altas por causa da influência dos seus vizinhos.

O algoritmo de *label propagation* possui propriedades desejáveis de convergência, com uma função objetivo bem definida para minimizar o erro quadrático entre os valores dos nós adjacentes e uma equivalência para computar *random walks* pelo grafo.

A principal diferença entre o algoritmo original e o algoritmo de propagação de grafos é que um nó com múltiplos caminhos desde os nós semente serão influenciados por todos esses caminhos no algoritmo de *label propagation*. Já no algoritmo de propagação de grafos, apenas os caminhos com o maior peso a partir de cada nó semente será considerado.

A intuição para o algoritmo de *label propagation* é que, se um nó possui múltiplos caminhos até o nó semente, isso deve resultar em uma nota maior. Isso é verdade quando o grafo é de alta qualidade e todos os caminhos são confiáveis. Porém, em um grafo

construído a partir de fontes menos confiáveis, como a web ou *tweets*, raramente esse é o caso.

Essa é a motivação para a adaptação do algoritmo original no algoritmo de propagação de grafos, utilizado nessa pesquisa. Essa adaptação pode ser visualizada como o Algoritmo 4, apresentado no Capítulo 6.

3.4 Métricas

Nesta seção, são apresentadas as métricas utilizadas no presente trabalho. Em 3.5, é discutida a distância de Levenshtein. Em seguida, é apresentada a similaridade de cossenos. Por fim, a última seção diz respeito às métricas de avaliação de resultados.

3.4.1 Distância de Levenshtein

Em teoria da informação e ciência da computação, a distância de Levenshtein ou distância de edição é uma métrica para calcular a diferença entre duas sequências de caracteres. A distância de Levenshtein entre duas palavras é o número mínimo de operações necessárias para transformar uma palavra na outra [29]. Dentre essas operações, estão inserção, deleção e substituição.

Matematicamente, a distância de Levenshtein entre duas *strings* a e b é igual a $lev_{a,b}(|a|,|b|)$ onde $|a|$ é o tamanho da string a e lev é calculado segundo a Fórmula 3.5.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{se } min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{caso contrário} \end{cases} \tag{3.5}$$

Na equação, $1_{(a_i \neq b_j)}$ é igual a 0 se $a_i = b_j$ e 1 caso contrário.

Por exemplo, a distância de Levenshtein entre as palavras 'solar' e 'ensolação' é 5, já que com apenas 5 edições é possível transformar uma palavra na outra e não há maneira de o fazer com menos de 5 edições.

1. solar
2. nsolar (inserção de 'n' no início)
3. ensolar (inserção de 'e' no início)
4. ensolaç (substituição de 'r' por 'ç')

5. ensolaçã (inserção de 'ã' no final)

6. ensolação (inserção de 'o' no final)

A distância de Levenshtein foi proposta em 1966 por Vladimir Levenshtein e é usada em várias aplicações diferentes, como corretor ortográfico, sistemas de correção para reconhecimento ótimo de caracteres e sistemas de tradução de linguagem natural.

3.4.2 Similaridade de Cossenos

Similaridade de cossenos (*cosine similarity*, em inglês) é a medida de similaridade entre dois vetores de um espaço interno do produto que mede o cosseno do ângulo entre eles [57]. O cosseno de 0° é 1 e é menor do que 1 para qualquer outro ângulo. Por isso, é uma medida que leva em conta a orientação e não a magnitude: dois vetores com a mesma direção terão a similaridade de cossenos igual a 1, dois vetores a 90° terão similaridade igual a 0 e dois vetores diametralmente opostos terão similaridade igual a -1, independente da sua magnitude.

Dados dois vetores de atributos A e B, a similaridade de cossenos entre eles é representada pela Fórmula 3.6.

$$similaridade = cos(\theta) = \frac{A.B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{3.6}$$

A similaridade resultante pode ser -1 significando que são opostos, 1 indicando o mesmo significado e 0 indicando independência. Os valores intermediários indicam o grau de similaridade ou dissimilaridade.

Similaridade de cossenos é bastante usada em aplicações de recuperação de informação e mineração de textos. Nesses casos, cada termo é considerado como uma dimensão diferente e um documento é caracterizado como um vetor onde o valor de cada dimensão corresponde ao número de vezes que o termo aparece no documento. A similaridade de cossenos então dá uma medida útil do quão similar dois documentos realmente são em termos de conteúdo.

3.4.3 Métricas de Avaliação

Nesta seção, são apresentadas as métricas utilizadas para comparar o desempenho dos algoritmos propostos. Dentre essas métricas, estão precisão, *recall* e *f-measure* com diferentes parâmetros.

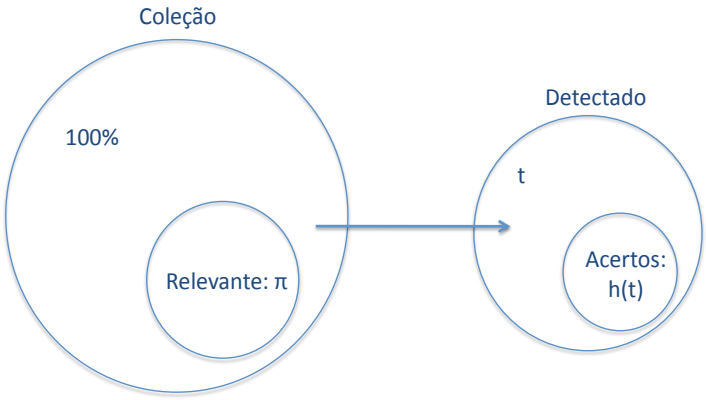

Figura 3.1: Ilustração de uma Operação de Detecção Típica

Precisão

No campo da recuperação de informação, precisão é a fração de documentos recuperados que são relevantes [68]. Precisão também é chamada de valor preditivo positivo.

Seja $\mathcal{C}$ uma grande coleção de itens, dos quais apenas uma fração π ($\pi \ll 1$) é relevante para a aplicação. Seja um algoritmo que detecte uma fração $t \in [0, 1]$ de $\mathcal{C}$, dos quais apenas $h(t) \in [0, t]$ são depois confirmados como relevantes. Esses são os acertos (ou *hits*) do algoritmo. Os demais são chamados de erros (ou *misses*). A Figura 3.1 mostra graficamente essas informações.

Seja ω um item randomicamente escolhido da coleção $\mathcal{C}$. Matematicamente, precisão pode ser definida em função de y e $\hat{y}$, de acordo com as Equações 3.7, 3.8 e 3.9.

$$y = \begin{cases} 1 & \text{se } \omega \text{ é relevante} \\ 0 & \text{caso contrário} \end{cases} \tag{3.7}$$

$$\hat{y} = \begin{cases} 1 & \text{se o algoritmo detecta } \omega \\ 0 & \text{caso contrário} \end{cases} \tag{3.8}$$

$$\text{Precisão} = Pr(y = 1 | \hat{y} = 1) \tag{3.9}$$

Para uma tarefa de classificação, precisão pode ser definida como o número de verdadeiros positivos (ou seja, número de itens corretamente rotulados como pertencentes à classe positiva) dividido pelo número total de elementos rotulados como positivos (ou seja, a soma dos verdadeiros positivos e falsos positivos). Por exemplo, para o resultado de uma busca textual em um conjunto de documentos, precisão é o número de resultados corretos dividido pelo número de todos os resultados retornados.

Considerando essa definição, pode-se constatar que maximizar a precisão significa minimizar os falsos positivos.

Recall

No campo da recuperação de informação, *recall* é a fração de documentos relevantes que são recuperados [68]. *Recall* também é chamada de sensibilidade. Essa métrica pode ser calculada segundo a Fórmula 3.10.

$$Recall = Pr(\hat{y} = 1|y = 1) \tag{3.10}$$

Para uma tarefa de classificação, *recall* pode ser definido como o número de verdadeiros positivos dividido pelo número total de elementos que efetivamente pertencem à classe positiva (ou seja, a soma de verdadeiros positivos e falsos negativos). Por exemplo, para o resultado de uma busca textual em um conjunto de documentos, *recall* é o número de resultados corretos dividido pelo número de resultados que deveriam ter sido retornados.

Considerando essa definição, pode-se constatar que maximizar o *recall* significa minimizar os falsos negativos.

F-Measure

F-measure ou *F-score* balanceado é uma medida da exatidão de um teste. É uma métrica que combina precisão e *recall* para calcular sua pontuação e é a média harmônica entre essas duas métricas, segundo a Equação 3.11.

$$F = 2 \cdot \frac{\text{precisão} \cdot \text{recall}}{\text{precisão} + \text{recall}} \tag{3.11}$$

Quando precisão e *recall* são igualmente importantes, deve-se focar em maximizar o *f-measure*. Nesse caso, deve-se levar em consideração que normalmente não é possível otimizar a precisão e o *recall* ao mesmo tempo, pois essas duas métricas normalmente constituem um *trade-off*. Esse *trade-off* entre precisão e *recall* é bem conhecido na literatura ([23], [10]).

Em alguns casos, uma das métricas é mais importante. Para isso, pode ser usada a métrica F_β genérica, cuja fórmula pode ser vista na Equação 3.12, onde $\beta \geq 0$. $\beta < 1$ dá uma ênfase maior à precisão, enquanto $\beta > 1$ dá uma ênfase maior ao *recall*. As métricas mais usadas são $F_{0.5}$, F_1 e F_2.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precisão} \cdot \text{recall}}{(\beta^2 \cdot \text{precisão}) + \text{recall}} \tag{3.12}$$

3.5 Comentários

Neste capítulo, foi apresentada a fundamentação teórica para o presente trabalho. A Seção 3.1 focou em léxicos de sentimento. Foram apresentados léxicos de sentimento específicos para *tweets* e relacionados a domínios. Em seguida, foram retratadas as principais abordagens para a construção de léxicos de sentimento. Por fim, uma descrição formal do problema foi proposta.

A Seção 3.2 apresentou a análise de sentimento e os problemas associados. A análise de sentimento de *tweets* foi discutida em seguida. Em seguida, uma descrição formal do problema foi proposta.

Já a Seção 3.3 explicou a propagação de grafos. Dois algoritmos foram apresentados: *label propagation* e *graph propagation*.

Por fim, a Seção 3.4 reuniu as métricas utilizadas no presente trabalho, englobando a distância de *Levenshtein* e a similaridade de cossenos. Também foram apresentadas métricas de avaliação, como precisão, *recall* e *f-measure*.

Capítulo 4

Coleta de *Tweets* Relacionados

Este capítulo trata da primeira etapa da ferramenta, que consiste em coletar *tweets* relacionados a um tópico específico. Normalmente, para coletar *tweets* sobre um tópico, o usuário deve fornecer um conjunto de termos e *hashtags* sobre esse tópico para ser passado como parâmetro para uma busca na *API* do *Twitter*. Para isso, o usuário precisa entrar no *Twitter* e pesquisar quais *hashtags* estão sendo usadas para esse propósito. Para evitar que o usuário tenha esse trabalho, foi criado o Algoritmo de Expansão de *Hashtags*. Em vez de um conjunto grande de termos e *hashtags*, o usuário só precisa fornecer um conjunto pequeno de termos e/ou *hashtags* sobre o tópico e o algoritmo fica responsável por expandir esse conjunto. Com base nesse conjunto maior, é possível pesquisar na *API* do *Twitter* por *tweets* que contêm essas *hashtags* e coletar *tweets* relacionados ao tópico.

Particularmente, duas *hashtags* são ditas relacionadas se elas se referem ao mesmo tópico (ex: #WorldCup2014 e #Brazil2014 falam do mesmo evento), algum atributo daquele tópico (ex: #iPhone6 e #iPhone6Battery) ou um tópico fortemente relacionado (ex: #ecig e #eliquid). E-líquido (e-liquid, em inglês) é o líquido usado nos cigarros eletrônicos.

A Seção 4.1 apresenta o algoritmo de expansão de *hashtags* proposto. A Seção 4.2 mostra o estudo de caso realizado. Por fim, a Seção 4.3 discute os trabalhos futuros a respeito dessa etapa da ferramenta.

4.1 Algoritmo de Expansão de *Hashtags*

A entrada do Algoritmo de Expansão de *Hashtags* consiste em três conjuntos: *hashtags* de entrada, termos de entrada e termos ignorados. *Hashtags* de entrada proporcionam as *hashtags* semente para o algoritmo, ou seja, *hashtags* sobre o tópico escolhido que serão usadas para capturar outras *hashtags*. Termos de entrada são termos alternativos

relacionados ao tópico. Termos ignorados são termos que se quer excluir do conjunto final de *tweets*.

Por exemplo, suponha que se quer encontrar *hashtags* sobre o novo produto da *Apple*: o *iPhone 6*. O conjunto de *hashtags* de entrada poderia ser a *hashtag* '#iphone6', os termos de entrada poderiam ser '*iphone 6*', '*iphone 6 plus*' e os termos ignorados poderiam ser '*iphone2*', '*iphone3*', '*iphone4*', '*iphone5*', pois não há intenção de coletar *tweets* sobre versões anteriores do *iPhone*.

A ideia principal do algoritmo é achar *hashtags* que contenham uma ou mais das *hashtags* ou termos semente e não contenham nenhum dos termos ignorados. O algoritmo é desenvolvido de forma a tratar alguns erros de digitação das *hashtags*, que ocorrem frequentemente devido à natureza informal dos *tweets*. Se uma *hashtag* é similar o suficiente a uma das *hashtags* ou termos semente, ela será considerada relacionada. Se a *hashtag* é mais similar a um termo ignorado do que a qualquer das *hashtags* ou termos semente, ela não será considerada relacionada.

Essa similiaridade é calculada com base em uma versão adaptada da distância de Levenshtein e leva em consideração se uma *string* é *substring* de outra. O algoritmo de *substring* captura as *hashtags* que contêm um dos termos ou *hashtags* iniciais como *substring* e não contém nenhum dos termos ignorados. Isso irá capturar muitos *tweets* que falam sobre algum atributo do tópico original.

A distância de Levenshtein é usada com diferentes pesos para inserção, deleção e substituição. Matematicamente, a distância entre duas *strings* a e b é igual a $lev_{a,b}(|a|,|b|)/min(|a|,|b|)$ onde $|a|$ é o tamanho da string a e lev é calculado segundo a Fórmula 4.1, onde α representa o custo de inserção, θ o custo de deleção e γ o custo de substituição.

Esses pesos são customizados para alcançar os parâmetros ótimos, que proporcionam os melhores resultados. Para implementar a ideia de que uma letra alterada em uma palavra de quatro letras significa mais do que uma letra alterada em uma palavra de doze letras, uma versão normalizada da distância de Levenshtein foi desenvolvida, dividindo a distância pelo tamanho da menor das duas *strings* sendo comparadas. É estabelecido um limiar do quão diferentes duas *hashtags* podem ser e ainda serem consideradas relacionadas. Esse limiar é customizável. Esse passo ajuda a capturar alguns erros de digitação que são comuns em plataformas informais como o *Twitter*.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{se } min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + \alpha \\ lev_{a,b}(i,j-1) + \theta \\ lev_{a,b}(i-1,j-1) + \gamma_{(a_i \neq b_j)} \end{cases} & \text{caso contrário} \end{cases} \tag{4.1}$$

Além disso, apenas são levadas em consideração *hashtags* que aparecem em mais do que uma porcentagem dos *tweets* que contêm as *hashtags* semente. Esse limiar também é customizável e foi proposto com a intenção de ignorar as *hashtags* que aparecem apenas um número pequeno de vezes no conjunto de dados. *Hashtags* com tamanho menor que um limiar estabelecido são ignoradas, uma vez que elas provavelmente não significam nada e aumentam bastante o volume de *hashtags* a serem analisadas.

Todos esses parâmetros customizáveis foram usados para achar a melhor combinação de parâmetros que levou aos melhores resultados, como será explicado na Seção 4.2.2.

4.1.1 Pseudo-código do Algoritmo

O Algoritmo 1 mostra o pseudo-código do algoritmo de expansão de *hashtags*. A entrada do algoritmo consiste de um conjunto de *hashtags* semente, um conjunto de termos semente e um conjuntos de termos ignorados. Primeiramente, é coletado um conjunto contendo todas as *hashtags* que co-ocorrem com uma das *hashtags* semente em mais do que uma porcentagem mínima de *tweets*. Depois, para cada *hashtag* nesse conjunto, é calculada a distância entre ela e cada uma das *hashtags* semente e a menor distância encontrada é escolhida. O mesmo é feito para os termos relacionados e os termos ignorados. Finalmente, é decidido se uma *hashtag* deve ser selecionada se a distância mínima encontrada para as *hashtags* e termos relacionados é menor do que um limiar e se essa distância é menor do que a distância mínima dos termos ignorados. O conjunto final contendo apenas as *hashtags* relacionadas é retornado.

4.1.2 Métricas

Nesta seção, as métricas usadas para comparar o desempenho do algoritmo são apresentadas.

Precisão

Na Seção 3.4.3, foram vistas a definição e a fórmula para o cálculo da precisão. Adaptando para o algoritmo de expansão de *hashtags*, precisão é a fração de *hashtags* selecionadas pelo algoritmo que são realmente *hashtags* relacionadas, como visto na Equação 4.2.

$$\text{Precisão} = \frac{\text{número de } \textit{hashtags} \text{ relacionadas selecionadas pelo algoritmo}}{\text{número de } \textit{hashtags} \text{ selecionadas pelo algoritmo}} \tag{4.2}$$

Algoritmo 1: Algoritmo de Expansão de *Hashtags*

```
   Data: SH, ST, IT
   Result: hashtags
   /* SH significa hashtags semente, ST termos semente e IT termos ignorados.   */
   /* O algoritmo retorna um conjunto de hashtags relacionadas.                 */
 1 hashtags = {};
 2 allHashtags = getHashtagsThatCooccur(ST, percentage);
   /* Retorna todas as hashtags que co-ocorrem com uma das hashtags semente em pelo
      menos uma porcentagem mínima de tweets                                    */
 3 for hashtag ∈ allHashtags do
 4     for seed ∈ SH do
 5         distance := NLD(hashtag, seed);
           /* NLD significa Distância de Levenshtein Normalizada                */
 6         minDist1 := minimum(distance, minDist1);
 7     end
 8     for term ∈ ST do
 9         distance := NLD(hashtag, term);
10         minDist1 := minimum(distance, minDist1);
11     end
12     for term ∈ IT do
13         distance := NLD(hashtag, term);
14         minDist2 := minimum(distance, minDist2);
15     end
16     if minDist1 < t AND minDist1 < minDist2 then
17         hashtags.add(hashtag);
18     end
19 end
20 return hashtags;
```

Se não há falsos positivos, todas as *hashtags* selecionados pelo algoritmo são realmente relacionadas, derivando uma precisão de 1. Se apenas metade das *hashtags* selecionadas pelo algoritmo são verdadeiramente relacionadas, a precisão é de 0.5. Com isso, pode-se constatar que maximizar a precisão significa minimizar os falsos positivos, o que significa que poucas *hashtags* selecionadas pelo algoritmo não são realmente relacionadas às *hashtags* semente.

Maximizar a precisão garante que a maior parte dos *tweets* capturados com essas *hashtags* são realmente relacionados ao tópico pretendido, mesmo que alguns *tweets* relacionados deixem de ser capturados.

Recall

Na Seção 3.4.3, foram vistas a definição e a fórmula para o cálculo do *recall*. Adaptando para o algoritmo de expansão de *hashtags*, *recall* é a fração de *hashtags* relacionadas que são selecionadas pelo algoritmo, como visto na Equação 4.3.

$$Recall = \frac{\text{número de } \textit{hashtags} \text{ relacionadas selecionados pelo algoritmo}}{\text{número de } \textit{hashtags} \text{ relacionadas}} \tag{4.3}$$

Se não há nenhum falso negativo, todas as *hashtags* relacionadas foram capturadas pelo algoritmo, derivando um *recall* de 1. Se metade das *hashtags* relacionadas foram encontradas, o *recall* é 0.5. Com isso, pode-se ver que maximizar o *recall* significa minimizar os falsos negativos.

Maximizar o *recall* garante que serão capturados a maior parte dos *tweets* relacionados, mesmo que alguns *tweets* que não são relacionados ao tópico também sejam capturados.

Otimização

Ao otimizar um algoritmo, a métrica utilizada como referência é importante. Se, na aplicação do algoritmo de expansão de *hashtags*, a intenção é minimizar os falsos negativos, para evitar deixar de fora qualquer *hashtag* relacionada, deve-se otimizar o algoritmo para maximizar o *recall*. Isso ocorre quando a ferramenta usando esse algoritmo quer construir uma lista com várias possíveis *hashtag* relacionadas para um humano avaliar e excluir as que não são realmente relacionadas. Nesse caso, não se quer deixar de fora nenhuma *hashtag* relacionada, mesmo que isso signifique capturar *tweets* que não são realmente relacionados ou precisar de uma análise manual para chegar ao conjunto final de *hashtags*.

Se o objetivo é minimizar os falsos positivos, para evitar selecionar *hashtags* que não são relacionadas, deve-se otimizar o algoritmo para maximizar a precisão. Esse é o caso quando a ferramenta que usa o algoritmo é completamente automática e sem interação

humana para descartar *hashtags* manualmente. Nesse caso, é melhor deixar de considerar algumas *hashtags* relacionadas, deixando de fora alguns *tweets* relacionados, do que considerar *tweets* que não são relacionados.

Quando ambas as métricas são importantes, mas uma delas é mais importante que a outra, pode-se usar F_b, onde b é o peso a ser ajustado. No caso estudado, precisão é mais importante do que *recall*, mas não se pode ignorar o *recall*. Por isso, foi escolhido maximizar o $F_{0.5}$, que corresponde à métrica *f-measure* dando um peso maior à precisão, como visto na Seção 3.4.3.

4.1.3 Implementação

Nesta seção, é apresentada a implementação do algoritmo de expansão de *hashtags*. O código desse algoritmo foi desenvolvido em *Python 2.7*, utilizando uma base de dados *MySQL* para armazenar os *tweets*, *hashtags* e usuários. Esses dados foram capturados utilizando a *API 140dev Streaming Twitter Framework* [1]. Essa *API* está disponível *online*, foi escrita em *PHP* e captura os *tweets* e seus dados relacionados utilizando a *API* do *Twitter*. Esses dados são então armazenados na base de dados. Essa base de dados é acessada pelo código *Python* para construir o conjunto de *hashtags* relacionadas. Todos os algoritmos apresentados neste trabalho utilizam essa arquitetura.

4.2 Estudo de Caso

Nesta seção, é apresentado um estudo de caso do algoritmo de expansão de *hashtags*. Na Seção 4.2.1, os dados utilizados no estudo de caso são descritos. Em seguida, na Seção 4.2.2, o treinamento do algoritmo para alcançar os melhores parâmetros é detalhado. Por fim, na Seção 4.2.3, os resultados obtidos são exibidos e analisados.

4.2.1 Dados

Os dados usados para treinar e testar o algoritmo consistem em *tweets* de quatro domínios diferentes: *iPhone 6*, ebola, Copa do Mundo 2014 e cigarros eletrônicos. Pode-se constatar que os tópicos são bem variados, incluindo produtos, eventos e doenças. Essa variedade de tópicos foi intencional e teve o objetivo de mostrar que o algoritmo pode ser adaptado para vários domínios diferentes.

[1] http://140dev.com

iPhone 6

O conjunto de dados sobre o *iPhone 6* foi capturado através da *API* do *Twitter* na semana de lançamento do produto pela *Apple*, que aconteceu em 09/09/2014. Os *tweets* foram coletados no período de 08/09/2014 a 13/09/2014. A quantidade de *tweets* capturados foi de 88.878. A *hashtag* utilizada para capturar os *tweets* foi #iPhone6. O domínio *iPhone 6* foi escolhido pois é um produto novo sendo lançado durante o desenvolvimento do projeto e é uma boa fonte de *tweets* com opiniões e sentimentos contrários.

Ebola

O conjunto de dados sobre ebola foi capturado no período de 08/09/2014 a 13/09/2014. A quantidade de *tweets* capturados foi de 17.688. A *hashtag* utilizada para capturar os *tweets* foi #ebola. Este domínio foi escolhido por causa do surto de ebola que estava acontecendo durante o desenvolvimento do projeto e a sua impoartância.

Copa do Mundo FIFA 2014

O conjunto de dados sobre a Copa do Mundo FIFA 2014 foi capturado utilizando a *API* do *Twitter* durante o evento, no período de 10/04/2014 a 14/04/2014. A quantidade de *tweets* na base de dados é de 6.063. A *hashtag* utilizada para capturar os *tweets* foi #worldCup2014. O domínio Copa do Mundo FIFA 2014 foi escolhido pelo impacto econômico e social que a Copa teria no Brasil.

Cigarros Eletrônicos

O conjunto de dados sobre cigarros eletrônicos foi capturado no período de 10/04/2014 a 14/04/2014. A quantidade de *tweets* na base de dados foi de 7.924. A *hashtag* utilizada para capturar os *tweets* foi #ecig. O domínio cigarros eletrônicos foi escolhido por ser um assunto extremamente controverso, que gera muita discussão no *Twitter*. Com a análise da opinião de uma quantidade significativa de *tweets*, pode-se concluir se há mais pessoas a favor ou contra o uso de cigarros eletrônicos.

4.2.2 Otimização

Neste algoritmo, vários parâmetros são customizáveis. O objetivo é testar o algoritmo com diferentes conjuntos de parâmetros e escolher o conjunto que proporciona os melhores resultados. Esses parâmetros são porcentagem de *tweets* com *hashtag*, custo de inserção, custo de deleção, custo de substituição, pontuação máxima normalizada, tamanho mínimo da *hashtag* e termos ignorados.

O primeiro parâmetro representa a porcentagem mínima de *tweets* com uma das *hashtags* semente que também possuem a *hashtag* analisada. Os custos de inserção, deleção e substituição são usados para calcular a distância de *Levenshtein*. A pontuação máxima normalizada significa o quanto uma *hashtag* deve ser similar a uma das *hashtags* ou termos semente para ser considerada uma *hashtag* relacionada. O tamanho mínimo da *hashtag* representa o tamanho mínimo que uma *hashtag* deve possuir para não ser ignorada. Finalmente, termos ignorados representa se são usados os termos ignorados no algoritmo.

Ajustou-se o algoritmo para testar várias combinações de parâmetros, ordenar os resultados pelo $F_{0,5}$ médio e escolher a melhor combinação de parâmetros. Os valores testados para cada parâmetro estão disponíveis na Tabela 4.1. Foram usados todas as combinações existentes, totalizando mais de 55.000 testes. O *ground-truth* foi construído manualmente através da análise de um banco de dados de *tweets* relacionados aos domínios escolhidos. Todas as *hashtags* encontradas no banco de dados foram analisadas e as *hashtags* relacionadas formaram o *ground-truth*.

Tabela 4.1: Valores testados para cada parâmetro

Número	**Parâmetro**	**Valores**
1	Percentagem de *Tweets* com a *Hashtag*	0.001, 0.01, 0.02, 0.03, 0.04, 0.05
2	Custo de Inserção	0.001, 0.005, 0.01, 0.02
3	Custo de Deleção	0.7, 0.8, 0.9, 1, 2, 3
4	Custo de Substituição	0.7, 0.8, 0.9, 1, 2, 3
5	Pontuação Máxima Normalizada	0.0001, 0.001, 0.01, 0.1
6	Tamanho Mínimo da *Hashtag*	1, 2, 3, 4, 5, 6, 7, 8
7	Termos Ignorados	Verdadeiro, Falso

Foram usados dois domínios (Copa do Mundo 2014 e ebola) para treinar o algoritmo e achar a melhor combinação de parâmetros. Os resultados obtidos nesse treinamento estão disponíveis na Tabela 4.2. Os valores de precisão, *recall* e $\mathbf{F_{0.5}}$ são os valores médios entre os resultados obtidos com os dois domínios. Pode ser visto no resultado que o parâmetro termos ignorados não fez diferença nas duas primeiras linhas da tabela, levando ao mesmo resultado usando ou não esses termos. Porém, analisando as linhas três e quatro da tabela, constata-se que não se pode ignorar esse parâmetro, visto que ele fez diferença nesses resultados e o melhor resultado foi alcançado usando os termos ignorados. Pode ser também observado que a pontuação máxima normalizada impactou em uma diminuição relativamente pequena entre os resultados das quatro primeiras linhas,

com o melhor resultado sendo alcançado com o parâmetro 5 setado como 0.0001. Porém, abaixo de 0.001 para esse parâmetro, os resultados já são muito inferiores, como se pode ver nas linhas cinco e seis. Além disso, uma alteração nos outros parâmetros levou a resultados muito inferiores, sendo conveniente usar os valores que chegaram ao melhor resultado.

Tabela 4.2: Melhores Resultados no Treinamento

1	**2**	**3**	**4**	**5**	**6**	**7**	**Precisão**	***Recall***	$\mathbf{F_{0.5}}$
0.001	**0.001**	**0.8**	**0.8**	**0.0001**	**6**	**True**	**1.00000**	**0.79977**	**0.94631**
0.001	**0.001**	**0.8**	**0.8**	**0.0001**	**6**	**False**	**1.00000**	**0.79977**	**0.94631**
0.001	0.001	0.8	0.8	0.001	6	True	0.81250	0.90847	0.82563
0.001	0.001	0.8	0.8	0.001	6	False	0.75000	0.95194	0.77089
0.001	0.001	0.8	0.8	0.01	6	True	0.02996	0.93898	0.03714
0.001	0.001	0.8	0.8	0.01	6	False	0.02993	0.98246	0.03711

4.2.3 Avaliação e Resultados

Na seção anterior, foram mostradas a otimização e treinamento do algoritmo e o conjunto de parâmetros que levou aos melhores resultados no conjunto de treinamento. Em seguida, esses parâmetros foram aplicados aos domínios de teste (*iPhone 6* e cigarros eletrônicos) para comparação dos resultados. A Tabela 4.3 mostra os resultados para os domínios de treinamento e a Tabela 4.4 mostra os resultados para os domínios de teste.

Pode-se ver que os resultados são consistentes, uma vez que o conjunto de treinamento produz resultados melhores do que o conjunto de testes e ambos possuem um bom $F_{0.5}$. Por exemplo, na Tabela 4.3, o resultado para o domínio ebola obteve uma precisão de 100% e um *recall* de mais de 94%. Já para o domínio de cigarros eletrônicos, exibido na Tabela 4.4, foi obtida uma precisão de 97% e um *recall* de 97%. Com isso, pode-se constatar que esse algoritmo pode ser aplicado facilmente a outros domínios, visto que ele obteve bons resultados em domínios completamente diferentes dos domínios de treinamento.

4.3 Trabalhos Futuros

Esse algoritmo foca na estrutura sintática das *hashtags*. Para melhorar o algoritmo e excluir a necessidade de prover os termos relacionados, o algoritmo poderia considerar a

Tabela 4.3: Expansão de *Hashtags* - Resultados com conjunto de treinamento

Domínio	**Precisão**	***Recall***	$\mathbf{F_{0.5}}$
#worldCup2014	1.00000	0.65217	0.90361
#ebola	1.00000	0.94737	0.98901
Média	**1.00000**	**0.79977**	**0.94631**

Tabela 4.4: Expansão de *Hashtags* - Resultados com conjunto de teste

Domínio	**Precisão**	***Recall***	$\mathbf{F_{0.5}}$
#iphone6	0.76470	0.97500	0.79918
#ecig	0.97142	0.97842	0.97282
Média	**0.86806**	**0.97671**	**0.88600**

estrutura dos *tweets* e as informações semânticas sobre *tweets* e termos. Uma ideia similar ao artigo [44], que encontra os termos mais usados em *tweets* sobre um determinado tópico, pode ser usada para alcançar isso. Outra abordagem interessante é aplicar um algoritmo de agrupamento às *hashtags* co-ocorrentes mais comuns e encontrar os maiores agrupamentos. Esses agrupamentos são provavelmente relacionados ao tópico especificado.

Usando informação semântica, uma possível abordagem seria levar em consideração uma distância como *Google Distance*, que usa a co-ocorrência de termos na web. Essa abordagem não foi usada pela falta de uma ferramenta gratuita e e com acesso ilimitado para prover os resultados de busca necessários para calcular essa distância, mas isso pode mudar em um futuro próximo. Outra ideia seria considerar a descrição das *hashtags*. Essas descrições estão disponíveis *online* em alguns *sites*, como TagDef [2]. A maioria desses *sites* utiliza descrições escritas de forma colaborativa pelos usuários. O problema de usar essas ferramentas atualmente é que elas não são completas o suficiente, pois faltam as descrições de muitas *hashtags* importantes, principalmente as mais novas. Finalmente, pode-se usar informação semântica proporcionada pela DBPedia [3], WordNet [4] ou outra ferramenta similar. O problema também estaria nos termos mais novos, como produtos recém-lançados, que ainda não foram cadastrados.

Por fim, o algoritmo de expansão de *hashtags* apresenta bons resultados, com precisão média superior a 90%. Um pequeno incoveniente é a necessidade de o usuário fornecer três

[2]https://tagdef.com/
[3]http://dbpedia.org
[4]http://wordnet.princeton.edu/

conjuntos como entrada. Melhorias neste algoritmo não almejam primariamente melhorar esses resultados, mas diminuir a quantidade de entradas exigidas pelo algoritmo.

Capítulo 5

Detecção de *Spam*

Neste capítulo, é discutido o problema de selecionar *tweets* relevantes dentre aqueles capturados no capítulo anterior. Para fazer isso, é preciso excluir os *tweets* que são *spam*, tais como propagandas, requisições para ser seguido e *tweets* gerados automaticamente (ex: os gerados por jogos).

Os trabalhos que focam em *spam* no *Twitter* são divididos em dois grupos: os que focam em detectar os *tweets* que são *spam* e os que focam em detectar os usuários que publicam *spam* (*spammers*). O presente trabalho foca na primeira opção, já que está tratando um *stream de tweets*. No segundo caso, os algoritmos normalmente analisam vários *tweets* do mesmo usuário e as características do usuário para decidir se a conta é de um *spammer*. Isso não se adequa ao caso atual, visto que se quer analisar vários *tweets* de usuários diferentes sobre um mesmo assunto e detectar quais mensagens são *spam*.

Neste trabalho, a definição oficial de *spam* do *Twitter* é utilizada. Segundo o *Twitter*, *spam* refere-se a uma variedade de comportamentos proibidos que violam as Regras do *Twitter*. *Spam* pode ser descrito como ações repetitivas e não solicitadas que impactam negativamente em outros usuários. Isso inclui várias formas automáticas de interações e comportamentos bem como tentativas de enganar usuários. Como exemplos desse tipo de comportamento, pode-se citar: postar *links* maliciosos, abusar de menções e respostas para postar conteúdo não desejado, postar repetidamente em *trending topics* para chamar atenção, postar repetidamente a mesma mensagem e postar *links* não relacionados ao conteúdo do *tweet*.

Com essa definição em mente, foi construído o Algoritmo de Detecção de *Spam*, apresentado na Seção 5.1. A Seção 5.2 mostra um estudo de caso do algoritmo. Por fim, a Seção 5.3 discute os trabalhos futuros a respeito dessa etapa da ferramenta.

5.1 Algoritmo de Detecção de *Spam*

Algoritmos que focam em identificar *spammers* normalmente levam em consideração características da conta do usuário e o histórico de *tweets* publicados por ele ([3], [60], [27], [13]). Dentre essas características da conta do usuário, pode-se citar a idade da conta de usuário, o número de *followers* e *followees* e a quantidade de *tweets* publicados por dia. Algoritmos que focam em identificar *tweets* que são *spam* analisam o *tweet* separadamente. Esses algoritmos normalmente usam características do *tweet*, tais como o número de *hashtags* e URLs. Algumas vezes, essas informações são combinadas com características do usuário que postou a mensagem.

Com isso em mente, foram comparadas características de *tweets* e usuários apresentadas nas Tabelas 2.2 e 2.3 e escolhidas as características mais usadas que podem ser aplicadas ao algoritmo, identificando uma combinação que levasse aos melhores resultados. Dentre as mais usadas, não foi considerada a média de URLs por *tweet*, pois é uma informação que não é possível obter analisando os dados estáticos da conta do usuário.

As características escolhidas foram consolidadas em um algoritmo e incluem: idade de conta em dias, número de *followers*, número de URLs, número de *hashtags*, número de *spam* words e se uma das URLs do *tweet* está em uma lista negra de URLs. O número de *spam words* é muito usado para detectar *spam* em emails e também pode ser aplicado a *tweets*. Foi consolidada uma lista de *spam words* incluindo palavras que são consideradas *spam* em vários contextos e palavras que são consideradas *spam* no *Twitter*, tais como '*follow*' e '*please*'. Essas palavras normalmente indicam um usuário pedindo para ser seguido.

Outra característica escolhida foi a idade da conta em dias. O motivo é que contas de *spammers* normalmente têm um período de vida curto, visto que elas são muitas vezes identificadas como *spammers* após poucos dias e excluídas do *Twitter*. O número de seguidores também é uma característica significante, visto que *spammers* normalmente têm menos seguidores do que uma conta normal. O número de URLs no *tweet* foi escolhido porque *spammers* normalmente postam *tweets* com URLs. Isso acontece porque essas mensagens têm tamanho máximo de 140 caracteres, o que normalmente não é suficiente para disseminar a informação pretendida, que são postadas como URLs. Muitas vezes eles postam uma URL não relacionada ao conteúdo do *tweet*, enganando o usuário a clicar no link malicioso. Relacionado a isso, outra característica escolhida é se o *tweet* contém uma URL que está em uma lista negra de URLs. Foi usada a *PhishTank blacklist* [1] para fazer essa verificação. Por fim, a última característica analisada é o número de *hashtags* do

[1] http://www.phishtank.com/

tweet, que é importante pois *spammers* normalmente usam muitas *hashtags* para aparecer em buscas e nos *trending topics*.

5.1.1 Pseudo-código do Algoritmo

Algoritmo 2: Algoritmo de Detecção de *Spam*

```
   Data: tweet, parameters
   Result: isSpam
   /* Retorna se o tweet é spam                                          */
 1 if tweet.user.accountAge <= parameters[0] then
 2 |   return true;
 3 end
 4 if tweet.user.followersCount <= parameters[1] then
 5 |   return true;
 6 end
 7 if tweet.hashtagsCount <= parameters[2] then
 8 |   return true;
 9 end
10 if tweet.spamWordsCount <= parameters[3] then
11 |   return true;
12 end
13 if tweet.urlCount <= parameters[4] then
14 |   return true;
15 end
16 if tweet.containsUrlBlacklisted() AND parameters[5] then
17 |   return true;
18 end
19 return false;
```

O Algoritmo 2 mostra o pseudo-código do algoritmo de detecção de *spam*. Ele recebe como parâmetro um *tweet* e um array de parâmetros e retorna se o *tweet* é *spam*. Para cada característica do *tweet* ou do usuário explicada na seção anterior, é estabelecido um limiar no array de parâmetros. Cada característica é verificada para constatar se é menor ou igual ao limiar e, caso positivo, o *tweet* é considerado *spam*. Por fim, é verificado se o *tweet* contém uma URL pertencente a uma lista negra. Se o *tweet* passar por todos os testes, ele não é considerado *spam*.

5.1.2 Métricas

Nesta seção, as métricas usadas para comparar o desempenho do algoritmo são apresentadas.

Precisão

Na Seção 3.4.3, foram vistas a definição e a fórmula para o cálculo da precisão. Adaptando para o algoritmo de detecção de *spam*, precisão é a fração de *tweets* selecionados que são realmente *spam*, como visto na Equação 4.2.

$$\text{Precisão} = \frac{\text{número de } \textit{spam} \text{ encontrados pelo algoritmo}}{\text{número de } \textit{tweets} \text{ considerados pelo algoritmo como } \textit{spam}} \tag{5.1}$$

Se não há falsos positivos, todos os *tweets* selecionados pelo algoritmo são realmente *spam*, alcançando uma precisão de 1. Se metade dos *tweets* selecionados são *spam*, a precisão é 0.5. Com isso, pode-se constatar que maximizar a precisão significa minimizar os falsos positivos, o que significa que poucos *tweets* selecionadas pelo algoritmo não são realmente *spam*.

Maximizar a precisão garante que a maior parte dos *tweets* capturados são realmente *spam*, mesmo que alguns *tweets* que são *spam* deixem de ser capturados.

Recall

Na Seção 3.4.3, foram vistas a definição e a fórmula para o cálculo do *recall*. Adaptando para o algoritmo de detecção de *tweets*, *recall* é a fração de *spam* que são encontrados pelo algoritmo.

$$\textit{Recall} = \frac{\text{número de } \textit{tweets} \text{ que são } \textit{spam} \text{ encontrados pelo algoritmo}}{\text{número de } \textit{tweets} \text{ que são } \textit{spam}} \tag{5.2}$$

Se não tiver nenhum falso negativo, todos os *tweets* que são *spam* foram capturadas pelo algoritmo, derivando um *recall* de 1. Se metade dos *spam* foram encontrados, o *recall* é 0.5. Com isso, pode ser visto que maximizar o *recall* significa minimizar os falsos negativos.

Maximizar o *recall* garante que a maior parte dos *tweets* que são *spam* são capturados, mesmo que alguns *tweets* que não são *spam* também sejam.

Otimização

Ao otimizar um algoritmo, a métrica utilizada como referência é importante. No caso atual, os falsos negativos devem ser minimizados. Isso se deve ao fato de que o impacto de ignorar alguns *tweets* que não são *spams* não é crítico, visto que há um número grande de *tweets* para usar. Por outro lado, o impacto de selecionar um *tweet* que é *spam* e usá-lo para construir o léxico é significativo. Por isso, o F_2 deve ser maximizado. Essa métrica se refere ao *f-measure* dando um peso maior ao *recall*, conforme visto na Seção 3.4.3.

5.1.3 Implementação

O código desse algoritmo foi construído em *Python*, usando a mesma arquitetura apresentada na Seção 4.1.3.

Tabela 5.1: Exemplos de *tweets* que não são *spam*

So far very disappointed with the #iPhone6Plus #iPhone6. Come on #Apple, step up your game. I'm not that impressed yet.
Once upon a time #apple was laughing at big screens, said it is for Neanderthals. Now with #iPhone6Plus lol #iPhone6
Well, I'm glad I saved my upgrade for the launch of the #iphone6 #iphone6plus but which one to choose? #decisionsdecisions #applelove
Ahhhh, I hope I can make the preorders for the iPhone 6 on Friday!! #iphone #iPhone6 #AppleEvent #hype
I want one #iPhone6Plus ...and of course I will need to accessorize #iWatch

5.2 Estudo de Caso

Nesta seção, um estudo de caso do algoritmo de detecção de *spam* proposto é apresentado, incluindo os dados utilizados, a otimização realizada para alcançar os parâmetros ideais e os resultados obtidos nos conjuntos de treinamento e teste.

5.2.1 Dados

O conjunto de dados utilizado consiste em 3.500 *tweets* sobre o *iPhone 6* que foram selecionados na etapa anterior. Esses *tweets* foram manualmente rotulados como *spam* ou não *spam*, considerando a definição de *spam* dada pelo *Twitter*, para construir o *ground truth*.

A quantidade de *spam* encontrada nos *tweets* sobre o *iPhone 6* foi bastante alta. No conjunto de dados rotulado manualmente, 49.4% dos *tweets* foram considerados *spam*. Acredita-se que essa quantidade de *spams* se deve ao fato de o *iPhone 6* ter sido lançado em 09/09/2014 e estar presente nos *trending topics* durante o período da coleta dos *tweets*, o que atrai *spammers*.

As Tabelas 5.1 e 5.2 mostram exemplos de *tweets* que são *spam* e que não são *spam*. Para os *spams*, também é exibido o motivo pelo qual eles foram considerados *spam*.

Tabela 5.2: Examplos de *tweets* que são *spam*

Tweets* que são *spam	**Motivo**
NEW MUSIC: #Freestyle with my bro Iyanya https://t.co/jKmcr7r8Ij #iPhone #iPhone6 #Apple http://t.co/Dixrb7E9Ev	Falando sobre um tópico diferente das *hashtags*
Top 5 Free iOS Apps of August 2014 http://t.co/mIq82S38gC #ipad #AppleLive #mac #apple #games #reviews #iphone6 #ios8 #iphone	Postando links com conteúdo não-relacionado
Buying #iPhone 6? #Amazon's Trade-In Program is right place for old phones http://t.co/RfeoVJmtGx #iPhone6 #Apple #iWatch #iOS8	Postando *tweets* repetidos
#news #Apple's launch event is tomorrow, 10 am PT. We'll be repoing live #iWatch #iPhone #iOS http://t.co/CbMasmJNY2	Postando *tweets* repetidos
#Iphone Therefore I AM Long Sleeve #TShirt http://t.co/OCZXfdkwKj @iPhoneTeam #Iphone6 #Gift #in #Fab #fashion	Postando *tweets* repetidos
Amazing #photo recovery results in DoctorPhoto #iOSapp! #iOSlove #iOS8 #iPhone6 #iPhone6Plus https://t.co/qzmSuwJ6n9 http://t.co/KnHVzVc4pN	Postando *tweets* repetidos

5.2.2 Otimização

Nesse algoritmo, vários parâmetros são customizáveis. O algoritmo foi testado com vários conjuntos diferentes de parâmetros para escolher o conjunto que leva aos melhores resultados. Esses parâmetros foram detalhados na Seção 5.1 e referem-se a um limiar para cada característica escolhida. Os valores testados para cada parâmetro estão disponíveis na Tabela 5.3. Os valores marcados como -1 significam que o parâmetro não foi levado em consideração pelo algoritmo. Todas as combinações foram testadas, resultando em 7.500 combinações. A métrica utilizada para comparação dos resultados foi F_2. Foram usados 3.000 *tweets* do conjunto de dados para treinar o algoritmo de detecção de *spam*.

Os resultados obtidos com o treinamento podem ser vistos na Tabela 5.4. Desses resultados, pode ser visto que idade da conta e número de *spam words* não são significativos para o resultado, pois os melhores resultados foram alcançados quando esses parâmetros não estavam sendo considerados, ou seja, eram iguais a -1. Pode ser observado que o número de *hashtags* é uma característica significativa e usada para identificar *spam* nesse domínio. Acredita-se que a razão é que *spammers* normalmente postam *tweets* com muitas *hashtags*, para aumentar a chance de aparecerem em buscas e *trending topics*. Além

Tabela 5.3: Valores testados para cada parâmetro

Número	**Parâmetro**	**Valores**
1	Idade da Conta	-1, 2, 5, 10, 15, 20, 25, 30
2	Quantidade de Seguidores	-1, 2, 5, 10, 15, 20, 25, 30
3	Número de URLs	-1, 1, 2, 3, 4
4	Número de *Hashtags*	-1, 2, 4, 6, 8, 10
5	Número de *Spam Words*	-1, 1, 2, 4, 6
6	Lista Negra de URLs	True, False

disso, a quantidade de seguidores, o número de URLs e a lista negra de URLs também são características significativas, visto que desconsiderar essas características piora os resultados.

Tabela 5.4: Melhores Resultados no Treinamento - *iPhone 6*

1	**2**	**3**	**4**	**5**	**6**	**Precisão**	***Recall***	$\mathbf{F_2}$
-1	**15**	**1**	**8**	**-1**	**True**	**0.72014**	**0.99394**	**0.92370**
-1	25	1	8	-1	True	0.70636	0.99394	0.91910
-1	15	1	4	-1	True	0.59286	0.99462	0.87590
-1	5	4	4	1	True	0.63991	0.93876	0.85857
-1	10	2	4	2	True	0.63529	0.94011	0.85779
5	-1	-1	4	-1	False	0.64425	0.93472	0.85741
2	-1	3	4	2	False	0.64425	0.93472	0.85741
15	-1	4	4	4	False	0.64425	0.93472	0.85741
10	-1	4	4	6	False	0.64425	0.93472	0.85741
5	-1	4	4	-1	False	0.64425	0.93472	0.85741
-1	2	4	4	1	True	0.64216	0.93472	0.85667
-1	-1	4	2	-1	False	0.49352	1.00000	0.82970
-1	-1	4	6	-1	False	0.19614	0.04105	0.04876

5.2.3 Avaliação e Resultados

Foram usados 3.000 *tweets* para treinar o algoritmo de detecção de *spam* no domínio *iPhone 6* e chegar ao melhor conjunto de parâmetros. Outros 500 *tweets* foram utilizados

para testar o algoritmo. Os resultados podem ser visualizados na Tabela 5.5. O conjunto de treinamento e testes possuem resultados similares, o que significa que o treinamento realizado foi consistente. Além disso, o *recall* alcançado no treinamento foi superior a 99% e no teste foi superior a 98%, o que significa que o algoritmo é capaz de capturar a maior parte dos *spams*.

Tabela 5.5: Resultados do treinamento e teste - *iPhone 6*

Conjunto de Dados	**Precisão**	***Recall***	$\mathbf{F_2}$
Treinamento	0.72014	0.99394	0.92370
Teste	0.69602	0.98790	0.91146

5.3 Trabalhos Futuros

O algoritmo de detecção de *spams* utiliza características que chegaram aos melhores resultados em estudos anteriores e alcança um resultado otimizado, permitindo excluir *spams* com *recall* superior a 98%. Com isso, a maior parte dos *spams* podem ser excluídos, o que é crucial para evitar que o *spam* passe para as próximas etapas.

Esse algoritmo considera cada característica dos *tweets* separadamente. Uma possível melhoria é aplicar um algoritmo de aprendizagem de máquina ao problema usando as mesmas características propostas aqui. O algoritmo de aprendizagem de máquina iria considerar todas as características de forma conjunta, o que pode levar a melhores resultados.

Pela observação dos *spams* durante o rotulamento manual dos *tweets*, foi observado que há muitos *spams* que podem ser classificados como mensagens iguais postadas várias vezes. Com isso em mente, uma possível melhoria é contar o número de *tweets* encontrados que possuem exatamente o mesmo texto e excluir os *tweets* que possuem muitas repetições.

Capítulo 6

Construção de Léxicos de Sentimento

Este capítulo apresenta a terceira etapa da ferramenta, que é responsável por construir um léxico de sentimentos para *tweets* de domínio específico. Isso é necessário para aprimorar o desempenho da ferramenta de análise de sentimento de *tweets*. Como *tweets* são textos curtos e informais e contêm abreviações e gírias, um léxico de sentimento tradicional construído para analisar textos formais não é adequado. Isso acontece porque léxicos de sentimento tradicionais são normalmente baseados em um dicionário ou um conjunto de textos formais e não incorporam características de textos informais, como gírias. Esses léxicos formais também não incorporam características específicas de *tweets*, como menções, URLs, *hashtags* e *emoticons*. Acredita-se que essas características, especialmente *emoticons*, são bastante significativas para extrair o sentimento dos *tweets*. Por isso, para conseguir capturar as peculiaridades dessas mensagens adequadamente, é importante construir um léxico específico para *tweets*.

De acordo com Liu [31], o sentimento de várias palavras varia de acordo com o domínio em que se encontra, então construir um léxico de sentimento genérico não irá capturar as especificidades de cada domínio. Por exemplo, a palavra '*loud*' (em português, alto ou barulhento) é considerada positiva quando se está falando de um aparelho de som, mas é negativa quando se refere a uma geladeira. Para melhorar a performance da análise de sentimento, é preciso construir um léxico que é específico para o domínio sendo avaliado. Por fim, para extrair as melhores características das duas abordagens para criar léxicos apresentadas aqui, a proposta é construir um léxico que é específico tanto para *tweets* quanto para o domínio escolhido.

Um léxico pode ser construído usando uma das duas abordagens: supervisionada e não-supervisionada. Foi escolhida uma abordagem não-supervisionada e baseada em corpus por causa da facilidade de aplicá-la a outros domínios. A entrada do algoritmo consiste em um conjunto de *tweets* sobre o domínio e dois conjunto de *hashtags* e *emoticons* independentes de domínio: um positivo e um negativo. Esses conjuntos independentes de

domínio podem ser criados apenas uma vez e reusados para vários domínios diferentes.

O algoritmo proposto é baseado no algoritmo de propagação de grafos proposto por Velikovich [64]. Nesse artigo, os autores aplicaram esse algoritmo em um grafo construído através de frases extraídas da web. Uma abordagem diferente para a construção do grafo é proposta, usando co-ocorrência e similaridade do cosseno entre termos encontrados nos *tweets*.

A Seção 6.1 apresenta uma descrição formal da solução proposta. A Seção 6.2 se refere ao algoritmo de construção de léxico. A Seção 6.3 mostra um estudo de caso para o algoritmo. Por fim, a Seção 6.4 discute os trabalhos futuros a respeito dessa etapa da ferramenta.

6.1 Descrição Formal da Solução Proposta

Para resolver o problema da criação do léxico de sentimento, foi proposta uma solução baseada em propagação de grafos. Essa seção tem como objetivo apresentar o embasamento matemático dessa solução.

Como visto na Seção 3.1.4, uma solução para resolver o problema de criar um léxico de sentimento deve receber como entrada um conjunto de *tweets* T e retornar um conjunto de tuplas $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, ...\}$, onde $\gamma_x = (w, s)$. A solução proposta consiste em criar um grafo ponderado e não-direcionado G a partir de T, onde G encapsula informações semânticas sobre os *tweets* de T. Em seguida, esse grafo é utilizado em um algoritmo de propagação de grafos, que expande as relações encontradas no grafo e retorna o conjunto de tuplas Γ referente ao léxico criado.

O grafo criado $G = (V, E)$ é ponderado e não-direcionado, onde o conjunto de nós V é o conjunto de *tokens* candidatos a fazer parte do léxico, ou seja, $V = TT = \cup tok(t), \forall t \in T$. O conjunto de arestas E consiste em todos os pares de arestas possíveis entre os nós de V, ou seja, $E = \cup\{(v_i, v_j)\}$, $\forall i, j \in V$.

O peso w_{ij} da aresta $(v_i, v_j) \in E$ entre v_i e v_j é tal que $w_{ij} \in [0, 1]$ e deve codificar a similaridade semântica entre esses nós. Por exemplo, $v_i = amazing$, $v_j = good$ e $v_k = disappointing$. A similaridade semântica entre '*amazing*' e '*good*' é maior do que entre '*amazing*' e '*disappointing*', então deve-se esperar que $w_{ij} > w_{ik}$.

O grafo G é um grafo de similaridade de cossenos, onde $w_{ij} = similarity(V_i, V_j)$. V_x pode ser definido como o vetor que indica a presença de $x \in TT$ em cada $t \in T$, ou seja, $V_x = (v_{x1}, v_{x2}, v_{x3}, ...)$ é um vetor de tamanho $|TT|$ e $v_{xi} = 0$ se $tt_x \notin t_i$ e $v_{xi} = 1$ se

$tt_x \in t_i$. A função *similarity* pode ser calculada através da Equação 6.1.

$$similarity(A, B) = \frac{A.B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{6.1}$$

De posse do grafo de similaridade de cossenos G, será feito uma propagação nesse grafo até obter o conjunto de tuplas Γ. Uma propriedade comum entre algoritmos de propagação de grafos é que eles tentam propagar as informações dos nós semente para o resto do grafo através de suas arestas. Como o objetivo é construir um léxico de sentimentos que estabelece se cada *token* é positivo ou negativo, é necessário definir nós sementes positivos e negativos para serem usados na propagação.

Seja P o conjunto semente de *tokens* positivos e N o conjunto semente de *tokens* negativos. O valor $pol+_w$ associado a cada nó v_w é igual à soma dos caminhos máximos de cada nó semente positivo até o nó v_w. O valor $pol-_w$ associado a cada nó v_w é igual à soma dos caminhos máximos de cada nó semente negativo até o nó v_w. O valor s da tupla $\gamma = (w, s)$ é igual à soma das polaridades positivas e negativas desse *token*, ou seja, $s = pol +_w + pol-_w$.

O conjunto de tuplas Γ resultante é a união das tuplas geradas pelos nós de V e seus respectivos valores de polaridade s. Esse conjunto é o léxico criado pela solução proposta. Um algoritmo para implementar essa abordagem é apresentado na Seção 6.2.

6.2 Algoritmo de Construção de Léxico

O algoritmo de construção de léxico consiste de quatro etapas: *tokenizador*, construção do grafo, propagação do grafo e consolidação do léxico. Cada uma das etapas é detalhada nas próximas seções.

6.2.1 *Tokenizador*

Tokenização é o processo de dividir uma *string* nas suas partes constituintes e é uma técnica fundamental para várias tarefas de processamento de linguagem natural. Cada uma dessas tarefas pode possuir uma necessidade de dividir as *strings* de uma forma diferente, dependendo da aplicação e do domínio. Construir um *tokenizador* específico para *tweets* é importante, pois essas mensagens possuem muitas particularidades no seu texto, como menções, *hashtags*, URLs e *emoticons*.

No *tokenizador* proposto, todas as menções existentes são substituídas por '*@mention*' e todas as URLs por '*www.url.com*'. Isso foi feito considerando que cada menção a um

usuário particular não tem muito significado para o sentimento daquela mensagem. Além disso, considerar todas as menções individualmente aumentaria bastante o tamanho do conjunto de termos extraídos dos *tweets*. A mesma justificativa foi usada para substituir as URLs. Os *emoticons* e as *hashtags* não devem ser substituídos, pois constituem um importante fator para avaliar o sentimento do *tweet*. Por isso, eles são mantidos sem modificação e considerados como um *token* isolado. Para ilustrar o comportamento do léxico proposto, exemplos de *tweets* e suas versões *tokenizadas* são mostrados na Tabela 6.1.

Além das características específicas de *tweets*, o *tokenizador* também lida com negação e *elongação*. Todas as palavras que se encontram entre uma palavra de negação e um sinal de pontuação são marcados com '_NEG' no final. Essa marcação significa que esse *token* tem o significado oposto à palavra original. Por exemplo, o segundo *tweet* apresentado na Tabela 6.1, as palavras '*love*' e '*anymore*' estão marcadas com o símbolo de negação. Isso significa que, se '*love*' é considerada uma palavra positiva, '*love_NEG*' é considerada uma palavra negativa. O problema da negação da negação, que transforma a palavra de volta ao seu significado original, também é considerado. Por exemplo, na frase '*I never said I didn't go*', a segunda negação (*didn't*) neutraliza a primeira negação (*never*). Por isso, não se pode considerar a palavra '*go*' como negada, pois ela está depois da segunda negação. Por outro lado, a palavra '*said*' continua negada, pois ela está no escopo da primeira negação. A partir dessa explicação, pode-se concluir que uma palavra é marcada como negada se ela estiver entre uma palavra de negação e um sinal de pontuação ou se ela estiver entre uma palavra de negação e outra palavra de negação. Esse comportamento pode ser visualizado no quarto exemplo da Tabela 6.1.

Elongação consiste em repetir um caractere de uma palavra várias vezes para indicar uma emoção intensificada. Por exemplo, '*cooooooool*' dá ênfase à palavra '*cool*' (legal, em português). O uso de *elongação* é muito comum em *tweets* e outros textos informais. O algoritmo lida com isso marcando a palavra com '_LONG' no final. Esse comportamento pode ser observado no segundo exemplo na Tabela 6.1.

Stop words são palavras que aparecem com muita frequência em textos, como '*the*', '*it*' e '*to*'. Essas palavras são normalmente filtradas dos documentos em tarefas de processamento de linguagem natural. *Stop words* geralmente possuem pouco conteúdo léxico associado e a sua presença no texto não influencia no sentimento do mesmo, na grande maioria dos casos. Por isso, as *stop words* são excluídas durante o processo de *tokenização*. Esse procedimento pode ser observado no primeiro exemplo na Tabela 6.1. As palavras '*this*', '*is*' e '*a*' são consideradas *stop words* e excluídas da versão *tokenizada*. Pode-se observar que as palavras que foram excluídas realmente não influenciam no sentimento do *tweet*.

Além disso, números de telefone também são identificados e mantidos inalterados, não perdendo a sua pontuação de separação dos números, como pode ser visto no exemplo 5 da Tabela 6.1. O *tokenizador* exclui toda a pontuação restante do texto e transforma todas as palavras em caixa baixa (usa apenas letras minúsculas).

Tabela 6.1: Exemplos de *tokenização* de *tweets*

Tweet	***Tokenização***
RT @patlustosa: this is a typical Twitter tweet :-) #ICWSM	rt, @mention, typical, twitter, tweet, :-), #icswm
I don't love him anymore. Sooooo #sad :'(o.O	love_NEG, anymore_NEG, so_LONG, #sad, :'(, o.O
Let's just compare them #iPhone6 #iPhone6Plus http://t.co/Zj3b3wfp30	let's, compare, #iphone6, #iphone6plus, http://url.com
I didn't say I wouldn't buy it.	say_NEG, wouldn't, buy
It's perhaps noteworthy that phone numbers like +1 (000) 123-4567 are treated as words	perhaps, noteworthy, phone, numbers, like, +1 (000) 123-4567, treated, words

6.2.2 Construindo o Grafo a Partir de *Tweets*

A ideia é construir um grafo que atenda as restrições descritas na Seção 6.1 fazendo o uso de *tweets* de domínio específico. Para isso, o *tokenizador* é executado com todos os *tweets* e o conjunto de todos os *tokens* encontrados nos *tweets* são considerados *tokens* candidatos. Considere T como o conjunto de todos os *tweets* recebidos como entrada e $tok(t)$ como o conjunto de *tokens* retornado pelo *tokenizador* com t passado como parâmetro. Então, $E = \cup tok(t), \forall t \in T$, onde E é o conjunto de nós do grafo.

A matriz com a contagem de co-ocorrências entre todos os $e \in E$ é computada. Essa matriz é então convertida em uma matriz de similaridade de cossenos chamada CM. O peso da aresta entre v_i e v_j é a similaridade de cossenos entre os dois *tokens*, ou seja, $w_{ij} = CM[i][j]$.

O pseudo-código desse algoritmo pode ser visto no Algoritmo 3.

Exemplo

Um exemplo concreto da criação do grafo é demonstrado neste exemplo. A Tabela 6.2 mostra o corpus de textos usados para a criação do léxico.

A matriz de co-ocorrência pode ser visualizada como a Tabela 6.3. Por exemplo, as palavras *beautiful* e *song* têm contagem de co-ocorrência igual a cinco porque elas

Algoritmo 3: Algoritmo de Construção do Grafo

Data: *corpus*
Result: *csMatrix*
/* *corpus* é o conjunto de *tokens* dos *tweets* */
/* Retorna um grafo representado como uma matriz */
1 d =buildCooccurenceMatrix(*corpus*);
2 *vocab* =getSortedVocab(d);
3 *csMatrix* =buildCosineSimilarityMatrix(*vocab*, d);
4 return *csMatrix*;

Tabela 6.2: Corpus

beautiful surprising
beautiful song
beautiful song
beautiful song
beautiful song
beautiful song
surprising song
surprising song
cool beautiful

aparecem juntas em cinco entradas do corpus. Já a matriz de similaridade de cossenos pode ser visualizada como a Tabela 6.4.

Tabela 6.3: Matriz de Co-ocorrência

	beautiful	***cool***	***song***	***surprising***
beautiful	0	1	5	1
cool	1	0	0	0
song	5	0	0	2
surprising	1	0	2	0

O grafo pode ser visualizado na Figura 6.1.

6.2.3 Algoritmo de Propagação do Grafo

O algoritmo de propagação do grafo utiliza o grafo gerado na seção anterior. Esse grafo deve codificar a similaridade semântica entre os nós. Uma propriedade comum desse tipo de algoritmo é que eles tentam propagar informações dos nós semente para o resto do

Tabela 6.4: Matriz de Similaridade de Cossenos

	beautiful	***cool***	***song***	***surprising***
beautiful	1.0	0.0	0.07	0.86
cool	0.0	1.0	0.93	0.45
song	0.07	0.93	1.0	0.42
surprising	0.86	0.45	0.42	1.0

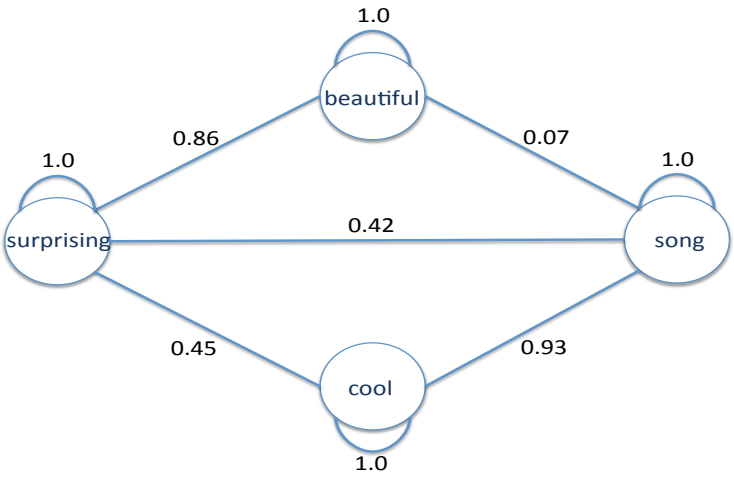

Figura 6.1: Grafo de Exemplo

grafo através das suas arestas. A saída do algoritmo é o valor da polaridaridade de cada nó do grafo.

O pseudo-código do algoritmo de propagação do grafo é exibido no Algoritmo 4. O conjunto de nós do grafo corresponde aos *tokens* dos *tweets*. O valor da polaridade de um nó é igual à soma dos caminhos de peso máximo entre cada palavra semente até esse nó. *Tokens* que são conectados a múltiplas palavras semente positivas com caminhos de alto peso são considerados positivos. Se o fluxo for maior para palavras semente negativas, o nó é considerado negativo. A variável b é usada para manter o equilíbrio entre os valores, caso haja diferença entre o fluxo positivo e negativo no grafo. T é o tamanho do maior caminho considerado pelo algoritmo e o número de iterações que o *loop* da linha 6 do algoritmo faz. l é o limiar que define a polaridade mínima que o *token* deve possuir para ser adicionado ao léxico. Por fim, a polaridade de cada nó é adicionada ao léxico e retornada pelo algoritmo.

Algoritmo 4: Algoritmo de Propagação de Grafo

Data: $G = (V, E), w_{ij} \in [0, 1], P, N, l, T$
Result: $map(V, pol)$
`/* Retorna o valor da polaridade de cada token */`
$polPos, polNeg, lexicon = map(V, pol)$;
set $a_{ii} = 1$ for i;
set $a_{ij} = 0$ for $i \neq j$;
for $v_i \in P$ **do**
  $F = \{v_i\}$;
  for t *from* 1 *to* T **do**
    for $(v_k, v_j) \in E$ *such that* $v_k \in F$ **do**
      $a_{ij} = max\{a_{ij}, a_{ik} * w_{kj}\}$;
      $F = F \cup \{v_j\}$;
    end
  end
end
for $v_j \in V$ **do**
  $polPos[j] = sum(a_{oj}), \forall i \in P$;
end
`/* Repete os passos 1-14 usando N para computar polNeg */`
$b = sum(polPos)/sum(polNeg)$;
for $v_i \in V$ **do**
  $lexicon[i] = polPos[i] - b * posPos[j]$;
  if $lexicon[i] < l$ **then**
    $lexicon[i] = 0$
  end
end
return $lexicon$;

6.2.4 Consolidação do Léxico

Como saída do algoritmo de propagação de grafo, estão os valores de sentimento associados a cada nó v. Esse conjunto pode conter *tokens* com '_NEG' ou '_LONG', já que *tokens* com essas terminações são possíveis saídas do *tokenizador*. Para chegar na polaridade final de uma palavra, deve-se considerar as polaridades da palavra original, dela com '_NEG' e dela '_LONG'. Os *tokens* com '_NEG' e '_LONG' são excluídos do léxico e os *tokens* originais são atualizados conforme a Equação 6.2.

$$final\_pol[v] = pol[v] + 1.5 * pol[v\_LONG] - pol[v\_NEG] - 1.5 * pol[v\_LONG\_NEG] \tag{6.2}$$

6.2.5 Algoritmo de Construção de Léxico

Nesta seção, é descrito como os algoritmos apresentados neste capítulo foram combinados para criar o algoritmo de construção de léxico. O pseudo-código do algoritmo é apresentado como Algoritmo 5. Esse algoritmo recebe como parâmetro os *tweets* sobre o domínio, dois conjuntos sementes de *hashtags* e *emoticons* positivos e negativos e o número de iterações do algoritmo (ou o número do tamanho máximo dos caminhos considerados). Primeiramente, todos os *tweets* recebidos são *tokenizados* usando o algoritmo apresentado na Seção 6.2.1. Em seguida, o grafo é construído usando os *tweets tokenizados*, de acordo com o algoritmo apresentado na Seção 6.2.2. Esse grafo, juntamente com os conjuntos semente e o número de iterações, são passados como parâmetro para o algoritmo de propagação de grafo (Seção 6.2.3), que retorna uma versão preliminar do léxico. Finalmente, essa versão preliminar é consolidada aplicando a Equação 6.2 e se transforma na versão final do léxico. Esse léxico é retornado pelo algoritmo. A Figura 6.2 mostra o passo-a-passo do algoritmo.

Algoritmo 5: Algoritmo de Construção de Léxico

Data: $tweets, positiveSeeds, negativeSeeds, iteractions$
Result: $lexicon$
`/* Retorna o léxico representado como um mapa entre palavras e valores */`
1 $tokenizedTweets$ = tokenize($tweets$);
2 $graph$ = buildGraph($tokenizedTweets$);
3 $tempLexicon$ = graphPropagation($graph, positiveSeeds, negativeSeeds, iteractions$);
4 $lexicon = new\ Map()$;
5 **for** $key, value \in tempLexicon$ **do**
6 $\quad lexicon[key] = tempLexicon[key] + 1.5 * tempLexicon[key\_LONG] - tempLexicon[key\_NEG]$;
7 **end**
8 return $lexicon$;

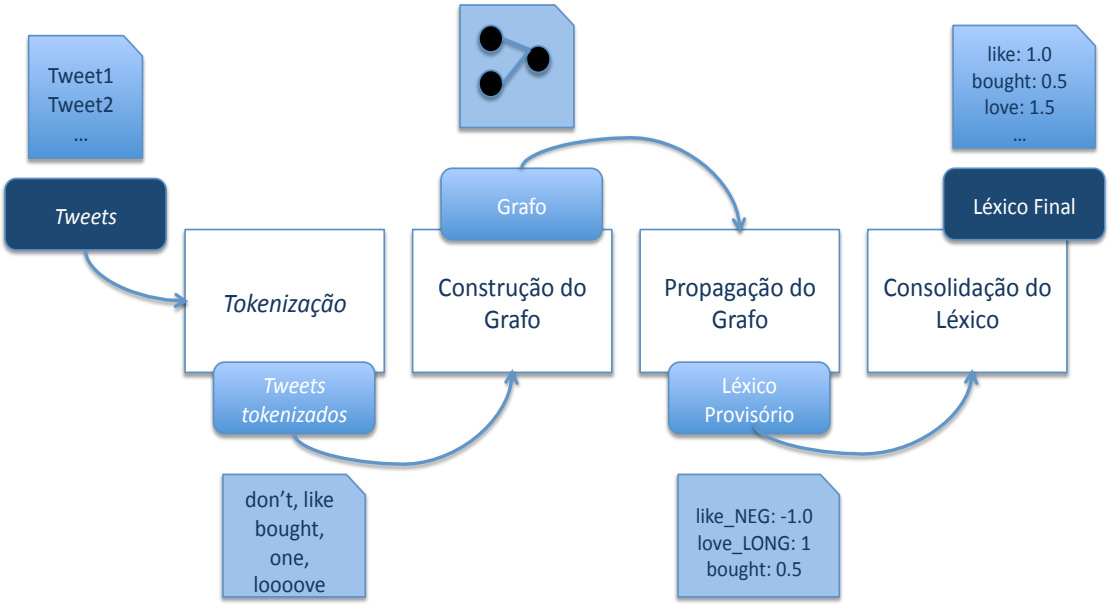

Figura 6.2: Algoritmo de Construção de Léxico

6.2.6 Implementação

O código dos algoritmos apresentados neste capítulo foram escritos em *Python*, usando a mesma arquitetura apresentada na Seção 4.1.3. Foi utilizada a biblioteka *nltk* [1] para a exclusão de *stop words*. Essa biblioteca contém uma lista com mais de 120 *stop words*, dentre as quais estão presentes '*our*', '*just*', '*about*' e '*where*'. O algoritmo de construção de grafos utilizou a biblioteca *numpy* [2].

O algoritmo de *tokenização* foi implementado usando a biblioteca de expressões regulares de *Python*: *re* [3]. Abaixo é exibida uma parte da expressão regular dos *emoticons*, que é uma das expressões regulares utilizadas na *tokenização*.

```
[<>]?                                   # hat
[:;=8x]                                 # eyes
[\-o\*\']?                              # optional nose
[xXoOsS\)\]\(\[dDpP/\:\}\{@\|\\]        # mouth
|
[*+xuoO@T^]                             # ^-^
[_.-]
[^*+xuoO@T^]
|
```

[1] http://www.nltk.org/
[2] http://www.numpy.org
[3] https://docs.python.org/2/library/re.html

```
[\\/\|]                                    # \o/
[oO]
[\\/\|]
```

6.3 Estudo de Caso

Nesta seção, é apresentado o estudo de caso do algoritmo de construção de léxico usando o domínio *iPhone 6*, o que resultou em um léxico de sentimento para *tweets* desse domínio.

6.3.1 Dados

Foram criados cinco léxicos para validar o algoritmo proposto e comparar os resultados. Os léxicos criados e testados variam na quantidade de *tweets* utilizados e no número de iterações utilizadas no algoritmo de propagação de grafo. Foram utilizados quatro conjuntos de dados diferentes para teste, além da variação do número de iterações. Todos os conjuntos de dados contêm *tweets* relacionados ao *iPhone 6*. Essas mensagem foram selecionadas usando o algoritmo da primeira etapa da ferramenta (Capítulo 4), excluindo os *spams* utilizando o algoritmo do Capítulo 5. A Tabela 6.5 apresenta o número de *tweets* utilizados em cada um dos conjuntos de dados e o número de iterações T. Pelos testes mostrados no próximo capítulo, pode-se perceber que aumentar o valor de T estava piorando os resultados e aumentando o tempo de execução do algoritmo de geração de léxicos. Por isso, não foi testado com T maior do que 2. Os *tweets* positivos correspondem a *tweets* que contêm um dos *emoticons* ou *hashtags* semente positivos, assim como os *tweets* negativos. Com isso, pode-se verificar que a classificação de *tweets* como positivos, negativos ou neutros nessa tabela apenas correspondem à verificação da presença de *emoticons* ou *hashtags* semente. Essa verificação pode ser feita automaticamente e não precisa de julgamento humano.

Tabela 6.5: Léxicos Criados

Léxico	**T**	***Tweets* Positivos**	***Tweets* Negativos**	***Tweets* Neutros**
1	1	2.631	648	3.000
2	2	2.631	648	3.000
3	1	2.631	648	5.000
4	1	2.631	648	15.000
5	1	2.631	648	25.000

Os conjuntos semente positivos e negativos foram compilados manualmente. O conjunto positivo contém 99 *emoticons* e 72 *hashtags*. O conjunto negativo contém 102 *emoticons* e 72 *hashtags*. A Tabela 6.6 apresenta alguns exemplos dos dois conjuntos.

Tabela 6.6: Exemplos de *emoticons* e *hashtags* dos conjuntos semente

Positivo	**Negativo**
:D	: (
=)	8/
<;P	=S
\o/	/o\
^.^	o.O
#amazing	#angry
#awesome	#disappointed
#excellent	#fail
#fantastic	#sad
#happy	#unacceptable

6.3.2 Resultados

Tabela 6.7: Número de Entradas dos Léxicos

Léxico	***Tokens* Positivos**	***Tokens* Negativos**
1	2.192	2.010
2	2.285	1.917
3	2.220	2.186
4	2.280	2.276
5	2.764	2.722

O algoritmo de construção de léxico foi executado com os conjuntos de dados descritos anteriormente e resultou em cinco léxicos. A quantidade de palavras positivas e negativas em cada léxico é apresentada na Tabela 6.7. A Tabela 6.8 mostra exemplos de palavras positivas e seus respectivos valores em cada léxico. A Tabela 6.9 exibe as mesmas informações para palavras negativas.

Tabela 6.8: Exemplos de palavras positivas e seus valores nos léxicos

Palavra	**Valor no Léxico**				
	1	**2**	**3**	**4**	**5**
lol	1.38	1.40	2.58	2.55	2.29
ordering	1.98	1.19	0.93	0.93	0.85
yay	0.99	0.85	3.39	3.47	3.57
#soexcited	0.35	0.29	1.36	1.28	0.7
:D	1.80	1.68	2.55	2.59	2.52

Tabela 6.9: Exemplos de palavras negativas e seus valores nos léxicos

Palavra	**Valor no Léxico**				
	1	**2**	**3**	**4**	**5**
kidney	-3.11	-2.85	-2.50	-2.48	-0.3
#camerafail	-0.48	-0.67	-0.75	-0.68	-0.86
#horrorstory	-3.38	-3.02	-3.11	-3.04	-0.51
#outdatedalready	-1.22	-1.07	-1.91	-1.92	-1.66
#pissedoff	-1.86	-2.41	-3.12	-3.03	-2.93

Analisando a Tabela 6.7, pode-se ver que o número de palavras no léxico aumenta à medida que o número de *tweets* utilizados aumenta. Além disso, mantendo o mesmo número de *tweets*, o léxico com T maior (léxico 2) possui mais entradas do que o léxico 1. Isso leva a acreditar que um léxico criado com mais *tweets* possa ser mais completo.

Analisando as Tabelas 6.8 e 6.9, pode-se constatar que os léxicos 1 e 2 possuem resultados parecidos, assim como os léxicos 3 e 4. O léxico 5 apresenta um resultado próximo aos léxicos 3 e 4 para a maioria das palavras. Porém, algumas das suas entradas possuem valores bem distintos, como '*#soexcited*', '*#horrorstory*' e '*kidney*'.

Um achado interessante ao analisar os léxicos é que '*kidney*' é uma palavra com conotação negativa relativa ao *iPhone 6*. Isso pode parecer surpreendente à primeira vista, mas passa a fazer sentido quando se analisa as mensagens do *iPhone 6* que contêm essa palavra. Na época de lançamento do produto, vários usuários estavam reclamando sobre o seu preço e dizendo que eles teriam que vender um rim (*kidney*) para conseguir comprar um *iPhone*. Dois exemplos desses *tweets* são '*Sells a kidney. Sells the car. Sells the home. Buys and iPhone 6. iPhone 7 is launched. :(#HorrorStory*' e '*#Apple has again made me realize that I am a poor man. Need to sell my kidney to buy one #iPhone6*

#sad story'. Esse é um exemplo de palavra cujo sentimento é dependente de domínio e que não seria considerada por um léxico genérico. Outra palavra dependente do domínio é '*ordering*', que tem um significado positivo e normalmente significa que o usuário está encomendando um *iPhone 6*.

É difícil avaliar a qualidade de um léxico de sentimento, porque isso normalmente exige bastante trabalho manual, especialmente para léxicos grandes. Comparar manualmente diferentes léxicos para escolher qual se comporta melhor é ainda mais trabalhoso e dá margem à interpretação, que pode ser diferente de uma pessoa para outra. Por causa disso, léxicos de sentimento são normalmente comparados através de uma ferramenta de análise de sentimento usando diferentes léxicos e comparando qual léxico leva ao melhor resultado. Essa é a abordagem escolhida para a comparação dos cinco léxicos criados e será apresentada no próximo capítulo, na Seção 7.2.

6.4 Trabalhos Futuros

Por motivos de performance, o número máximo de *tweets* que foi usado para testar esse algoritmo foi pouco mais de 28.000. Acredita-se que, com computadores melhores e um código otimizado, esse número possa ser aumentado. Os léxicos criados possuem um número relativamente pequeno de palavras se comparado com outros léxicos criados automaticamente. Executar o algoritmo com um conjunto de dados de 50.000 ou 100.000 *tweets* poderia criar um léxico mais completo. Porém, esse léxico mais completo deve ser avaliado pela ferramenta do próximo capítulo, para confirmar se realmente o desempenho é melhorado. Nos testes do próximo capítulo, é visto que o léxico criado com mais *tweets* não foi o que alcançou o melhor resultado, mas o esforço de testar o algoritmo com um conjunto maior de *tweets* e verificar o resultado é válido. Além disso, acredita-se que é oportuno testar esse algoritmo com valores diferentes de T, como $T = 3, 4, 5$. Esses valores não foram testados, pois os melhores resultados alcançados foram com $T = 1$, mas pode-se testar valores maiores de T, especialmente usando um número maior de *tweets*.

Outra possível melhoria do algoritmo é considerar *n-grams* na etapa de *tokenização*. Atualmente só são consideradas palavras isoladas (*unigrams*). Isso ajudaria a capturar expressões com mais de uma palavra, que muitas vezes possui um sentimento associado que é perdido quando as palavras são analisadas separadamente. Exemplos desses tipo de expressão são '*once in a life time*', '*melt in your mouth*' e '*run of the mill*'.

Para criar o grafo, foi usado similaridade de cossenos entre as palavras dos *tweets*. Um possível trabalho futuro seria testar o uso de *TF-IDF* e ver se isso teria um impacto positivo no desempenho do léxico. Isso pode ser interessante pois *TF-IDF* dá um peso menor a palavras muito frequentes e que não influenciam no sentimento.

Finalmente, seria interessante testar esse algoritmo com outros domínios diferentes e analisar o resultado. Acredita-se que esse algoritmo pode ser facilmente aplicável a outros domínios, visto que a única entrada específica para domínio necessária é um conjunto de *tweets* sobre esse domínio. Esse conjunto de *tweets* sobre o domínio pode ser facilmente coletado utilizando os algoritmos apresentados nos Capítulos 4 e 5. Para validar essa suposição, o algoritmo deve ser aplicado a outros domínios e os seus resultados devem ser avaliados. O Capítulo 8 apresenta um estudo de caso completo da ferramenta para um domínio diferente: cigarros eletrônicos.

Capítulo 7

Análise de Sentimento de *Tweets*

Este capítulo apresenta a última etapa da ferramenta proposta, que consiste em analisar o sentimento dos *tweets* selecionados. Isso é importante para avaliar a qualidade do léxico criado na etapa anterior e para completar o objetivo da ferramenta, que é fornecer a análise de sentimento de *tweets* de domínio específico. Neste trabalho, a análise de sentimento de *tweets* é definida como a classificação dessa mensagem como positiva ou negativa, de acordo com o seu conteúdo. Exemplos de *tweets* positivos e negativos são mostrados na Tabela 7.1.

A Seção 7.1 apresenta o algoritmo desenvolvido para prover a análise de sentimento de *tweets* de domínio específico. A Seção 7.2 relata um estudo de caso de aplicação do algoritmo e os resultados. Por fim, a Seção 7.3 discute os trabalhos futuros a respeito dessa etapa da ferramenta.

7.1 Algoritmo de Análise de Sentimento

O algoritmo de análise de sentimento recebe como entrada o léxico de sentimento e um *tweet* do domínio escolhido e retorna o sentimento dessa mensagem. Esse algoritmo não precisa de dados manualmente rotulados nem dados de treinamento. Por isso, é fácil aplicar esse algoritmo a outros domínios e léxicos e realizar a análise de sentimento para domínio específico fazendo uso de vários domínios diferentes.

A ideia do algoritmo é somar as polaridades de cada *token* do *tweet* no léxico, considerando a negação e *elongação* de *tokens*, que podem mudar sua polaridade. Se uma palavra é negada, a sua polaridade é invertida, multiplicando-a por -1. Se uma palavra é *elongada,* a sua polaridade é aumentada, multiplicando-a por 1.5. O sentimento é

Tabela 7.1: Exemplos de *Tweets* Positivos e Negativos

Tweet	**Sentimento**
The iPhone 6 looks amazing!!!	Positivo
#IPhone6, #iPhone6plus. #AppleWatch, and #U2. That's it! Apple rocked the house once again! #applelive	Positivo
Super excited for all of the new products coming out from #Apple with #AppleWatch, #ApplePay #iPhone6 and #iPhone6Plus	Positivo
All this new Apple stuff is blowing my mind. Cannot wait to get the new gadgets #Apple #AppleWatch #iPhone6 #iPhone6Plus	Positivo
Apple have made history yet again! #iPhone6 #iPhone6Plus #AppleWatch #ApplePay #AppleLive	Positivo
Completely disappointed with the #iPhone6 n #iPhone6Plus No wow factor this time Apple racing for benchmark rather than setting 1 #AppleLive	Negativo
200 Million already have an #Iwatch. It's called an #Iphone or #Itouch or a #ipad or a #macbook. Like a real Apple, it tastes the same.	Negativo
The iPhone 6 plus, where Galaxy was three years ago. #iPhone6Plus #AppleLive #iPhone6 #Samsung	Negativo
People who get the #iPhone6 or #iPhone6Plus are either retarded or have never used an android phone from last year	Negativo
But what if I don't want a phone as big as my face ? #iPhone6 #iPhone6Plus	Negativo

derivado da soma s de polaridades de acordo com a seguinte Equação 7.1

$$sentimento(t) = \begin{cases} positive & \text{if } s(t) > 0 \\ negative & \text{if } s(t) \leq 0 \end{cases} \tag{7.1}$$

7.1.1 Pseudo-código do Algoritmo

O pseudo-código do algoritmo é apresentado no Algoritmo 6. O primeiro passo do algoritmo consiste em extrair os *tokens* do *tweet* passado como entrada. Para isso, o mesmo *tokenizador* apresentado na Seção 6.2.1 é utilizado. Com o conjunto de *tokens* do *tweet*, cada *token* é analisado individualmente para extrair sua polaridade. Caso o *token* seja negado, a polaridade é multiplicada por -1. Caso o *token* seja *elongado*, a polaridade é multiplicada por 1.5. Caso contrário, é considerada a polaridade original do *token* no léxico. Se o *token* não existir no léxico, sua polaridade é considerada zero. A soma das

polaridades de todos os *tokens* é computada. Caso essa soma seja maior que zero, o *tweet* é considerado positivo. Caso contrário, o *tweet* é considerado negativo.

Algoritmo 6: Algoritmo de Análise de Sentimento

```
   Data: tweet, lexicon
   Result: sentiment
   /* Retorna o sentimento do tweet dado o léxico                    */
 1 tokens = tokenizer(tweet);
 2 polarity = 0;
 3 for token ∈ tokens do
 4 |   if token contains _NEG then
 5 |   |   polarity := polarity − lexicon[token];
 6 |   end
 7 |   else
 8 |   |   if token contains _LONG then
 9 |   |   |   polarity := polarity + 1.5 ∗ lexicon[token];
10 |   |   end
11 |   |   else
12 |   |   |   polarity := polarity + lexicon[token];
13 |   |   end
14 |   end
15 end
16 if polarity > 0 then
17 |   return Positive;
18 end
19 else
20 |   return Negative;
21 end
```

7.1.2 Métricas

Nesta seção, as métricas usadas para comparar o desempenho do algoritmo são apresentadas.

Precisão

Na Seção 3.4.3, foram vistas a definição e a fórmula para o cálculo da precisão. Adaptando para o algoritmo de análise de sentimento, precisão consiste na fração de *tweets* considerados positivos pelo algoritmo que são realmente positivos, como visto na Equação 7.2.

$$\text{Precisão} = \frac{\text{número de } \textit{tweets} \text{ verdadeiramente positivos selecionados pelo algoritmo}}{\text{número de } \textit{tweets} \text{ considerados positivos pelo algoritmo}} \tag{7.2}$$

Se não há falsos positivos, todos os *tweets* selecionados pelo algoritmo são realmente positivos, derivando uma precisão de 1. Se apenas metade dos *tweets* considerados positi-

vos pelo algoritmo são verdadeiramente positivos, a precisão é de 0.5. Com isso, pode-se constatar que maximizar a precisão significa minimizar os falsos positivos, o que significa que poucos *tweets* selecionados pelo algoritmo como positivos são na realidade negativos.

Recall

Na Seção 3.4.3, foram vistas a definição e a fórmula para o cálculo do *recall*. Adaptando para o algoritmo de análise de sentimento, *recall* é a fração de *tweets* positivos que são encontrados pelo algoritmo, como visto na Equação 7.3.

$$Recall = \frac{\text{número de } tweets \text{ verdadeiramente positivos selecionados pelo algoritmo}}{\text{número de } tweets \text{ positivos}} \tag{7.3}$$

Se não tiver nenhum falso negativo, todos os *tweets* positivos foram capturados pelo algoritmo, derivando um *recall* de 1. Se metade dos *tweets* positivos foram encontrados, o *recall* é 0.5. Com isso, pode ser visto que maximizar o *recall* significa minimizar os falsos negativos.

Otimização

Ao otimizar um algoritmo, é necessário saber qual métrica é mais importante. No caso atual, tanto precisão quanto *recall* são igualmente importantes, visto que não se quer deixar nenhum *tweet* positivo de fora, como também não se quer considerar como positivos *tweets* que são na realidade negativos. Com isso, o *F-measure* tradicional é utilizado para comparar os resultados no estudo de caso apresentado na Seção 7.2.

7.1.3 Implementação

O código desse algoritmo foi desenvolvido em *Python*, usando a mesma arquitetura apresentada na Seção 4.1.3.

7.2 Estudo de Caso

Nesta seção, é apresentado o estudo de caso do algoritmo de análise de sentimento de *tweets*. Na primeira seção é detalhado o conjunto de dados utilizado no estudo de caso, que consiste em *tweets* relacionados ao domínio *iPhone 6*. Em seguida, é apresentado o trabalho realizado para escolher o melhor léxico entre os criados no capítulo anterior. Por fim, são exibidos os resultados de comparação desse léxico escolhido com outros léxicos existentes na literatura e será feita uma avaliação desses resultados.

7.2.1 Dados

O conjunto de dados usados para testar o algoritmo consiste em 508 *tweets* relacionados ao domínio *iPhone 6*. Essas mensagens foram escolhidos do conjunto de *tweets* relacionados selecionados na primeira etapa da ferramenta, excluindo os *spams* da segunda etapa. Eles foram manualmente rotulados como positivos ou negativos e usados como '*ground truth*', resultando em um conjunto com 413 *tweets* positivos e 95 *tweets* negativos.

7.2.2 Otimização

Nesta seção, é exibida a otimização do algoritmo de análise de sentimentos para alcançar o melhor resultado. Dois parâmetros foram configurados para esse propósito: léxico e limiar para o léxico. O léxico corresponde aos léxicos criados na etapa anterior da ferramenta, variando o número de *tweets* do domínio utilizado pelo algoritmo e o número de iterações do algoritmo. Conforme mostrado no capítulo anterior, foram criados cinco léxicos sobre *iPhone 6* exibidos na Tabela 6.5. O parâmetro limiar para o léxico corresponde ao parâmetro existente no algoritmo de propagação de grafo, apresentado na Seção 6.2.3.

O algoritmo foi executado com esses cinco léxicos, considerando o limiar com a polaridade mínima do *token* como zero. O resultado está disponível na Tabela 7.2. Pelo resultado, pode-se ver que o léxico que apresenta o melhor desempenho é o léxico 4. Com isso, ele passa a ser chamado de *UnB Sentiment Lexicon* e é o léxico escolhido para comparação com outros léxicos na próxima seção.

Tabela 7.2: Resultados com Léxicos Criados

Léxico	**Precisão**	***Recall***	***F-Measure***
1	0.81799	0.94673	0.87766
2	0.81933	0.94431	0.87739
3	0.82452	0.94431	0.88036
4	**0.82316**	**0.94673**	**0.88063**
5	0.82189	0.92736	0.87144

O segundo teste diz respeito ao limiar de polaridade mínima do *token*. O léxico 4 foi utilizado com o limiar nos valores: 0, 0.1, 0.15, 0.2 e 0.3. Os resultados estão disponíveis na Tabela 7.3. Pelo resultado, pode-se ver que o melhor limiar é 0 e 0.1, que obtiveram um *recall* acima de 94% e *f-measure* acima de 88%. Como 0 foi um dos valores que obteve

o melhor resultado e é o mesmo que não usar nenhum parâmetro, esse limiar não é usado nos testes da próxima seção.

Tabela 7.3: Resultados do Léxico 4, variando o limiar

Limiar	**Precisão**	***Recall***	***F-Measure***
0	**0.82316**	**0.94673**	**0.88063**
0.1	**0.82316**	**0.94673**	**0.88063**
0.15	0.82143	0.94673	0.87964
0.2	0.82143	0.94673	0.87964
0.3	0.81953	0.93462	0.87330

7.2.3 Avaliação e Resultados

Para avaliar a qualidade do léxico criado, *UnB Sentiment Lexicon*, foram comparados o sentimento de *tweets* sobre o *iPhone 6* usando quatro léxicos: *Bing Liu Opinion Lexicon*, *NRC Hashtags Sentiment Lexicon*, *NRC Sentiment140 Sentiment Lexicon* e *UnB Sentiment Lexicon*. *Bing Liu Opinion Lexicon* é um léxico de sentimentos geral compilado manualmente por Bing Liu [25]. Esse léxico não é específico para *tweets* nem específico para nenhum domínio. É um léxico trabalhoso de ser feito, por ter sido construído de forma manual, mas pode ser utilizado para analisar qualquer texto. Os léxicos da NRC [38] são específicos para *tweets* mas não específicos para domínio e foram criados de maneira automática.

A Tabela 7.4 mostra a quantidade de entradas positivas e negativas em cada léxico. Analisando essa tabela, pode-se ver que os léxicos do NRC são bem maiores do que os léxicos de Bing Liu e UnB Sentiment. Isso acontece pois o NRC foi gerado automaticamente utilizando um corpus de 775.000 *tweets*, enquanto o léxico criado neste trabalho foi gerado com pouco mais de 18.000 *tweets*. O léxico de Bing Liu foi gerado manualmente, então também faz sentido ele ser menor do que o NRC.

O Algoritmo de Análise de Sentimento foi executado com cada um dos léxicos utilizando o conjunto de dados descrito. Os resultados são exibidos na Tabela 7.5.

Com esses resultados, pode-se ver que *UnB Sentiment Lexicon* obtém o melhor *recall*. Isso acontece porque esse léxico captura palavras específicas do domínio que os outros léxicos não capturam. O *Bing Liu Lexicon* possui a maior precisão. Isso acontece porque é um léxico geral, compilado manualmente e suas entradas são verificadas por humanos. Com isso, há uma confiança maior no seu conteúdo. Apesar disso, esse léxico não é prático para esse propósito, visto que ele captura menos da metade dos *tweets* positivos, como

Tabela 7.4: Número de Entradas dos Léxicos

Léxico	***Tokens* Positivos**	***Tokens* Negativos**
Bing Liu Lexicon	2.006	4.782
NRC Hashtag	32.048	22.080
NRC Sent. 140	38.308	24.159
UnB Sentiment	2.280	2.276

Tabela 7.5: Resultados da Análise de Sentimento

Lexicon	**Precisão**	***Recall***	***F-measure***
Bing Liu Lexicon	**0.87317**	0.43341	0.57928
NRC Hashtag	0.82844	0.88862	0.85748
NRC Sent. 140	0.82989	0.87409	0.85142
UnB Sentiment	0.82316	**0.94673**	**0.88063**

se pode verificar com o seu baixo *recall*. Isso acontece porque esse léxico não captura palavras específicas do domínio, especificidades do *Twitter* (*hashtags*, *emoticons*, que são bons indicativos de sentimento) e o contexto informal desse tipo de mensagem. Os léxicos do NRC possuem uma precisão muito similar ao *UnB Sentiment Lexicon*, mas com um *recall* menor. Eles capturam especificidades do *Twitter*, mas não do domínio. No geral, *UnB Sentiment Lexicon* supera os outros léxicos e obtém o melhor resultado.

7.3 Trabalhos Futuros

Alguns léxicos possuem uma precisão maior mas não contêm palavras específicas de domínio. Outros léxicos contêm uma vasta gama de palavras específicas para domínio, mas não são verificados manualmente, o que aumenta o risco de erros no léxico. Uma maneira de melhorar o desempenho do algoritmo de análise de sentimento é considerar o melhor de cada léxico e combiná-los para chegar ao resultado. Com isso, podem ser obtidos resultados melhores, como já foi demonstrado anteriormente [64].

Uma dificuldade encontrada ao analisar sentimento é o sarcasmo, que é difícil de ser detectado. No *Twitter*, especialmente, o sarcasmo é muito presente. Alguns estudos ([22], [15], [30]) focam em detectar sarcasmo, mas estava além do escopo desse trabalho lidar com isso. Um trabalho futuro é a incorporação do tratamento para sarcasmo, que pode melhorar o desempenho do algoritmo de análise de sentimento.

Outra maneira de aprimorar o algoritmo é considerar a estrutura retórica do texto, para conseguir diferenciar partes do texto separadas por conjunções adversativas, como '*but*' ou '*although*'. Muitas vezes, as duas partes do texto possuem sentimentos opostos. Por exemplo, no *tweet*: '*I've had each #iPhone since the original, but I may have to buck the trend, since the #iPhone6 looks #hideous #FireJonyIve #BadDesign*', a primeira parte possui uma conotação positiva e a segunda frase possui uma conotação negativa. Essas conjunções precisam ser analisadas para entender qual parte prevalece. Nesse caso, a segunda parte prevaleceu. Algoritmos que usam a teoria da estrutura retórica do texto (*RST - Rethorical Structure Theory*) podem ajudar a avaliar melhor esse tipo de frase.

Por fim, esse algoritmo não utiliza dados manualmente rotulados e, por isso, não pode utilizar um algoritmo de aprendizagem de máquina supervisionado. Uma possível alteração seria utilizar um desses algoritmos, o que provavelmente aumentaria o desempenho do algoritmo, mas com o custo de precisar de um conjunto de dados rotulados como positivos ou negativos.

Capítulo 8

Estudo de Caso da Ferramenta

Este capítulo tem como objetivo apresentar um estudo de caso completo da ferramenta proposta usando o domínio de cigarros eletrônicos. Cigarros eletrônicos, também conhecidos como *e-cigs* ou *electronic cigarettes*, são aparelhos mecânico-eletrônicos desenvolvidos com o objetivo de similar um cigarro e o ato de fumar. Esse dispositivo possui um sabor inalável com ou sem nicotina e apresenta diversos sabores. Os efeitos dos cigarros eletrônicos para a saúde ainda estão sendo analisados e não há consenso se esses dispositivos são melhores para a saúde do que os cigarros tradicionais. Esse domínio foi escolhido por ser um assunto bastante controverso e discutido, principalmente nos EUA e na Europa.

Há alguns estudos que dizem que cigarros eletrônicos não fazem mal à saúde, como [62]. Eles afirmam que cigarros eletrônicos possuem bem menos elementos tóxicos comparados aos cigarros convencionais e que o conteúdo cancerígeno dos cigarros eletrônicos é insignificante. Além disso, afirmam que os cigarros eletrônicos podem ajudar os fumantes a parar com o cigarro convencional.

Por outro lado, outras pesquisas apontam que os cigarros eletrônicos não são tão inofensivos assim e podem trazer efeitos colaterais tanto para fumantes ativos quanto passivos. Segundo [54], cigarros eletrônicos podem produzir gases que prejudicam a garganta, os olhos e o nariz de fumantes passivos. Além disso, de acordo com a Centro Americano para Controle de Doenças e Prevenção (*U.S. Centers for Disease Control and Prevention - CDC*), quase 250.000 jovens que nunca tinham experimentado cigarros convencionais, experimentaram cigarros eletrônicos em 2013 [66]. Isso mostra que os cigarros eletrônicos estão se popularizando entre os adolescentes e jovens, o que é um dado preocupante. A Organização Mundial de Saúde divulgou um estudo afirmando que os cigarros eletrônicos devem ser banidos de lugares fechados e que as vendas para menores de idade deveriam ser proibidas [43].

Com isso, pode-se esperar que os *tweets* sobre o tema sejam bastante controversos. Os usuários e fabricantes de cigarros eletrônicos defendem o seu livre uso, enquanto os

fumantes passivos e outras instituições de saúde são favoráveis à regulamentação mais estrita e ao banimento dos cigarros eletrônicos em ambientes fechados.

As Seções 8.1, 8.2, 8.3 e 8.4 apresentam os resultados para cada fase do algoritmo. Por fim, a Seção 8.5 discute o resultado final alcançado pela ferramenta nesse estudo de caso.

8.1 Coleta de *Tweets* Relacionados

No Capítulo 4, foi apresentado um estudo de caso do algoritmo de expansão de *hashtags*. Nele, o domínio de cigarros eletrônicos foi usado como teste e obteve precisão e *recall* acima de 97%. As *hashtags* obtidas como resultado desse algoritmo foram usadas para coletar um conjunto de dados de quase 40.000 *tweets* sobre cigarros eletrônicos. Esse conjunto de dados foi usado nas etapas seguintes, para treinar e testar os algoritmos.

8.2 Detecção e Exclusão de *Spam*

Nesta seção, são apresentados o treinamento e teste do algoritmo de detecção de *spam*. Foram usados 500 *tweets* sobre cigarros eletrônicos manualmente rotulados como *spam* ou não *spam* para chegar ao melhor conjunto de parâmetros para o algoritmo, conforme explicado no Capítulo 5. Em seguida, esse conjunto de parâmetros foi testado em um conjunto de dados com 110 *tweets* manualmente rotulados.

Os resultados obtidos no conjunto de treinamento e teste podem ser visualizados na Tabela 8.1. A quantidade de *tweets* utilizados para esse domínio foi bem inferior ao usado para o domínio *iPhone 6*, apresentado na Seção 5.2. Porém, pode-se observar que os resultados alcançados no domínio de cigarros eletrônicos também foram consistentes. Isso mostra que um número menor de mensagens rotuladas de treinamento pode ser suficiente para chegar a um bom resultado. Além disso, a aplicação do algoritmo com sucesso em um domínio bastante diferente do primeiro mostra que ele pode se adaptar para vários domínios diferentes.

Tabela 8.1: Resultados do treinamento e teste - Cigarros Eletrônicos

Conjunto de Dados	**Precisão**	***Recall***	***F-measure***
Treinamento	0.80695	0.81008	0.80851
Teste	0.88095	0.64912	0.74747

Em seguida, o melhor conjunto de parâmetros obtidos for usado para detectar *spam* na base de dados de 40.000 *tweets* sobre cigarros eletrônicos. Dessas mensagens, 22.579 foram

categorizadas como *spam*, representando 59.79% do conjunto de dados. No conjunto de *tweets* rotulados manualmente (*ground-truth*), a porcentagem de *tweets* que são *spam* foi 51.39%, um pouco abaixo do encontrado para a base de dados completa. Isso é esperado por causa dos valores encontrados para precisão, que são bons mas inferiores a 1. Isso significa que alguns *tweets* que não são *spam* são erroneamente classificados, aumentando a porcentagem de *spam* encontrado.

Tabela 8.2: Exemplos de *Tweets Spam* e Não-*Spam*

Tweet	**Classificação**
Aspire Atlantis on Sale for $35.99 w/ shipping!!! http://t.co/YUS8J36VgR #ecig #vapeporn #vapeon #subohm #clouds	*Spam*
360 Case Panasonic CGR18650CG Li-Ion Battery, 2250mAh NEW e-cig vape cells http://t.co/TZNbwDT0Dr #vape #vaporizer #vapejuice	*Spam*
2-PACK NEW BLACK 650 mAh Battery 510 Thread Vape Vaporizer http://t.co/2IDDRvVOwF #vape #vaporizer #vapejuice	*Spam*
Enter to win a full Leviathan E-Liquids sampler (150ml!): http://t.co/Uv5bEFpeEV via @cigbuyer.com #ecig #vape #eliquid #ejuice #giveaway	*Spam*
An electronic cigarette is made up of several pieces. http://t.co/kgSAEZzRRl #vape #weed #e-cig	*Spam*
Bought myself an e-cig as a Christmas present today. Best idea I've had all year.	Não-*Spam*
Whenever I see someone puffing on an e-cig I'm always a little disappointed they don't do a little synth jazz trumpet solo.	Não-*Spam*
I don't think asking mel for my ecig back was a good move tho	Não-*Spam*
My nose and throat feel like I have inhaled glass. Would e-cig 'second hand smoke' be better? Cause I will buy one for my mother :(	Não-*Spam*
This e-cig thing seems kind of silly, but I suppose it's "healthier/better"than regular cigs. Maybe my eyes will fall out after 10 years.;)	Não-*Spam*

A Tabela 8.2 apresenta alguns exemplos de classificação dos *tweets*. Pelos exemplos e pela classificação manual realizada, a conclusão é que uma parte significativa dos *tweets* que são *spam* nesse domínio são propagandas de cigarros eletrônicos ou do líquido usado com ele. Há diversas marcas responsáveis por esses produtos e elas ficam constantemente

publicando *tweets*, muitas vezes repetidos, para divulgar seus produtos. Esse é o caso dos três primeiros exemplos apresentados na planilha. A divulgação de promoções ou sorteios envolvendo cigarros eletrônicos também é encontrada, como no quarto exemplo da planilha. Há também diversos *tweets* que aparentam ser uma matéria sobre cigarros eletrônicos mas o *link* leva à página de uma marca de cigarros eletrônicos. Esse é o caso do quinto exemplo da planilha.

Muitos dos *tweets* que não são *spam* representam uma opinião ou experiência do usuário em relação à cigarros eletrônicos. Exemplos dessas mensagens podem ser vistas nos exemplos não-*spam* da planilha.

8.3 Construção do Léxico de Sentimento

Dentre os *tweets* da base de dados que não são *spam*, 5.000 foram utilizados para gerar o léxico de sentimentos. Foram usados os melhores parâmetros alcançados no Capítulo 6. Com isso, o algoritmo de construção de léxico foi executado usando $T = 1$.

O léxico resultante possui 3.248 entradas positivas e 2.209 entradas negativas. Na Tabela 8.3 podem ser vistos exemplos de palavras positivas e negativas e o valor de sentimento associado. Através desses exemplos, pode-se ver que há bastante termos específicos para o domínio de cigarros eletrônicos. A *hashtag* *#vapemas* é uma junção da palavra '*vape*' (que significa 'fumar' um cigarro eletrônico) com a palavra '*christmas*' (que é Natal em inglês). A palavra '*diy_ ejuice*' diz respeito ao líquido utilizado nos cigarros eletrônicos feito de forma manual pelos próprios usuários. Dentre as palavras negativas, pode-se citar '*banning*' e '*dangerous*', que dizem respeito às discussões sobre a segurança dos cigarros eletrônicos e se os mesmos deveriam ser banidos. *FDA* é o órgão governamental americano responsável pelo controle de alimentos, remédios, cigarros eletrônicos, dentre outros, e é constantemente citado nas discussões sobre a regulamentação de cigarros eletrônicos.

8.4 Análise de Sentimento

Por fim, o léxico criado na etapa anterior foi utilizado no algoritmo de análise de sentimentos para avaliar o sentimento dos 17.180 *tweets* que não são *spam*. O resultado obtido foram 14.672 *tweets* positivos (85.4%) e 2.508 *tweets* negativos (14.6%). A Tabela 8.4 mostra exemplos de *tweets* detectados pelo algoritmo como positivos e negativos.

Alguns exemplos falam de Natal ou Ano Novo por causa do período da coleta dos *tweets*, que incluiu o fim do ano de 2014. Dentre os *tweets* positivos, encontram-se mensagens postadas por usuários de cigarros eletrônicos falando sobre o seu uso. Os *tweets* que usam a *cashtag* $ECIG muitas vezes dizem respeito a ações da bolsa de valores do *Electronic*

Tabela 8.3: Exemplos de palavras positivas e negativas e seus valores no léxico

Positiva		**Negativa**	
Palavra	**Valor**	**Palavra**	**Valor**
=D	1.899	#disappointed	-2.260
#best	1.343	:/	-1.885
#vapemas	1.341	#nolife	-1.150
#awesome	1.069	pulmonology	-1.059
diy_ejuice	0.764	fda	-0.939
#merryvapingxmas	0.542	banning	-0.752
vapewild	0.523	dangerous	-0.698

Cigarettes International Group. Dentre os *tweets* negativos, há mensagens de pessoas totalmente contrárias ao uso de cigarros eletrônicos, inclusive empregando palavras de baixo calão. Há também mensagens relacionadas ao banimento de cigarros eletrônicos e aos malefícios que os mesmos podem trazer à saúde.

8.5 Resultado Final

A Tabela 8.5 apresenta o número total de *tweets* e sua divisão entre *spams*, positivos e negativos. Pelo resultado, pode-se constatar que uma grande parte dos *tweets* sobre cigarros eletrônicos são *spams*. Como foi visto anteriormente, a maior parte dos *spams* nesse domínio são relativos a propagandas.

Dentre os *tweets* relevantes, 85.4% são positivos (36.9% do total). Isso mostra que a maior parte dos *tweets* possuem um sentimento positivo sobre cigarros eletrônicos.

Para refinar a análise feita sobre cigarros eletrônicos, pode-se filtrar os *tweets* encontrados de acordo com a marca citada pelo usuário na mensagem. Com isso, pode ser extraído o sentimento dos usuários em relação àquele produto específico. Podem ser feitos também outros tipos de filtros de acordo com a necessidade do usuário da ferramenta.

Os testes para o domínio de cigarros eletrônicos foram feitos com sucesso e chegaram a um resultado satisfatório. Isso mostra que a ferramenta proposta pode ser utilizada para outros domínios diferentes do utilizado originalmente (*iPhone 6*). Testes com outros domínios podem ser feitos no futuro para validar esse posicionamento.

Tabela 8.4: Exemplos de *Tweets* Positivos e Negativos

Tweet	**Sentimento**
Remember remember... Oh wait... Wrong holiday :P merry #vapemas everyone :D Quiet night in #vaping on Christmas eve	Positivo
$ECIG it is going to make HUGE BOUM!!! :OP and news will appear soon too :) no worries for me here :P http://t.co/f42jX2y27C	Positivo
Happy new year guys :) and be ready Coz' 2015 gonna really rock hard in $ECIG world :)	Positivo
$ecig I love how everyone freaks out when it goes red for 20 min!!! CHILL does anyone know patience? $ecig is gold!	Positivo
Now relaxing, working the day in a vape shop and loving to about hear peeps stopping smoking! Makes my day... #Vape	Positivo
Mm figured it out, haven't had a cigarette since saturday. Trying to quit is a pain in my ass. This ecig isn't cutting it right now #shit	Negativo
Three e-cigarette TV adverts banned Nothing makes a person unattractive as quickly as a cigarette, e- or other. http://t.co/2BaH2snfvy	Negativo
My hands are so frozen that I didn't even feel my ecig slip out of my hand :-(#byeecig	Negativo
@MontgomeryMayor "Electronic Smoking Devices"= deliberate misrepresentation of the product.Codifying the #lie doesn't make it right. #ecigs	Negativo
Hate all these old fuckers that think an e-cig is a gate way to smoking indoors IT ISNT! FUCK OFF OUTSIDE in The COLD where you belong	Negativo

Tabela 8.5: Resultado Final - Cigarros Eletrônicos

Tweets	***Spams***	**Positivos**	**Negativos**	**Total**
Número	22.579	14.672	2.508	**39.759**
Porcentagem	56.8%	36.9%	6.3%	**100%**

Capítulo 9

Conclusão

Esta pesquisa teve como objetivo a proposição de uma abordagem e o desenvolvimento de uma ferramenta unificada para executar a análise de sentimento de *tweets* de forma eficiente e para um domínio particular. Foram apresentadas as quatro etapas da ferramenta proposta: coleta de *tweets* relacionados, detecção de *spam*, construção do léxico de sentimento e análise de sentimento de *tweets*. Por fim, foi discutido um estudo de caso completo da ferramenta para o domínio de cigarros eletrônicos.

O objetivo principal deste trabalho é lidar com todas as etapas necessárias e suficientes para fazer a análise de sentimento de *tweets* de domínio específico. A primeira etapa é a coleta de *tweets* relacionados. Essa etapa é fundamental, pois proporciona a coleta de um grande conjunto de *tweets* relacionados ao domínio sem a necessidade de o usuário ter que pesquisar no *Twitter* para encontrar todas as *hashtags* sobre um determinado assunto. Um algoritmo para expandir *hashtags* limitadas em um conjunto maior e mais completo de *hashtags* foi proposto para coletar os *tweets* relacionados e demonstrou ter obtido bons resultados, com precisão média acima de 90%.

A segunda etapa é a detecção de *spam*. Essas mensagens devem ser excluídas porque iriam impactar no léxico de forma indesejada, já que não representam opiniões reais de usuários. Além disso, dependendo do domínio, a porcentagem de *spam* é muito alta, o que impactaria ainda mais no léxico. Isso torna essa etapa da ferramenta muito importante para o seu resultado final. O algoritmo de detecção de *spam* usou várias características dos *tweets* e do usuário e foi capaz de identificar *tweets* que são *spam* com *recall* acima de 98%.

A terceira etapa da ferramenta é a criação do léxico de sentimento de *tweets* de domínio específico. Ela pode ser considerada a mais importante, pois o léxico têm um impacto significativo no desempenho da ferramenta. Um algoritmo foi proposto para criar o léxico sem fazer uso de dados rotulados do domínio escolhido, usando uma abordagem baseada em corpus. Esse algoritmo é baseado na propagação de grafos e é dividido em quatro

etapas: *tokenização*, construção do grafo, propagação do grafo e consolidação do léxico. O léxico final é chamado de *UnB Sentiment Lexicon* e incorpora expressões cujo sentimento variam de um domínio para outro.

É importante ressaltar que, apesar de existirem vários léxicos na academia, não foi encontrado algum com todos os benefícios da metodologia proposta. Foram apresentados alguns léxicos específicos para *tweets*, o que traz o benefício de se adequarem melhor à realidade do *Twitter*. Porém, o fato deles serem genéricos em relação ao domínio diminui a sua precisão. Foram também apresentados léxicos relacionados a domínios específicos, porém eles não se adaptam bem à realidade do *Twitter*. Foi apresentado um léxico específico para *tweets* e relacionado ao domínio Justin Bieber [18], porém a sua construção foi altamente manual.

Por fim, a última etapa é a análise de sentimento de *tweets*. Ela é totalmente necessária, pois completa o objetivo da ferramenta, que é prover o sentimento desses *tweets*, além de servir como teste para validar o desempenho dos léxicos de sentimento criados. Esse algoritmo é responsável por receber um *tweet* e identificar se ele possui um sentimento positivo ou negativo sobre o domínio. Esse algoritmo foi testado com o nosso léxico e outros três léxicos existentes na literatura. O *UnB Sentiment Lexicon* alcançou o melhor resultado entre os léxicos, obtendo uma precisão de 82% e um *recall* acima de 94%.

Importa ressaltar que o presente trabalho consolida em uma única ferramenta todo o trabalho necessário para a análise de sentimento de *tweets*, o que não tinha sido feito anteriormente. A ferramenta proposta foi testada integralmente para o domínio '*iPhone 6*' e 'cigarros eletrônicos' e obteve resultados convincentes. Ainda, essa ferramenta pode ser adaptada para funcionar com outros domínios, tendo como único requisito de entrada um conjunto de *tweets* rotulados como *spam* ou não.

Foi também resultado da pesquisa a publicação de um artigo na International Conference on Information Fusion 2015 [52], que irá acontecer em julho de 2015, em Washigton, D.C. e de um artigo no International Congress on Industrial and Applied Mathematics 2015 [51], que irá acontecer em agosto de 2015, em Beijing, China.

9.1 Considerações Finais e Trabalhos Futuros

Foram propostas possíveis melhorias e trabalhos futuros em cada uma das quatro etapas da presente ferramenta. Para o algoritmo de expansão de *hashtags*, é interessante fazer uso de informações semânticas sobre o *tweet*. Além disso, um algoritmo de agrupamento aplicado à *hashtags* pode trazer bons resultados. O trabalho futuro para esse algoritmo consiste em tentar diminuir a quantidade de entrada necessária para o algoritmo, mantendo os bons resultados encontrados.

O algoritmo de detecção de *spam* poderia se beneficiar do uso de algoritmos de aprendizagem de máquina. No algoritmo proposto, cada característica é considerada individualmente. Um algoritmo de aprendizagem de máquina leva em consideração todas as características combinadas para chegar ao resultado final, então possivelmente iria trazer melhores resultados. Um trabalho futuro para essa fase é testar diferentes algoritmos de aprendizagem de máquina e compará-los entre si e com o algoritmo atual.

Já o algoritmo de construção de léxico poderia ser testado com conjuntos de dados maiores e outros valores de T. Isso pode ser feito com uma adaptação no código do algoritmo e computadores mais potentes. Além disso, o algoritmo atual considera apenas *unigrams*. O uso de *n-grams* de diferentes tamanhos pode melhorar o desempenho do algoritmo, visto que há muitas expressões constituídas de mais de uma palavra e que possuem um sentimento associado.

Para o algoritmo de análise de sentimento, o uso de aprendizagem de máquina poderia melhorar o seu desempenho. Por outro lado, isso acrescentaria a necessidade de ter dados rotulados de cada domínio, o que não é necessário com o algoritmo atual. Outro possível trabalho futuro é levar em consideração a estrutura retórica do texto ao analisar o sentimento do *tweet*. Isso é importante para decidir o sentimento final de um texto com sentimentos contraditórios, para saber qual sentimento prevalece no *tweet*.

Além disso, seria interessante testar todas as etapas da ferramenta com outros domínios. No presente trabalho, esse teste foi feito para dois domínios: *iPhone 6* e cigarros eletrônicos. Contudo, seria interessante a realização desse teste para outros domínios diferentes dos testados inicialmente, para validar a hipóteste de que essa abordagem e ferramenta pode ser aplicada a vários domínios diferentes.

Outro trabalho futuro seria testar a ferramenta para *tweets* em português. Esse trabalho se baseou em mensagens em inglês. Porém, do jeito que a ferramenta foi criada, funcionaria também para outras linguas. Para isso, basta criar os conjuntos de *hashtags* semente positivas e negativas usados no algoritmo de construção de léxico na linguagem escolhida.

Finalmente, o presente trabalho propôs uma abordagem e uma ferramenta para análise de sentimento de *tweets* de domínio específico, ficando responsável por todas as etapas necessárias e suficientes para atingir o seu resultado final. Cada etapa da ferramenta foi testada individualmente e obteve resultados satisfatórios. A ferramenta como um todo foi testada para o domínio '*iPhone 6*', alcançando um *recall* final acima de 94%. Além disso, a ferramenta foi testada integralmente com um domínio bastante diferente do primeiro, que foi o de cigarros eletrônicos. Enfim, o trabalho realizado produziu várias contribuições científicas, dentre as quais é possível citar: o algoritmo de expansão de *hashtags*, o algoritmo de detecção de *spam*, o algoritmo automático de construção de

léxicos de sentimento para *tweets* de domínio específico e uma abordagem que consolida todas as etapas necessárias para a análise de sentimentos.

Referências

[1] Hazim Almuhimedi, Shomir Wilson, Bin Liu, Norman Sadeh, and Alessandro Acquisti. Tweets are forever: a large-scale quantitative analysis of deleted tweets. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 897–908. ACM, 2013. 12

[2] Akshat Bakliwal, Piyush Arora, Senthil Madhappan, Nikhil Kapre, Mukesh Singh, and Vasudeva Varma. Mining sentiments from tweets. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, pages 11–18. Association for Computational Linguistics, Jeju, Korea, July 2012. 23, 26, 32

[3] Fabrício Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgílio Almeida. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, volume 6, pages 75–83, 2010. 10, 13, 14, 52

[4] James Benhardus and Jugal Kalita. Streaming trend detection in twitter. *International Journal of Web Based Communities*, 9(1):122–139, 2013. 12, 13

[5] John Rupert Lyon-Bowes Bernard. *The Macquarie Thesaurus*. Macquarie Library, 1986. 16

[6] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009. 30

[7] J. Bollen and Huina Mao. Twitter mood as a stock market predictor. *Computer*, 44(10):91–94, Oct 2011. 2

[8] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011. 2

[9] Danah M. Boyd and Nicole B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007. 1

[10] Michael Buckland and Fredric Gey. The relationship between recall and precision. *Journal of the American Society for Information Science*, 45(1):12–19, 1994. 38

[11] Simon Carter, Manos Tsagkias, and Wouter Weerkamp. Twitter hashtags: Joint translation and clustering. *Proceedings of the ACM Web Science Conference*, pages 1–3, 2011. 8, 10

[12] Yejin Choi and Claire Cardie. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 2, pages 590–598. Association for Computational Linguistics, 2009. 19, 25

[13] Zi Chu, Indra Widjaja, and Haining Wang. Detecting social spam campaigns on twitter. In *Applied Cryptography and Network Security*, pages 455–472. Springer, 2012. 11, 13, 14, 52

[14] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 241–249. Association for Computational Linguistics, 2010. 23, 26

[15] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116. Association for Computational Linguistics, 2010. 32, 80

[16] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A high-coverage lexical resource for opinion mining. Technical report, Institute of Information Science and Technologies (ISTI) of the Italian National Research Council (CNR), 2007. 16, 25

[17] Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok N Choudhary. Towards online spam filtering in social networks. In *19th Annual Network and Distributed System Security Symposium, NDSS*, 2012. 12, 13

[18] M. Ghiassi, J. Skinner, and D. Zimbra. Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network. *Expert Systems with Applications: An International Journal*, 40(16):6266–6282, 2013. 21, 25, 28, 89

[19] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520, 2011. 19, 25

[20] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford University, California*, pages 1–12, 2009. 2, 21

[21] Pollyanna Gonçalves, Matheus Araújo, Fabrício Benevenuto, and Meeyoung Cha. Comparing and combining sentiment analysis methods. In *Proceedings of the first ACM conference on Online social networks*, pages 27–38. ACM, 2013. 22, 26

[22] Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 2, pages 581–586. Association for Computational Linguistics, 2011. 80

[23] Michael Gordon and Manfred Kochen. Recall-precision trade-off: A derivation. *Jounal of the American Society for Information Science*, 40(3):145–151, 1989. 38

[24] Bas Heerschop, Frank Goossen, Alexander Hogenboom, Flavius Frasincar, Uzay Kaymak, and Franciska de Jong. Polarity analysis of texts using discourse structure. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1061–1070. ACM, 2011. 22, 26

[25] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177. ACM, 2004. 15, 25, 79

[26] Jan Kalmeijer. Hashtag clustering to summarize the topics discussed by dutch members of parliament. *Report for Dutch Parliament*, 2014. 8, 10

[27] Myungsook Klassen. Twitter data preprocessing for spam detection. In *Future Computing 2013, The Fifth International Conference on Future Computational Technologies and Applications*, pages 56–61, 2013. 11, 13, 14, 52

[28] Su Mon Kywe, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. On recommending hashtags in twitter networks. In *Social Informatics*, pages 337–350. Springer, 2012. 9, 10

[29] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710, 1966. 35

[30] Christine Liebrecht, Florian Kunneman, and Antal Van den Bosch. The perfect solution for detecting sarcasm in tweets #not. *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 29–37, June 2013. 80

[31] Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012. 27, 28, 29, 30, 31, 59

[32] Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *Proceedings of the 20th international conference on World wide web*, pages 347–356. ACM, 2011. 17, 25

[33] Juan Martinez-Romo and Lourdes Araujo. Detecting malicious tweets in trending topics using a statistical analysis of language. *Expert Systems with Applications*, 40(8):2992–3000, 2013. 11, 13, 14

[34] Allie Mazzia and James Juett. Suggesting hashtags on twitter. Technical report, University of Michigan, 2009. 9, 10

[35] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093 – 1113, 2014. 23

[36] Zachary Miller, Brian Dickinson, William Deitrick, Wei Hu, and Alex Hai Wang. Twitter spammer detection using data stream clustering. *Information Sciences*, 260:64–73, 2014. 11, 14

[37] Saif Mohammad. #emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 246–255. Association for Computational Linguistics, 2012. 20

[38] Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *Proceedings of the Seventh International Workshop on Semantic Evaluation Exercises (SemEval-2013)*, 2:321–327, 2013. 3, 20, 23, 25, 26, 32, 79

[39] Saif Mohammad and Peter Turney. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34. Association for Computational Linguistics, 2010. 16, 25

[40] Saif Mohammad and Peter Turney. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465, 2013. 16

[41] Andrés Montoyo, Patricio Martínez-Barco, and Alexandra Balahur. Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments. *Decision Support Systems*, 53(4):675–679, 2012. 23

[42] Subhabrata Mukherjee and Pushpak Bhattacharyya. Sentiment analysis: A literature survey. *CoRR*, abs/1304.4520, 2013. 23

[43] Smitha Mundasad. 'ban e-cigarette use indoors,' says who @ONLINE. url: http://www.bbc.com/news/health-28937610, August 2014. 82

[44] Cristina Ioana Muntean, Gabriela Andreea Morar, and Darie Moldovan. Exploring the meaning behind twitter hashtags through clustering. In *Business Information Systems Workshops*, pages 231–242. Springer, 2012. 8, 10, 49

[45] Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. Summarizing sporting events using twitter. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 189–198. ACM, 2012. 12, 13

[46] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. From tweets to polls: Linking text sentiment to public opinion time series. *International AAAI Conference on Weblogs and Social Media - ICWSM 2010*, 11:122–129, 2010. 2

[47] Ozer Ozdikis, Pinar Senkul, and Halit Oguztuzun. Semantic expansion of hashtags for enhanced event detection in twitter. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '12, pages 20–24. IEEE Computer Society, 2012. 7, 10

[48] Soujanya Poria, Nir Ofek, Alexander Gelbukh, Amir Hussain, and Lior Rokach. Dependency tree-based rules for concept-level aspect-based sentiment analysis. In *Semantic Web Evaluation Challenge*, volume 475 of *Communications in Computer and Information Science*, pages 41–47. Springer International Publishing, 2014. 22, 26

[49] Aniket Rangrej, Sayali Kulkarni, and Ashish V Tendulkar. Comparative study of clustering techniques for short text documents. In *Proceedings of the 20th International Conference Companion on World Wide Web*, pages 111–112. ACM, 2011. 9, 10

[50] Delip Rao and Deepak Ravichandran. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics, 2009. 14, 25, 34

[51] Patricia L V Ribeiro, Li Weigang, and Tiancheng Li. Domain-specic tweet sentiment analysis. *Proceedings of the International Congress on Industrial and Applied Mathematics (ICIAM 2015)*, August 10-14. (in press). 89

[52] Patricia L V Ribeiro, Li Weigang, and Tiancheng Li. A unified approach for domain-specific tweet sentiment analysis. *Proceedings of the International Conference on Information Fusion (FUSION 2015)*, July 2015. (in press). 89

[53] Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 105–112. Association for Computational Linguistics, 2003. 15

[54] Jennifer Rooney. E-cigs carry harmful side effects for smokers, nonsmokers @ONLINE. url: http://www.dailynebraskan.com/opinion/rooney-e-cigs-carry-harmful-side-effects-for-smokers-nonsmokers, February 2014. 82

[55] Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. Topical clustering of tweets. *Proceedings of the ACM SIGIR 3rd Workshop on Social Web Search and Mining (SIGIR - SWSM 2011)*, 2011. 8, 10

[56] Hassan Saif, Yulan He, and Harith Alani. Semantic sentiment analysis of twitter. In *The Semantic Web – 11th International Semantic Web Conference*, volume 7649 of *Lecture Notes in Computer Science*, pages 508–524. Springer, 2012. 23, 26, 32

[57] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. Computer Science Series. McGraw-Hill, 1983. 36

[58] Philip Stone, Dexter Dunphy, Marshall Smith, and Daniel Ogilvie. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press, 1966. 15, 16

[59] Carlo Strapparava and Alessandro Valitutti. Wordnet affect: an affective extension of wordnet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, volume 4, pages 1083–1086. European Language Resources Association, 2004. 16

[60] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 1–9. ACM, 2010. 11, 13, 14, 52

[61] Songbo Tan and Qiong Wu. A random walk algorithm for automatic construction of domain-oriented sentiment lexicon. *Expert Systems with Applications*, 38(10):12094–12100, 2011. 18, 25

[62] Wellness Team. New research: E-cigs safer alternative to regular cigarettes @ONLINE. url: http://health.clevelandclinic.org/2014/08/new-research-e-cigs-safer-alternative-to-regular-cigarettes, August 2014. 82

[63] Oren Tsur, Adi Littman, and Ari Rappoport. Efficient clustering of short messages into general domains. In *International AAAI Conference on Weblogs and Social Media - ICWSM 2013*, pages 621–630. AAAI Press, 2013. 8, 10

[64] Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 777–785. Association for Computational Linguistics, 2010. 13, 25, 34, 60, 80

[65] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354. Association for Computational Linguistics, 2005. 15, 25

[66] Amanda Woerner. E-cigarettes: The side effects nobody talks about @ONLINE. url: http://www.thedailybeast.com/articles/2014/09/25/e-cigarettes-the-side-effects-nobody-talks-about.html, September 2014. 82

[67] Sarita Yardi, Daniel Romero, Grant Schoenebeck, and Danah Boyd. Detecting spam in a twitter network. *First Monday*, 15(1), 2009. 11, 13, 14

[68] Mu Zhu. Recall, precision and average precision. Technical report, Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, 2004. 37, 38

[69] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002. 33