

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Relevância da Tradução de Textos de Português para
Inglês no processo de Classificação Binária de
Sentimento de postagens rápidas em
Redes Sociais Online

Evelyn de Souza Farias

Campina Grande, Paraíba, Brasil

Agosto - 2016

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Relevância da Tradução de Textos de Português para
Inglês no processo de Classificação Binária de
Sentimento de postagens rápidas em
Redes Sociais Online

Evelyn de Souza Farias

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande – Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Sistemas de Computação

Reinaldo Gomes (Orientador)

Anderson Costa (Orientador)

Campina Grande, Paraíba, Brasil

© Evelyn de Souza Farias, 10/08/2016

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

F224r

Farias, Evelyn de Souza.

Relevância da tradução de textos de português para inglês no processo de classificação binária de sentimento de postagens rápidas em redes sociais online / Evelyn de Souza Farias. – Campina Grande, 2017.

108 f. : il. color.

Dissertação (Mestrado em Ciências da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2016.

"Orientação: Prof. Dr. Reinaldo César de Moraes Gomes, Prof. Dr. Anderson Fabiano Batista Ferreira da Costa".

Referências.

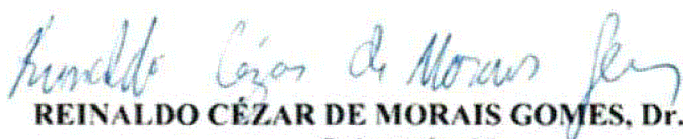
1. Análise de Dados. 2. Mineração de Opinião. 3. Tradução Automática de Textos. 4. Classificação de Sentimento. I. Gomes, Reinaldo César de Moraes. II. Costa, Anderson Fabiano Batista Ferreira da. III. Título.

CDU 004.62(043)

**"RELEVÂNCIA DA TRADUÇÃO DE TEXTOS DE PORTUGUÊS PARA INGLÊS NO
PROCESSO DE CLASSIFICAÇÃO BINÁRIA DE SENTIMENTO DE POSTAGENS
RÁPIDAS EM REDES SOCIAIS ONLINE"**

EVELYN DE SOUZA FARIAS

DISSERTAÇÃO APROVADA EM 29/08/2016




**REINALDO CÉZAR DE MORAIS GOMES, Dr., UFCG
Orientador(a)**



**ANDERSON FABIANO BATISTA FERREIRA DA COSTA, Dr., IFPB
Orientador(a)**

**MARCO AURELIO SPOHN, Ph.D, UFFS
Examinador(a)**



**ALISSON VASCONCELOS DE BRITO, D.Sc, UFPB
Examinador(a)**

CAMPINA GRANDE - PB

Resumo

A análise e mineração de opinião em dados de texto extraídos de redes sociais online tem ganhado bastante força nos últimos anos, tornando-se uma área muito pesquisada e difundida em todo o mundo. Entretanto, esses processos são dependentes do idioma dos dados escritos, sendo o inglês o mais estudado e, conseqüentemente, o idioma que abrange uma maior quantidade de técnicas e soluções. Nesse contexto, a obtenção de resultados globais nessas áreas de pesquisa torna-se bastante custosa em tempo, sendo assim, com o avanço dos tradutores automáticos de texto e a tradução de dados para o inglês ser utilizada por alguns autores, julgamos necessário analisar o impacto dessas traduções no processo de classificação de sentimento. Os experimentos realizados em nosso trabalho mostraram resultados satisfatórios em métricas de avaliação e comparados aos resultados obtidos em trabalhos de outros autores em experimentos semelhantes de tradução de dados de texto e classificação de polaridade de sentimento. Os sistemas de tradução automática utilizados em nosso trabalho apresentaram uma tendência de traduções equiparadamente eficientes, mostrando que esses sistemas evoluíram bastante nos últimos anos. Quanto à classificação de dados de texto traduzidos automaticamente podemos dizer que, a partir dos resultados obtidos, a tradução automática de texto pode apresentar bons resultados para alguns casos. Porém, há a necessidade de experimentação com volumes de dados de treino mais abrangentes nas duas línguas estudadas neste documento.

Abstract

The sentiment analysis and opinion mining in text data extracted from online social media services has gained enough strength in recent years, making it an area very researched and disseminated worldwide. However, these processes are language dependent and the english language is the most studied one, covering a larger amount of techniques and solutions in the field. In this context, obtaining overall results in these research areas becomes quite time consuming, so with the advancement of automatic text translators and that data automatic translated to english is used by some authors, it is necessary to analyze the impact these automatic translations cost in a text classification process. The experiments performed in our study showed satisfactory results in evaluation metrics and compared to the results obtained in works by other authors in similar experiments using automatic translations of text data and sentiment polarity classification. The machine translation systems used in our study showed a trend of equally efficient translations, showing that these systems have evolved considerably in recent years. As for the automatically translated text data classification we can say that from the results obtained, automatic text translation can present good results in some scenarios and case studies. However, there is a need for experimentation with more comprehensive training data volumes in the two languages studied in this document.

Agradecimentos

Agradeço primeiramente a Deus por me proporcionar a vida, nesta os desafios e dificuldades, mas também a força e foco para superar os obstáculos.

À minha família pelo apoio, especialmente à minha mãe guerreira que trabalhou duro para dar educação a meus irmãos e a mim. Obrigada, mãe!

Aos amigos que fiz durante o curso de mestrado, que me ajudaram durante todo esse processo. Aos amigos de longa data que me escutaram por vezes a ditar partes da dissertação pra ver como soava. Muito obrigada a todos vocês! Mas em especial à Thaciana e Carol por me aguentarem quando ligava nas madrugadas. Obrigada, pessoal!

Aos professores e funcionárias da COPIN que ajudaram direta e indiretamente na realização deste trabalho. Em especial ao professor Carlos que me ajudou muito a abrir o leque de opções e horizontes em *machine learning*, e ao professor Nazareno por me alertar que o que eu pretendia desenvolver precisaria de dez de mim.

Aos meus orientadores Reinaldo e Anderson, pela contribuição substancial para a concretização deste trabalho e orientação recebida. Obrigada pela paciência comigo em todo esse processo e a compreensão quando eu passei por momentos difíceis. Muito obrigada!

Agradeço a meu pai que me deixou esse ano e foi para o plano superior. Ele que morria de orgulho de mim quando dizia “Babá vai ser mestra!” com aquela voz de brincadeira, que saía da cidade onde morava para ir me buscar na UFCG quando tinha aulas à noite, que escutava minhas lamúrias e reclamações no telefone, que me dava conselhos,... Meu pai, meu amigo, meu anjo. Meu agradecimento especial vai para ele que tanto me ensinou e a quem eu espero continuar dando orgulho sempre.

Ao CNPq, pelo apoio financeiro.

Conteúdo

1 Introdução	1
1.1 Problema de Pesquisa	2
1.2 Objetivos	4
1.3 Relevância	5
1.4 Estrutura da Dissertação	6
2 Fundamentação Teórica	7
2.1 A rede social Twitter	7
2.1.1 Twitter como fonte de dados para classificação de sentimento	7
2.1.2 Twitter API	8
2.2 Classificação de sentimento	11
2.2.1 Níveis de Classificação de Sentimento	12
2.2.2 Seleção/Extração de <i>features</i> de texto	13
2.2.3 Representação do Texto	17
2.2.4 Técnicas de classificação de texto	18
3 Trabalhos Relacionados	24
3.1 Classificação de Sentimento utilizando a plataforma Twitter como fonte de dados de texto.....	24
3.2 Classificadores Multilinguísticos utilizando dados de texto traduzidos por máquina	26
4 Coleta e Preparação dos Dados	29
4.1 Coleta de Dados	29
4.1.1 Definição do contexto da pesquisa	30
4.2 Preparação dos Dados	33
4.2.1 Definição dos conjuntos de dados.....	34
4.2.2 Etapas de tratamento dos dados	34
5 Classificação de Sentimento	44

5.1	Marcação e classificação de subjetividade	45
5.2	Classificação prévia de polaridade de sentimento do conjunto de dados em inglês e marcação das sentenças.....	47
5.3	Métricas de avaliação da classificação de polaridade de sentimento	49
5.4	Classificação de sentimento em nossos conjuntos de dados.....	51
5.4.1	Classificação de sentimento de dados de teste submetidos à tradução automática	54
5.4.2	Classificação de sentimento de conjuntos não previamente marcados e submetidos à tradução automática	57
6	Avaliação da Qualidade de Classificação de Sentimentos em dados com e sem tradução.....	60
6.1	Classificação de sentimento nos conjuntos de dados sem tradução	60
6.1.1	Análise do Primeiro Experimento.....	63
6.2	Classificação de sentimento de dados de teste submetidos à tradução automática ...	64
6.2.1	Análise do Segundo Experimento.....	67
6.3	Classificação de sentimento de conjuntos não previamente marcados e submetidos à tradução automática	68
6.3.1	Análise do Terceiro Experimento	71
7	Considerações finais.....	72
7.1	Considerações finais	72
7.2	Trabalhos Futuros	73
	Bibliografia	75
A	Código Fonte do <i>crawler</i> de dados do Twitter.....	78
B	Listas de <i>Stop Words</i> e N-gramas	94
C	Listas de palavras polares	100
D	Saídas das classificações realizadas no Weka 3.8	101

Lista de Símbolos

API - *Application Programming Interface*

SVM - *Supervised Vector Machine*

SMO - *Sequential Minimal Optimization*

PLN - *Processamento de Linguagem Natural*

URL - *Uniform Resource Locator*

HTTP - *HyperText Transfer Protocol*

POS - *Part of Speech*

IR - *Information Retrieval*

BOW - *Bag of Words*

NB - *Naive Bayes*

REST - *Representational State Transfer*

JSON - *JavaScript Object Notation*

XML - *EXtensible Markup Language*

BSON - *Binary Script Object Notation*

Lista de Figuras

1.1	Número de usuários de redes sociais no mundo de 2010 a 2018 (em bilhões)	3
2.1	Funcionamento da API do Twitter do tipo REST.....	11
2.2	Funcionamento da API de streaming do Twitter	12
2.3	Processo de Análise de Sentimento em postagens do Twitter (tweets)	13
2.4	Representação de texto como um vetor de <i>features</i>	19
2.5	Representação de um documento de texto d , com <i>features</i> positivas e negativas, segundo o algoritmo de classificação baseado em centróides (Rocchio)	23
2.6	Representação do hiperplano encontrado pelo algoritmo SVM para um dado conjunto de treino, com a zona de buffer, ou margem máxima.....	24
4.1	Coleções de documentos armazenados em um banco de dados MongoDB	31
4.2	Comparação dos resultados das coletas em inglês e português	33
4.3	Separação dos conjuntos de dados a serem utilizados nos experimentos	35
4.4	Exemplos de tweets coletados em inglês	36
4.5	Processos e resultado da primeira etapa de tratamento dos dados.....	37
4.6	Exemplo de separação de frases integrantes de uma sentença que apresentam polaridades distintas.....	37
5.1	Resultados das execuções dos algoritmos de marcação e classificação de subjetividade	47
5.2	Médias das acurácias X médias dos tempos de execução dos algoritmos NB Multinomial e SVM em conjuntos de 50 mil sentenças	49
6.1	Funcionamento do processo k-fold cross-validation com $k = 10$	53
6.2	Divisão dos conjuntos em subconjuntos de treino e teste e subsequente tradução dos conjuntos de teste.....	56
6.3	Processos de treinamento dos classificadores e classificação dos subconjuntos traduzidos.....	57
6.4	Diagrama do funcionamento do experimento.....	61

Lista de Tabelas

2.1	Exemplos de n-gramas de uma dada sentença	15
2.2	Exemplos do resultado do processo de lemmatização em inglês e português	15
2.3	Exemplos do resultado do algoritmo de Porter para stemming em inglês.....	16
2.4	Alguns passos do algoritmo de Porter para stemming em inglês	17
4.1	Resultado da primeira rodada de coletas	32
4.2	Resultados das coletas nas duas linguagens e períodos de exibição dos títulos	33
4.3	Exemplos de features modificadas e/ou substituídas antes do processo de tradução	38
4.4	Utilização da expressão “no way” com diferentes polaridades	39
4.5	Utilização da expressão “can’t wait” com diferentes polaridades	40
4.6	Comparação de traduções de sentenças contendo a expressão “can’t wait”	40
4.7	Exemplos de palavras de polaridade conhecida e a mudança de polaridade com a adição da negação	42
4.8	Parte das listas de stop words em inglês e português.....	43
5.1	Exemplos de palavras e construções verbais que expressam opinião ou emoção em uma sentença textual.....	46
6.1	Resultados das classificações em 10-fold cross-validation através dos classificadores NB Multinomial e SVM nos conjuntos de dados irmãos	54
6.2	Resultados das classificações através dos os algoritmos NB Multinomial e SVM com os subconjuntos de treino e subconjuntos de teste construídos.....	55
6.3	Resultados das classificações nos classificadores NB Multinomial e SVM com subconjunto de treino em inglês e subconjuntos de teste traduzidos PT-EN	58
6.4	Resultados das classificações nos classificadores NB Multinomial e SVM com subconjunto de treino em português e subconjuntos de teste traduzidos EN-PT.....	58
6.5	Valores das médias das métricas de avaliação calculadas após classificação do conjunto PT-EN pelo classificador c2	62

Capítulo 1

Introdução

As redes sociais ganharam popularidade rapidamente entre as pessoas, organizações e outras entidades sociais, dando a seus usuários uma maneira fácil e rápida de interagir, comunicar e compartilhar conteúdos entre si. Essas redes têm crescido tanto que hoje fazem parte da vida dos seus usuários.

Diariamente, usuários de redes sociais publicam postagens sobre suas vidas, compartilham opiniões sobre vários tópicos diferentes e discutem problemas atuais, fazendo com que milhões de mensagens apareçam nessas redes. Ao passo que, cada vez mais, estes usuários postam suas opiniões sobre produtos e serviços que utilizam, ou expressam suas visões políticas[23] e religiosas[24], sites de *microblogging* tornam-se fontes valiosas de opiniões e sentimento das pessoas. Esses dados podem ser usados de forma eficiente para diversas áreas, como estudos de *marketing* ou sociais [20, 21, 22].

O número de usuários de plataformas de redes sociais cresce todos os dias [17], como pode ser observado na Figura 1.1 que apresenta um levantamento do número de usuários de redes sociais de 2010 a 2014, e uma projeção até 2018. Com esse crescimento e popularização dessas plataformas, mais e mais opiniões de usuários podem ser extraídas dessas fontes, a partir de postagens feitas diariamente expressando o que gostam ou desgostam sobre vários aspectos, incluindo produtos ou serviços, ou sobre suas próprias vidas.

O estudo e análise de redes sociais dão a habilidade de modelar muitos problemas reais complexos, por essa razão vem sendo muito utilizado na predição de soluções para problemas do mundo real. Nessas redes, usuários compartilham os mais diversos tipos de informações, gerando dados úteis para análise de vários aspectos, inclusive tendências de mercado. Neste contexto, temos várias empresas e organizações que usam as redes sociais para desenvolver estratégias de *marketing* de seus produtos e serviços.

A análise de redes sociais para uso em identificação de tendências de mercado mostrou muitas vantagens devido ao fato de que as mídias sociais são atualizadas com frequência e incluem resultados bastante imparciais [5]. Sites de *microblogging*, como Twitter [11], Facebook [15] e

Tumblr [16], são fontes ricas de dados para mineração de opinião e sentimento [14] sobre produtos e serviços.

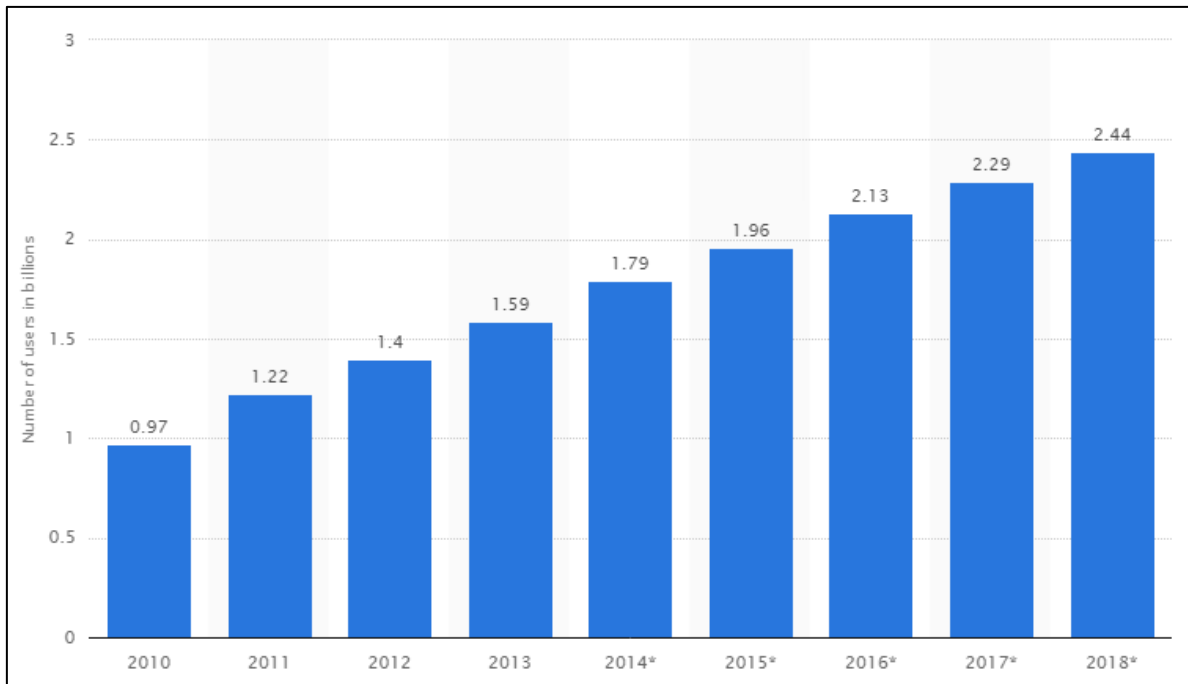


Figura 1.1 - Número de usuários de redes sociais no mundo de 2010 a 2018 (em bilhões)

A partir da mineração de opinião e análise de sentimento de postagens de usuários sobre determinados produtos ou serviços, podemos responder questões como:

- o que os consumidores pensam sobre determinado produto ou serviço?
- o sentimento dos consumidores sobre determinado produto ou serviço é positivo (ou negativo)?

As respostas destas perguntas são importantes para estratégias futuras de *marketing* de empresas, podendo prever o sucesso, ou não, nas vendas de um produto ou serviço.

1.1 Problema de Pesquisa

A pesquisa desenvolvida neste trabalho, começou com um escopo bem mais reduzido, como pode ser visto em [13], focando em uma análise de popularidade de marcas e aparelhos *smartphones* com base no número de menções dos mesmos realizadas pelos usuários da rede social Twitter. Para tanto, conjuntos de *tweets* foram coletados através de cinco conjuntos de palavras-chave distribuídas em cinco *queries* de busca. Os *tweets* coletados não foram filtrados por idioma, de forma que os resultados refletissem a popularidade global no ramo de *smartphones*.

Como evolução do trabalho, seria implementada a análise de sentimento para avaliar a polaridade das menções, se as marcas e aparelhos objetos de busca tinham popularidade negativa ou positiva. Nessa etapa, nos deparamos com a barreira do idioma e a seguinte questão:

- Como fazer com que essa análise de sentimento seja global, assim como a primeira análise realizada?

Para uma análise global de sentimento, todas as sentenças escritas em idiomas diferentes do inglês precisam ser traduzidos para este, de forma que haja um processo de classificação de sentimento em todo o conjunto de dados, e não em partes dele (uma classificação para cada idioma identificado).

Nos últimos anos, muitos estudos foram realizados para melhorar a análise de sentimento em vários idiomas. Pak e Paroubek [28] afirmaram que a mesma abordagem para o problema de classificação de sentimento em inglês, utilizando *machine learning*, especificamente com os algoritmos Naive Bayes e SVM, poderia ser implementada para outros idiomas com elevada acurácia, em projetos de classificadores multilinguais. Alguns trabalhos focaram na análise e classificação de sentimento para idiomas específicos, diferentes do idioma inglês, como português [29][30], chinês[31], idiomas do norte da Índia [32], etc., enquanto outros trouxeram abordagens para implementação de classificadores multilinguais [28][33], também trazendo idiomas selecionados.

Alguns poucos trabalhos utilizam motores de tradução automática para traduzir idiomas para o inglês e a partir daí realizar a classificação de sentimento. Duarte [29], por exemplo, implementou uma classificação de *tweets* na língua portuguesa utilizando a tradução destes para o inglês, conseguindo resultados satisfatórios. Entretanto, com a complexidade de certos

idiomas (morfologia, gírias, distância entre línguas de origem saxã das de origem latina, elementos culturais, etc):

- como garantir que o texto traduzido terá a mesma polaridade de sentimento final que o texto original?

Mesmo com o evidente avanço nos algoritmos de tradução automática, em especial o *Google Translator*¹ e o *Microsoft Bing Translator*², ainda há traduções realizadas por estes que modificam completamente o sentido e estrutura do texto original. Especialmente, quando nos referimos a textos extraídos de redes sociais como o Twitter, onde as mensagens postadas são rápidas e cheias de abreviações, gírias, falta de acentuação e pontuação.

Os poucos trabalhos que utilizaram a abordagem de aprendizagem de máquina supervisionado e dados traduzidos para o inglês, através de serviços de tradução automática, focaram apenas na classificação de sentimento [33], sem a preocupação com o impacto que a qualidade dessa tradução pode causar nos resultados finais, nem a diferença que o uso de diferentes sistemas de tradução automática pode fazer nos conjuntos de dados traduzidos. Com a preocupação da literatura acerca da utilização de tradutores automáticos em dados de texto e a visível evolução desses sistemas de tradução automática de texto, torna-se de extrema importância avaliar o impacto que essas traduções podem apresentar.

1.2 Objetivos

Como principal objetivo de pesquisa deste trabalho, podemos ressaltar o estudo da relevância da utilização de máquinas de tradução automática de texto em processos de classificação de sentimento de dados de texto, extraídos da plataforma Twitter, para idiomas diferentes do inglês. Como estudo de caso, dois serviços de tradução automática de texto, *Google Translate* e *Microsoft Bing Translate*, foram utilizados na composição de conjuntos de dados de treino e de teste para a linguagem português e subsequente construção de modelos de aprendizado de máquina utilizando os algoritmos de classificação Naive Bayes e SVM.

¹ Acessível em <https://translate.google.com/>

² Acessível em <https://www.bing.com/translator>

Primeiramente, um *crawler* foi desenvolvido utilizando a Twitter Search API para a coleta de dados de texto. Após vários testes de coleta, um contexto de pesquisa foi escolhido e palavras-chave foram selecionadas para a coleta de dados em inglês e português. Tendo em vista que as palavras-chave devem permitir que se consiga um número satisfatório de *tweets* tanto em inglês quanto em português, o contexto escolhido para a coleta de dados foi a franquia de quadrinhos, seriado de TV e videogames, *The Walking Dead*³.

Os classificadores em inglês e português foram validados através de *cross-validation* utilizando os conjuntos de treino para cada linguagem. Para a linguagem português o conjunto de treino foi traduzido através do *Google translate* e posteriormente, revisado e corrigido manualmente. Todos os conjuntos de dados foram pré-processados antes de passarem pelo processo de classificação de sentimento.

Compararemos se após a tradução, os classificadores conseguem marcar as sentenças de texto traduzidas para um idioma com a mesma classe de sentimento que as sentenças marcadas no outro idioma. A avaliação de desempenho da classificação de sentimento em ambos as linguagens se dará através das métricas: acurácia, precisão, *recall* e *F-measure*.

1.3 Relevância

Com crescimento rápido do interesse da comunidade de Processamento de Linguagem Natural (PLN) em desenvolver técnicas de análise de subjetividade e sentimento em texto, diferentes métodos são propostos para lidar com os diferentes tipos de texto e domínios, alcançando resultados satisfatórios para a linguagem inglês.

No desenvolvimento de classificadores multilinguísticos, pesquisadores da área de análise de sentimento relutam em utilizar sistemas de tradução de texto automática devido à baixa performance que estes costumavam apresentar. Nos últimos anos, o *Google Translate* e o *Microsoft Bing Translate* melhoraram bastante seus algoritmos de tradução, provendo notadamente, cada vez mais, traduções mais acuradas para linguagens de uso frequente.

Como citado anteriormente, poucos trabalhos estudaram o impacto que a qualidade da tradução do texto pode causar nos resultados finais da classificação de sentimento, tampouco a diferença

³ [https://en.wikipedia.org/wiki/The_Walking_Dead_\(franchise\)](https://en.wikipedia.org/wiki/The_Walking_Dead_(franchise))

que o uso de diferentes sistemas de tradução automática pode fazer nos conjuntos de dados traduzidos. Também não há um estudo desse tipo para a linguagem português, estudando o impacto da tradução automática de português para inglês, e inglês para português, em uma classificação simples de sentimento nas duas linguagens.

1.4 Estrutura da Dissertação

O restante deste trabalho está organizado da seguinte forma:

- No *Capítulo 2* é apresentada uma fundamentação teórica sobre a área de classificação de sentimento em dados de texto, destacando as principais técnicas utilizadas. Também é apresentada uma breve explicação sobre a rede social Twitter.
- No *Capítulo 3* são apresentados os trabalhos relacionados, destacando suas contribuições e semelhanças com este trabalho.
- No *Capítulo 4* são apresentados os processos de coleta de dados e pré-processamento de texto, explicando como foi escolhido o contexto das palavras-chave de busca, um pouco do funcionamento da Twitter Search API ⁴e o banco de dados utilizado, o MongoDB⁵. Por fim são detalhadas as etapas de pré-processamento de texto utilizadas neste trabalho.
- No *Capítulo 5* é apresentado o processo de classificação de sentimento, destacando os processos de marcação dos dados de treino, métricas de avaliação de desempenho, os algoritmos utilizados e comparação entre estes, e por fim, os cenários dos experimentos realizados de classificação de texto.
- No *Capítulo 6* são apresentados os experimentos realizados, bem como a discussão dos resultados obtidos.
- Por fim, no *Capítulo 7* são apresentadas as considerações finais do trabalho, suas contribuições e trabalhos futuros.

⁴ Acessível em: <https://dev.twitter.com/rest/public/search>

⁵ Acessível em: <https://www.mongodb.org/downloads>

Capítulo 2

Fundamentação Teórica

Neste capítulo é apresentado o embasamento teórico que norteia os principais domínios de conhecimentos envolvidos neste trabalho. Inicialmente, será apresentada brevemente a plataforma de *microblogging* e rede social, Twitter, falando um pouco da sua relevância como fonte de dados, e a Twitter Search API, utilizada na coleta de dados, destacando seus principais métodos. Em seguida, é apresentada uma descrição da área de classificação de sentimento, destacando os principais processos e técnicas propostas pela literatura na área relevantes ao nosso estudo.

2.1 A rede social Twitter

Criada com o objetivo de transformar a forma como mensagens eram trocadas pelos usuários em seus aparelhos celulares, a rede social e *microblogging* Twitter completou 10 anos de funcionamento. A ideia de seus criadores era a de um produto que girasse em torno do que as pessoas estivessem fazendo em um determinado momento, o *status* do usuário.

O Twitter é uma rede social *online* que permite que usuários enviem (postem) e leiam mensagens curtas, chamados *tweets*, em tempo real. Através dos *tweets*, que permitem um limite de apenas 140 caracteres de texto, o Twitter oferece um serviço rápido e dinâmico de postagem de mensagens, o que corrobora para sua utilização por usuários no mundo inteiro para compartilhar notícias e informações de todos os tipos. Catástrofes, eventos sociais, acontecimentos mundiais, etc, são amplamente discutidos através de mensagens em tempo real. Políticos utilizam o Twitter para conhecer seu eleitorado, e este expõe suas ideias e opiniões. Empresas utilizam a rede social para se aproximar de seus clientes, divulgando seus produtos, ou ainda, monitorando a popularidade de seus serviços e produtos através da opinião dos usuários.

2.1.1 Twitter como fonte de dados para classificação de sentimento

Sendo uma rede social onde seus usuários postam mensagens diariamente sobre conteúdos diversos, o Twitter representa uma fonte de dados bastante variada para classificação de sentimento. A literatura da área de análise e classificação de sentimento, ou mineração de opinião, frequentemente utiliza dados coletados da rede social em suas pesquisas e experimentos. Existem alguns dados interessantes sobre o Twitter, informados pelo próprio site do mesmo:

- Cerca de 310 milhões de usuários ativos por mês
- 83% de usuários ativos em plataformas móveis
- 79% das contas são de fora dos Estados Unidos
- Mais de 40 linguagens suportadas

O Twitter disponibiliza APIs para que desenvolvedores possam coletar, ou manipular, dados de *tweets* para pesquisa. As postagens do Twitter, *tweets*, possuem dezenas de atributos, variando de data da postagem até informações do usuário que postou a mensagem.

2.1.2 Twitter API

As APIs públicas de acesso ao Twitter são de dois tipos: REST e *streaming*. A API pública baseada na arquitetura REST, depende de um protocolo que permite *cache*, comunicação cliente-servidor e sem monitoramento de estado (geralmente, HTTP é o protocolo utilizado [39]). As APIs REST fornecem acesso às funções de leitura e escrita de dados do Twitter, além de identificar aplicações, que implementam o Twitter, e usuários utilizando OAuth⁶, e as respostas às requisições são em JSON⁷. Este é um formato aberto que utiliza texto legível para transmitir objetos de dados que consistem em pares do tipo atributo-valor. É utilizado principalmente para transmitir dados entre um servidor e uma aplicação web, como uma alternativa ao uso do formato XML⁸.

OAuth é um protocolo aberto que permite autorização segura em aplicações web, móveis e *desktop* [38]. O Twitter utiliza a versão OAuth 1.0A para prover dois tipos de acesso autorizado à sua API:

⁶ Disponível em: <http://oauth.net/>

⁷ Disponível em: <http://www.json.org/>

⁸ Disponível em: <http://www.w3schools.com/xml/>

- *Autenticação do tipo aplicação - usuário*: A forma mais comum de autenticação de recursos na implementação do OAuth1.0A do Twitter. A requisição assinada identifica a identidade da aplicação e a identidade juntamente com permissões concedidas ao usuário final o qual está fazendo as requisições à API, representadas pelo *token* de acesso do usuário.
- *Autenticação do tipo aplicação apenas*: A aplicação faz requisições à API para si mesma, sem o contexto do usuário.

A API pública de acesso ao Twitter do tipo *streaming* oferecem a possibilidade de monitoramento ou processamento de tweets em tempo real, sendo essa API a indicada para mineração de dados. Uma vez que a conexão com um *stream* é estabelecida, não será necessária nenhuma interação subsequente. As *streams* podem ser filtradas por, por exemplo, palavras-chave, localização do usuário ou linguagem. A API fornece toda essa informação no formato JSON. A autenticação da aplicação que se conectará à API de *streaming* é realizada através de 2 chaves de acesso, *consumer key (API key)* e *consumer secret (API secret)* e 2 *tokens* de acesso, *access token* e *access token secret*, fornecidas pelo Twitter por aplicação. O *tokens* de acesso permitem que a aplicação faça requisições à API em nome da conta do usuário na qual a dada aplicação se encontra cadastrada.

Para entender a diferença entre as APIs do Twitter, REST e *streaming*, a Figura 2.1 apresenta o funcionamento da API REST e a Figura 2.2 apresenta o funcionamento da API *streaming*.

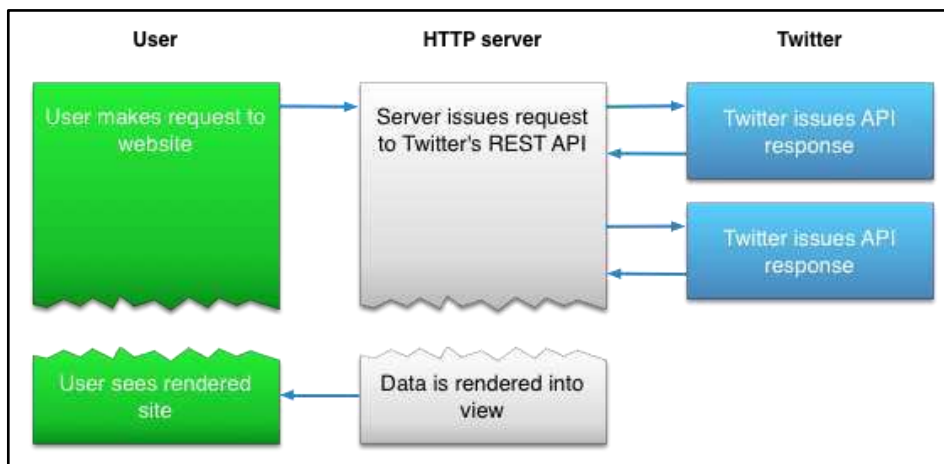


Figura 2.1 - Funcionamento da API do Twitter do tipo REST [9]

Conectar à uma API de *streaming* requer a permanência de uma conexão HTTP aberta e persistente. E, em muitos casos isso envolve uma estruturação diferente de uma aplicação que se conecta à uma API REST. Uma aplicação web que aceita requisições do usuário, por exemplo, realiza uma ou mais requisições à API do Twitter, para então formatar e imprimir o resultado para o usuário, como resposta à requisição inicial do usuário, como pode ser visto na Figura 2.1. Já uma aplicação que se conecta à API de *streaming* não poderá estabelecer uma conexão em resposta à uma requisição realizada por um usuário. O código que mantém a conexão de *streaming*, geralmente, roda em um processo separado do processo que lida com requisições HTTP, como pode ser visto na Figura 2.2.

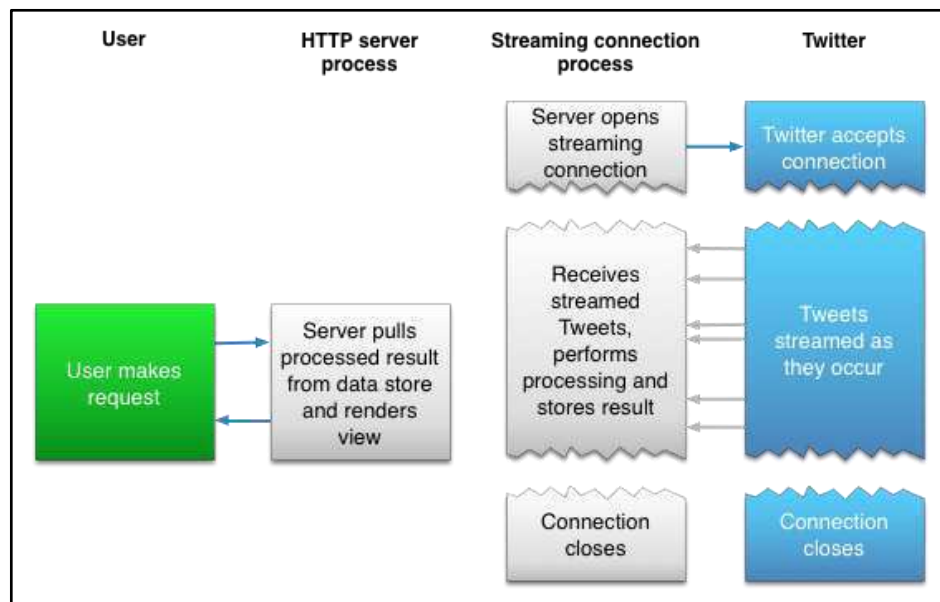


Figura 2.2 - Funcionamento da API de streaming do Twitter [9]

O processo de *streaming* captura os *tweets* de entrada e executa qualquer processo de análise, filtragem e/ou agregação necessária antes de armazenar os resultados em um banco de dados. O processo que lida com requisições HTTP realiza buscas (*queries*) nos dados armazenados por resultados em resposta à requisições do usuário.

2.2 Classificação de sentimento

Análise de sentimento, ou mineração de opinião, é uma área de estudo computacional sobre opiniões e emoções de pessoas sobre determinada entidade, esta pode ser um tópico, um indivíduo ou um evento. Apesar de análise de sentimento e mineração de opinião apresentarem um significado comum, alguns pesquisadores da área afirmam que ambas apresentam noções um pouco diferentes. Mineração de opinião extrai e analisa opiniões de pessoas sobre uma entidade, enquanto que análise de sentimento identifica o sentimento expresso em um texto e o analisa.

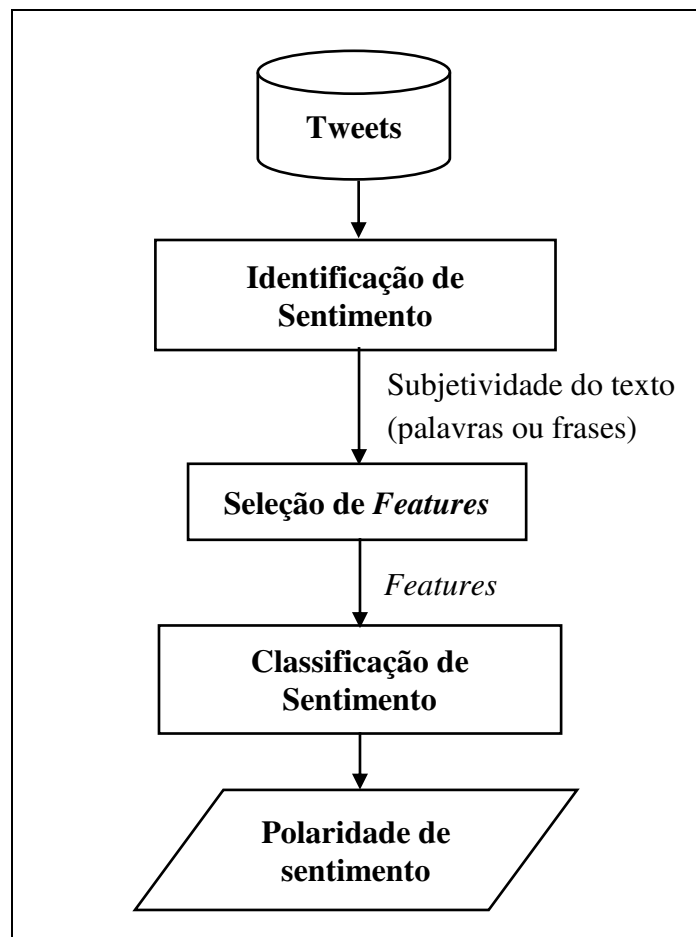


Figura 2.3 - Processo de Análise de Sentimento em postagens do Twitter (*tweets*)

Finalmente, como podemos observar na Figura 2.3, a análise de sentimento pode ser considerada um processo de classificação, onde o objetivo é encontrar opiniões, identificar o

sentimento que estas expressam e em seguida, classificar a polaridade do sentimento identificado.

2.2.1 Níveis de Classificação de Sentimento

Há três níveis principais de classificação de sentimento em texto:

- Nível de documento
- Nível de sentença
- Nível de aspecto

A classificação de texto em nível de documento realiza classificação de opinião de um documento de texto, como uma opinião positiva, negativa ou como um sentimento (emoções como raiva, tristeza, alegria, etc.), considerando todo o documento de texto como uma unidade de informação que trata de um único tópico. A classificação em nível de sentença classifica o sentimento expresso em cada sentença de texto, identificando primeiramente, se a sentença é subjetiva (opinativa) ou objetiva (informativa). Caso a sentença seja identificada como subjetiva, a classificação de texto em nível de sentença determinará se essa sentença expressa opinião (polaridade do sentimento) positiva ou negativa. Embora a identificação de subjetividade seja a mais indicada na literatura para encontrar expressões de sentimento em texto, alguns autores [10] apontam que estas expressões não necessitam ser subjetivas em sua natureza. Na verdade, as classificações em nível de sentença e em nível de documento de texto não apresentam uma diferença fundamental, já que sentenças podem ser vistas como documentos de texto reduzido [35].

A classificação de texto em nível de documento ou de sentença, não oferece todos os detalhes necessários das opiniões, em todos os aspectos da entidade, que são necessários em muitas aplicações. A classificação em nível de aspecto classifica o sentimento com respeito a aspectos específicos das entidades. Primeiramente, as entidades e seus aspectos são identificados. Diferentes opiniões podem ser atribuídas a aspectos diferentes de uma mesma entidade.

2.2.2 Seleção/Extração de *features* de texto

O primeiro passo em um problema de classificação de sentimento é a seleção e extração de *features* de texto. As *features* mais comuns são:

- *Presença e frequência de termos*: Essas *features* podem ser unigramas ou n-gramas e suas contagens de frequência. Dessa maneira, é possível obter o peso binário das palavras (0 se a palavra está presente ou 1, se não está presente) ou pode-se usar os pesos da frequência dos termos para indicar a importância relativa das *features*. Na área de linguística computacional, um n-grama é uma sequência de n itens consecutivos de uma dada sequência de texto ou discurso. Esses itens podem ser fonemas, sílabas, letras, palavras ou pares de base, de acordo com a aplicação. Na Tabela 2.1 são apresentados alguns exemplos de n-gramas compostos por palavras consecutivas de uma sentença, que é a representação utilizada no nosso estudo.

Sentença original	to be or not to be
unigramas	to, be, or, not, to, be
bi-gramas	to be, be or, or not, not to, to be
tri-gramas	to be or, be or not, or not to, not to be

Tabela 2.1 - Exemplos de n-gramas de uma dada sentença

- *Parts of Speech (POS)*: adjetivos são indicadores de opinião importantes, logo encontrá-los aponta a subjetividade de um dado texto.
- *Palavras ou frases opinativas*: algumas palavras são muito comuns em expressões de opinião, como por exemplo, “odeio” para opiniões negativas e “amo” para opiniões positivas. Entretanto, também existem frases completas que não possuem palavras indicadoras de opinião, mas que expressam uma opinião. Por exemplo, “tive que vender um rim” que indica sentimento negativo, ou a famosa frase do teatro “quebre a perna” que indica sentimento positivo.
- *Termos com capitalização de letras*: Há termos que apresentam significados diferentes com a capitalização das letras, por exemplo, em inglês, “US” é diferente de “us”. No caso de postagens como os *tweets*, a verificação de capitalização de letras em *hashtags* para separação de palavras contidas nesta.

Inglês
<p><i>am, are, is → be</i></p> <p><i>car, cars, car's, cars' → car</i></p> <p><i>"the boy's cars are different colors" → the boy car be different color</i></p>
Português
<p><i>quero, queres, quer, querem → querer</i></p> <p><i>carros, carro → carro</i></p> <p><i>"os carros do garoto tem cores diferentes" → o carro do garoto ter cor diferente</i></p>

Tabela 2.2 - Exemplos do resultado do processo de lemmanização em inglês e português

- *Tratamento de negações*: os termos indicadores de negação podem modificar a orientação de opinião, como por exemplo, “não é bom” que é equivalente a “ruim”.
- *Remoção de Stop Words*: termos que não tem relevância para a computação da polaridade de sentimento de um documento de texto. Geralmente são artigos e pronomes na linguagem em que o texto é escrito.
- *Lemmanização*: do inglês *lemming*, é a tarefa de reduzir uma palavra à sua forma base, ou radical. Processo dependente da linguagem a qual o texto se encontra escrito [40], muito usado para redução de verbos aos seus radicais. Podemos ver na Tabela 2.2 alguns exemplos de lemmanização de verbos em inglês e português, termos em inglês e o processo realizado em uma sentença inteira em inglês.
- *Stemming*: tarefa em IR de reduzir termos aos seus *stems*, ou radicais (raiz). Assim como a lemmanização, é dependente da linguagem em que o texto se encontra escrito [40]. O algoritmo de Porter é o *stemmer* mais comum para a língua inglesa. A Tabela 2.3 apresenta exemplos de *stemmings* de termos e sentenças em inglês, ao passo que na Tabela 2.4 podemos entender como funciona o processo de *stemming* através de exemplos de resultados de algumas das etapas do algoritmo de Porter. Ressaltando que as substituições propostas só ocorrem em termos onde em sua parte anterior há a

presença de, pelo menos, uma vogal, como podemos ver nos exemplos de resultado dos passos 1b e 3.

Termos ou sentença originais	Resultado do processo de <i>stemming</i>
automate, automates, automatic, automation	Automatic
for example compressed and compression are both accepted as equivalent to compress	for example compress and compress ar both accept as equal to compress

Tabela 2.3 - Exemplos do resultado do algoritmo de Porter para stemming em inglês

Passo	Regra	Exemplos de Resultado
1a	<i>sses</i> → <i>ss</i>	<i>caresses</i> → <i>caress</i>
	<i>ies</i> → <i>i</i>	<i>ponies</i> → <i>poni</i>
	<i>ss</i> → <i>ss</i>	<i>caress</i> → <i>caress</i>
	<i>s</i> → \emptyset	<i>cats</i> → <i>cat</i>
1b	<i>ing</i> → \emptyset	<i>walking</i> → <i>walk</i> <i>sing</i> → <i>sing</i>
	<i>ed</i> → \emptyset	<i>plastered</i> → <i>plaster</i> <i>bed</i> → <i>bed</i>
2	<i>ational</i> → <i>ate</i> <i>izer</i> → <i>ize</i> <i>ator</i> → <i>ate</i>	<i>relational</i> → <i>relate</i> <i>digitizer</i> → <i>digitize</i> <i>operator</i> → <i>operate</i>
3	<i>al</i> → \emptyset	<i>revival</i> → <i>reviv</i>
	<i>able</i> → \emptyset	<i>adjustable</i> → <i>adjust</i> <i>able</i> → <i>able</i> <i>table</i> → <i>table</i>
	<i>ate</i> → \emptyset	<i>activate</i> → <i>activ</i> <i>fate</i> → <i>fate</i>

Tabela 2.4 - Alguns passos do algoritmo de Porter para stemming em inglês

2.2.2.1 Métodos para seleção de features:

Os métodos aplicados na seleção de *features* se dividem em duas categorias: o primeiro à base de dicionários de termos léxicos que necessitam de anotações manuais. Essa abordagem normalmente começa com um conjunto de palavras ‘semente’, este conjunto é inicializado através de detecção de sinônimos ou de fontes *online* para obter um dicionário maior. Esse método apresenta muitos problemas. M. Baumgarten et al. [18] afirmaram que essas dificuldades se encontram, por exemplo, no fato de palavras simbolizarem unigramas apenas, tornando mais difícil a extração de opinião ou emoção de uma estrutura de texto mais complexa. Isso pode ser observado na sentença “Um leitor achará difícil não recomendar esse livro”, onde o unigrama “recomendar” normalmente é tido como um indicador positivo, em respeito à polaridade de uma frase. Entretanto, como está precedido por “não”, que indica uma negação de “recomendar”, deveria levar à uma classificação de sentimento negativa. Ainda assim, a primeira parte da frase “Um leitor achará difícil” nega novamente, apontando para um sentimento positivo para o livro em questão. O método de classificação baseado em dicionários de termos léxicos não chegaria, muito provavelmente, à tal conclusão.

A segunda abordagem para seleção de *features* utiliza métodos estatísticos que são automáticos e são os mais frequentemente utilizados para a tarefa de seleção de features.

As técnicas de seleção de features tratam os documentos de texto como agrupamentos de palavras (ou BOWs), ou como uma *string* que contém a sequência de palavras do documento de texto. BOWs são mais usadas pois são mais simples de serem implementadas no processo de classificação. A etapa de seleção de features mais comum é a remoção de stop words e stemming (redução de uma palavra ao seu radical ou palavra raiz).

2.2.3 Representação do Texto

Documentos, que são geralmente *strings* de caracteres, precisam ser transformados em uma representação que seja adequada para o algoritmo de classificação de texto. Cada palavra distinta w_i em um documento d , corresponde à uma *feature* com $count(w_i, d)$ que representa o número de vezes que a palavra w_i ocorre no documento d . A Figura 2.4 apresenta a representação de um documento de texto como um vetor de *features*. Para evitar tamanhos

desnecessariamente grandes de vetores de *features*, as palavras de um documento são consideradas *features* apenas se ocorrerem pelo menos três vezes em um conjunto de dados de treino e se não estiverem listadas como *stop words* [37].

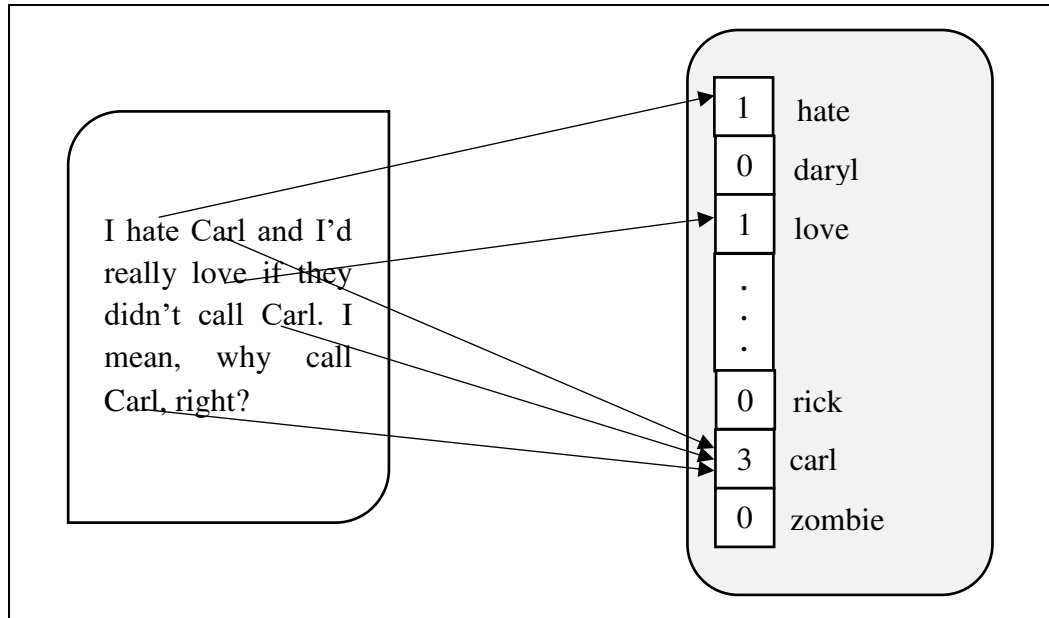


Figura 2.4 - Representação de texto como um vetor de *features*

2.2.4 Técnicas de classificação de texto

As técnicas de classificação de sentimento podem ser divididas em: abordagem de aprendizado de máquina, abordagem de dicionários de termos léxicos e abordagem híbrida.

A abordagem léxica baseia-se no uso de uma coleção de termos com sentimentos conhecidos e pré-compilados, e pode utilizar dicionários, ou métodos estatísticos ou semânticos para identificar polaridade de sentimento. A abordagem híbrida, como o nome indica, combina a abordagem léxica com a abordagem de aprendizado de máquina.

Em nosso trabalho, utilizaremos a abordagem de aprendizado de máquina para a classificação de texto, portanto daremos uma ênfase maior às suas técnicas.

2.2.4.1 Abordagem de aprendizado de máquina

As técnicas de classificação de texto implementam algoritmos de aprendizado de máquina. Geralmente, o problema de classificação de texto é definido da seguinte forma: temos um conjunto de textos de treino $T = \{t_1, \dots, t_n\}$ e um conjunto de classes de classificação $C = \{c_1, \dots, c_n\}$. Cada texto $t_i \in T$ é marcado com uma classe de classificação $c_j \in C$. O conjunto de treino T é aplicado ao classificador e um modelo de classificação é gerado. O modelo de classificação relaciona às *features* presentes em um texto $t_i \in T$ à uma classe de classificação $c_j \in C$. Esse modelo de classificação é utilizado para classificar novos textos os quais a classe de classificação $c_j \in C$ não é conhecida.

As técnicas que implementam aprendizado de máquina podem ser divididas em métodos de aprendizado supervisionado e não supervisionado. O método de aprendizado supervisionado requer dados de texto de treino previamente marcados com as respectivas classes de classificação. Já os métodos de aprendizado não supervisionados não necessitam desses dados, sendo utilizados quando não é possível encontrar tais dados de treino.

2.2.4.1.1 Método de aprendizado supervisionado

Na literatura há várias abordagens de classificação de texto utilizando classificadores supervisionados. Walaa Medhat et al. [36] apresentou uma coletânea das principais técnicas e algoritmos de classificação utilizados na literatura nos últimos anos. Para a classificação de sentenças curtas e *tweets* foi constatado que as técnicas mais utilizadas, consequentemente que apresentam melhor performance, são as técnicas com classificadores que implementam o algoritmo SVM e os baseados no algoritmo Naive Bayes. Neste trabalho, utilizaremos essas duas técnicas citadas de aprendizado supervisionado, as quais serão explicadas mais adiante.

2.2.4.1.1 Classificadores Probabilísticos

A partir de um conjunto de dados de entrada, classificadores probabilísticos conseguem prever uma distribuição de probabilidade para um conjunto de classes de classificação, com um certo grau de certeza, não só apontando simplesmente a classe à qual um elemento do conjunto de entrada deve pertencer.

2.2.4.1.1.1 Naive Bayes (NB)

O modelo de classificação Naive Bayes computa a probabilidade final de uma dada classe, baseado na distribuição das palavras no documento (ou sentença) de texto. O algoritmo Naive Bayes utiliza a técnica de BoW e admite a total independência das palavras em um texto, logo, a posição destas no texto é ignorada. O algoritmo baseia-se no teorema de Bayes para calcular a probabilidade de um dado conjunto de *features* $F = \{f_1, \dots, f_n\}$ pertencer à uma classe de classificação $c_j \in C$, como podemos ver nas Equações 2.1 e 2.2.

$$P(c_j|F) = \frac{P(c_j) * P(F|c_j)}{P(F)} \quad (2.1)$$

$$P(c_j|F) = \frac{P(c_j) * P(f_1|c_j) * \dots * P(f_n|c_j)}{P(F)} \quad (2.2)$$

$P(c_j)$ é a probabilidade de uma classe $c_j \in C$ ocorrer ou de que uma *feature* $f_i \in F$ aleatória seja associada à essa classe. $P(F|c_j)$ é a probabilidade de um dado conjunto de *features* F ser classificado em uma classe $c_j \in C$. $P(F)$ é a probabilidade de um dado conjunto de *features* F ocorrer. A Equação 2.2 reescreve a Equação 2.1 de acordo com a hipótese do teorema de Bayes que afirma que todas as *features* são independentes. Sendo assim, cada $f_i \in F$ possui uma probabilidade condicional de ocorrer. Pak e Paroubek [28] adaptaram a equação do Teorema de Bayes para o problema específico de classificação de sentimento de mensagens, ou postagens, do Twitter, como podemos ver na Equação 2.3.

$$P(s|M) = \frac{P(s) * P(M|s)}{P(M)} \quad (2.3)$$

M é uma mensagem (*tweet*) que, de maneira análoga à Equação 2.2, representa um conjunto de *features*. s é um sentimento $s \in S$. Em nosso trabalho, o conjunto de sentimento possui dois elementos, positivo e negativo, logo, temos $S = \{s_1, s_2\}$. Partindo dessas informações, podemos reescrever a Equação 2.3, adaptando-a para o problema de classificação de sentimento especificamente.

$$P(s|F) = \frac{P(s) * P(f_1|s) * ... * P(f_n|s)}{P(F)} \quad (2.4)$$

2.2.4.1.1.2 Naive Bayes Multinomial

O algoritmo NB, apresentado anteriormente, modela um documento de texto através da presença ou ausência de *features*. Já o algoritmo NB multinomial, que é uma versão especializada do algoritmo NB, modela um documento de texto através da contagem das *features*, ajustando os cálculos base. A Equação 2.5 apresenta como é calculada da probabilidade condicional de uma *feature* f dada uma classe c .

$$P(f|c) = \frac{\text{count}(f, c) + 1}{\text{count}(c) + |V|} \quad (2.5)$$

$\text{count}(f, c)$ é a contagem do número de repetições de uma dada *feature* f em todos os documentos de texto marcados com a classe c . $\text{count}(c)$ representa o número total de *features* marcadas com a classe c . E $|V|$ representa o vocabulário de *features*, ou seja, todas as *features* distintas presentes em todos os documentos. A contagem de *features* faz com que o algoritmo NB multinomial obtenha uma performance superior ao algoritmo NB.

2.2.4.1.2 Classificadores lineares

Métodos de classificação linear de texto se baseiam no valor de uma combinação linear de *features* para gerar uma decisão de classificação.

2.2.4.1.2.1 Support Vector Machine (SVM):

Um modelo de classificação SVM representa o conjunto de dados de treino como pontos mapeados no espaço. Os documentos do conjunto de treino são separados pela classe em que estão classificados, de forma que são divididos por um espaço vazio. Para entender como o algoritmo SVM funciona, usaremos como base o algoritmo de classificação em centróides (Rocchio). Onde devemos ter duas categorias de classificação (positivo e negativo, por exemplo) e um documento de texto d que é um vetor de *features*. O texto deve ser linearmente separável, ou seja, muitas *features*. Na Figura 2.5 podemos ver a representação do algoritmo de centróides em um documento de texto d . c_- e c_+ são os centróides das categorias negativo e positivo, respectivamente.

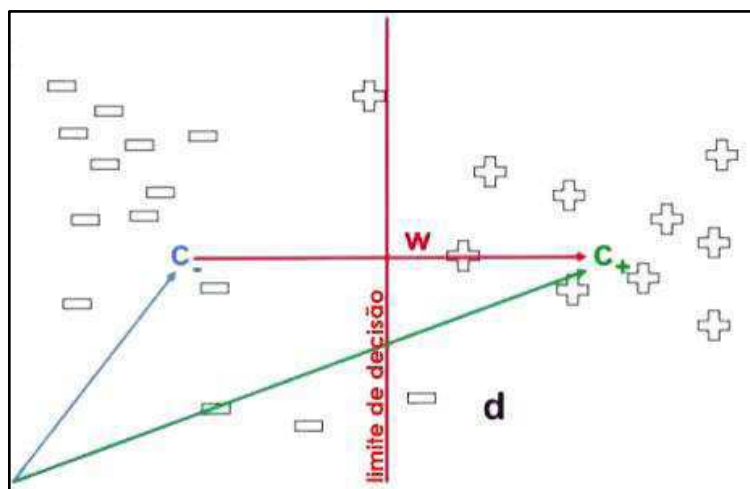


Figura 2.5 - Representação de um documento de texto d , com *features* positivas e negativas, segundo o algoritmo de classificação baseado em centróides (Rocchio)

A linha vertical representa o limite de decisão entre as categorias de classificação. Se a similaridade entre os centróides c_+ do documento de texto d for maior que a similaridade entre os centróides c_- , também do documento de texto d , este é classificado como da categoria positiva. Do contrário, d é classificado como negativo. \vec{w} é o vetor distância entre os centróides

c_- e c_+ e deve ser perpendicular ao limite de decisão. Temos que a regra de decisão pode ser representada por $d^T c_+ - d^T c_- > 0$ ou $d^T w > 0$, onde $w = c_+ - c_-$.

Entretanto, novos dados a serem classificados podem estar em qualquer posição no plano, portanto é necessária uma margem maior, ou uma zona de *buffer* maior, ao redor do limite de decisão. O algoritmo de classificação SVM tem como objetivo encontrar o melhor hiperplano h que represente a maior separação entre as duas classes de classificação. Para tanto, o vetor normal \vec{w} , perpendicular a h , precisa ser encontrado. A Figura 2.6 apresenta a representação de um hiperplano h encontrado pelo algoritmo SVM, que separa as duas classes de classificação com margem máxima. Os dados que se encontram mais próximos do hiperplano são chamados de vetores de suporte.

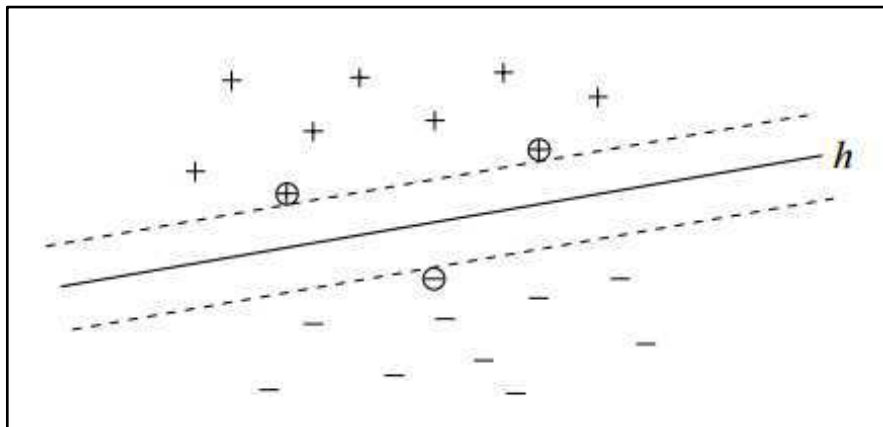


Figura 2.6 - Representação do hiperplano encontrado pelo algoritmo SVM para um dado conjunto de treino, com a zona de buffer, ou margem máxima

A tarefa de encontrar o melhor hiperplano é um problema de otimização onde é necessário minimizar o tamanho do vetor normal \vec{w} para maximizar a separação entre as duas classes de classificação, ou seja, encontrar a margem máxima ao redor de \vec{w} . Dessa forma, \vec{w} representa uma combinação ótima de *features*.

O objetivo do algoritmo SVM é minimizar classificações incorretas, ou seja, aumentar a acurácia de classificação, maximizando a distância entre as duas classes de classificação. Entretanto, SVM pode apresentar lentidão ao treinar grandes conjuntos de dados.

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados, dentre todos os pesquisados, os principais trabalhos relacionados à abordagem utilizada no presente estudo. Os trabalhos referem-se à análise e classificação de sentimento utilizando técnicas de aprendizado de máquina, bem como soluções para classificação em linguagens diferentes do inglês. Em nossa pesquisa, não encontramos estudos que realizassem comparação entre a classificação de conteúdo nativo em português e conteúdo traduzido por máquina de português para inglês, e vice-versa.

Com base na revisão de literatura realizada, os trabalhos relacionados foram separados em abordagens para análise e classificação de sentimento em inglês e abordagens para classificação de sentimento em outras linguagens, os quais são descritos e discutidos a seguir.

3.1 Classificação de Sentimento utilizando a plataforma Twitter como fonte de dados de texto

Pak e Paroubek [28] utilizaram a Twitter Search API para coletar dados, através de *queries* de busca baseadas em *emoticons* de felicidade - “:”)”, “=:)”, “=:D”, etc - e de tristeza – “:((", “=((", etc. Os *tweets* coletados contendo *emoticons* de felicidade foram classificados como positivos e os contendo *emoticons* de tristeza foram classificados como negativos, ambos os conjuntos de *tweets* foram utilizados para treinar um classificador para sentimento positivo e negativo. Para o conjunto de postagens objetivas de texto, foram coletados *tweets* de contas de 44 jornais.

As seguintes etapas foram realizadas na preparação dos *tweets*:

1. Remoção de URLs, menções de usuários, nomes de usuários, RTs.
2. Separação do texto em *tokens* formando um *bag of words*.
3. Remoção de *stop words*. Os autores removem apenas artigos (“a”, “an”, “the”).

4. Construção de *n-grams* a partir de palavras consecutivas e negações. A exemplo da sentença “I do not like fish” que forma dois *bigrams*: “I do+not”, “do+not like”, “not+like fish”.

Os classificadores Naive Bayes Multinomial e SVM foram utilizados, porém o primeiro obteve melhor performance. Os resultados mostraram que o tratamento de palavras negativas (construção de *n-grams*) aumentou a acurácia. Os experimentos também mostraram que o aumento do conjunto de dados de treino aumenta a acurácia, mas que em certo ponto, esse aumento do conjunto de treino não será suficiente para melhorar a performance da classificação.

Os autores afirmam que a abordagem apresentada no trabalho pode ser utilizada para qualquer outra linguagem diferente do inglês, valendo-se do fato que a API do Twitter permite a especificação da linguagem dos *tweets* capturados pela *query* de busca. Entretanto, a linguagem é especificada pelo usuário em sua conta do Twitter, de forma que essa linguagem não necessariamente é a mesma em que o usuário escreve suas postagens. Esse fato foi comprovado em nosso processo de coleta de dados, onde a *query* de busca especificou a linguagem como inglês, porém muitos *tweets* em espanhol, francês, e etc, também foram coletados. Esse trabalho serve como referência para muitos outros estudos em classificação de sentimento, inclusive o nosso, por trazer uma abordagem simples e eficaz para o problema.

Yu e Hatzivassiloglou [34] apresentaram uma solução para identificar se uma sentença pertence à classe de opinião (subjativa) ou à classe de fato (objetiva). Um classificador é treinado com um conjunto de dados de treino completo, o modelo de classificação construído é utilizado para classificar o mesmo conjunto de dados de treino. As sentenças do conjunto de dados onde as marcações de classe (opinião ou fato) não correspondem às classificadas são removidas do conjunto de treino. O conjunto de dados de treino resultante é utilizado para treinar outro classificador, o modelo de classificação irá classificar o mesmo conjunto de dados de treino. O processo se repete até que nenhuma sentença seja classificada com classe diferente da previamente marcada. Esse método alcançou uma acurácia de 80% a 90% para a classe opinião e 50% para a classe fato com classificadores Naive Bayes, e é uma alternativa ao uso de marcações POS para classificação de subjetividade em sentenças de texto.

Há poucos trabalhos na literatura que exploram a classificação de *tweets* em português. A.L.F. Alves et al. [30] utilizaram a mesma abordagem de Yu e Hatzivassiloglou na classificação de subjetividade e posterior classificação de polaridade de *tweets* em português. O processo foi

realizado em duas etapas, na primeira os *tweets* foram classificados quanto à sua subjetividade, na segunda os *tweets* subjetivos foram classificados quanto à sua polaridade. Conforme as recomendações da literatura na área de classificação de sentimento, os classificadores SVM e Naive Bayes foram utilizados. Ao contrário da maioria dos trabalhos na literatura, os autores mediram o desempenho das classificações não só com a acurácia, mas também com as métricas IR - Precisão, *Recall* e *F-measure*. A classificação de subjetividade + polaridade utilizando Naive Bayes alcançou a *F-measure* de 0.791 e acurácia de 72.7%. Já a classificação de subjetividade + polaridade utilizando SVM alcançou a *F-measure* de 0.873 e acurácia de 80%, confirmando o que é dito por vários autores na literatura ao atribuírem uma melhor performance em classificação de texto para a técnica SVM.

Nosso trabalho se assemelha com o de A.L.F. Alves et al. e, conseqüentemente, com o de Yu e Hatzivassiloglou, por não utilizar marcações POS na identificação de subjetividade das sentenças de texto (*tweets*). O conjunto de dados inicial de 7000 *tweets* em inglês foi marcado manualmente em subjetivo, ou objetivo, e este conjunto foi utilizado para treinar um classificador SVM. O modelo gerado foi utilizado para classificar a subjetividade do mesmo conjunto de *tweets*. O processo se repetiu até que a marcação classificada para nenhum *tweet* diferisse da marcação prévia. O conjunto final de dados, o qual terá suas sentenças de texto classificadas por polaridade, positiva ou negativa, é constituído das sentenças classificadas como subjetivas.

3.2 Classificadores Multilinguísticos utilizando dados de texto traduzidos por máquina

Balahur e Turchi [33] abordaram o problema da distinção na detecção de sentimento em três linguagens diferentes – francês, alemão e espanhol – utilizando três serviços de tradução automática – Bing, Google e Moses⁹. A partir destes três serviços de tradução, obtiveram três conjuntos traduzidos do inglês para cada linguagem citada, com qualidades de tradução variadas. Os conjuntos de teste foram traduzidos através do serviço de tradução automática do Yahoo¹⁰

⁹ Disponível em: <http://www.statmt.org/moses/>

¹⁰ Acessível em: <https://www.yahoo.com/>

e depois corrigido à mão por uma pessoa nas três linguagens. A maioria das sentenças não apresentou polaridade negativa nem positiva, estas sentenças neutras foram eliminadas.

Para a classificação de sentimento foi utilizado o método estatístico SVM SMO (*Support Vector Machines Sequential Minimal Optimization*), que segundo os autores, é o método de *machine learning* que a literatura confirma ser o mais apropriado para a tarefa proposta. Em seus experimentos, as sentenças foram representadas através dos unigramas e bigramas encontrados no conjunto de treino. O classificador SVM SMO foi treinado para cada um dos serviços de tradução propostos, utilizando os dados traduzidos através destes, e um modelo foi gerado para cada linguagem proposta em cada classificador.

Na primeira rodada de experimentos, testaram os classificadores com dados de teste traduzidos automaticamente pelos respectivos serviços de tradução, depois testaram os dados traduzidos pelo Yahoo e corrigidos manualmente por uma pessoa. Os resultados mostraram que reduzindo a variância nos modelos estimados, produz-se um efeito positivo na performance do classificador, aumentando o *F-score*. Essas melhorias aumentam ainda mais com os dados em alemão, devido à má qualidade da tradução, aumentando a variância dos dados.

Na segunda rodada, combinaram os dados traduzidos pelos três serviços de tradução para cada uma das três linguagens. Os testes foram feitos com os dados traduzidos pelo Yahoo e corrigidos manualmente por uma pessoa. Os resultados mostraram que adicionado todos os dados de treino traduzidos no classificador, drasticamente aumenta o nível de ruído dos dados de treino, criando efeitos nocivos na performance do classificador: cada classificador perde sua capacidade discriminatória.

Os autores constataram que claramente os resultados dependem da performance da tradução. Com exceção da linguagem espanhol, cuja os classificadores foram capazes de aprender através dos dados de treino traduzidos e tentar marcar apropriadamente as amostras de teste, os dados traduzidos para alemão e francês contém tanto ruído que seus classificadores não foram capazes de apropriadamente aprender a informação correta das classes positiva e negativa, conseqüentemente marcando erroneamente os dados de teste (praticamente todos os dados de teste marcados em uma classe apenas).

Não foram encontradas evidências de que um conjunto de teste melhor traduzido (diferença entre tradutores) permita uma melhor performance de classificação.

A abordagem do trabalho descrito acima se assemelha ao presente nesta pesquisa, contudo, abordaremos a linguagem português, com as métricas de avaliação: acurácia, precisão, *recall* e *F-measure*. Seguindo os resultados obtidos pelos autores para a linguagem espanhol, esperamos que o português também apresente boa performance nos classificadores, já que ambas são linguagens latinas com estruturas semelhantes.

Em seu trabalho posterior [27], os mesmos autores continuaram a implementação de um classificador multilinguístico, agora com 4 linguagens: italiano, espanhol, francês e alemão. Utilizando dados (*tweets*) coletados do Twitter e um classificador (SVM) de sentimento para *tweets* em inglês, sem nenhuma ferramenta específica para processamento de linguagem de texto. Dessa vez, o serviço de tradução Google foi utilizado para traduzir os dados de treino nos quatro idiomas e posteriormente, estes foram manualmente revizados. As *features* dos dados são extraídas assim como se fosse em inglês, *unigrams* e *bigrams*, e são utilizadas para treinar o classificador SVM SMO, da mesma maneira que seria em inglês. Os testes foram feitos com linguagens individuais, pares de linguagens, três linguagens latinas e cinco linguagens (incluindo inglês). Os resultados mostraram que a inclusão de todas as linguagens no treino do classificador melhorou a performance, aumentando a acurácia do mesmo, assim como pares de linguagens semelhantes em estrutura (italiano e espanhol). Os autores concluíram que a simples união da normalização com a tradução automática dos *tweets* pode prover dados de treino de qualidade para muitas linguagens.

Assim como o anteriormente mencionado, esse trabalho também se assemelha ao estudo apresentado neste documento. Entretanto, os autores focaram apenas na acurácia como métrica de avaliação, não explorando precisão e recall, conseqüentemente, a *F-measure*, que são métricas de avaliação de desempenho importantes quando se trata de classificação de sentimento. Os resultados obtidos para as linguagens espanhol e italiano podem indicar que português também apresentará bom desempenho nas mesmas circunstâncias.

Capítulo 4

Coleta e Preparação dos Dados

Neste capítulo serão apresentados o processo de coleta de dados, explicando a construção da aplicação para coleta de dados do Twitter e os experimentos de coleta realizados para a definição do conjunto de dados de texto para a realização das etapas seguintes do processo de classificação de sentimento. Posteriormente, apresentaremos a preparação dos dados para classificação de sentimento, explicando as etapas realizadas neste trabalho, falando um pouco sobre os desafios encontrados no processo.

4.1 Coleta de Dados

O processo de coleta de dados começa com a implementação de um *crawler* de dados, que irá se comunicar com a API de *streaming* do Twitter. *Crawler*, como o termo em inglês sugere, é uma aplicação que rastreia dados de uma forma metódica e automatizada. Um *Web Crawler*, por exemplo, realiza uma varredura na internet de maneira sistemática, através de informação vista como relevante à sua função, capturando os textos das páginas e cadastrando os *links* encontrados, possibilitando assim, encontrar novas páginas. São a base dos motores de busca *online*, sendo responsáveis pela indexação dos *sites*, armazenando-os na base de dados dos motores de busca, e mantendo estas atualizadas. *Crawlers* também são utilizados na obtenção de informação específica, como minerar endereços de *e-mail* em páginas da internet [41]. De maneira análoga, um *crawler* de dados do Twitter rastreia o fluxo de *tweets* em busca de informação. Esta informação pode ser filtrada ou não, mediante *queries* de busca. Para a comunicação do *crawler* com a API de *stream* do Twitter foi utilizada a biblioteca de código aberto, Twitter4J¹¹. Esta biblioteca oferece uma classe contendo vários métodos preparados para lidar com a API de *stream* (utilizada em nosso trabalho).

A *stream* pode ser *sample* ou filtrada. A *sample* de *stream* do Twitter captura todo o fluxo de *tweets* a partir do início de processamento da *stream* criada, não fazendo distinção de termos de busca, ou linguagem em que os *tweets* se encontram escritos. A *stream* filtrada captura apenas

¹¹ Disponível em: <http://twitter4j.org/en/>

os *tweets* que encaixam em uma *query* de busca. Em nosso trabalho, palavras-chave e a linguagem dos *tweets* foram definidas através de *queries* de busca. Um objeto do tipo *Query* configura o conjunto de palavras-chave e a linguagem dos *tweets* através de uma arquivo de entrada contendo estas informações. Se o arquivo de entrada estiver vazio, o objeto *query* não configurará informações de palavras-chave ou linguagem, logo a *stream* será *sample*.

A persistência dos dados coletados foi implementada através da integração com o banco de dados de código aberto, MongoDB¹². Neste banco de dados são armazenados documentos, que são estruturas de dados compostas por pares de campo e valor. Esses documentos são similares a objetos JSON. Os valores dos campos podem conter outros documentos, *arrays*, ou *arrays* de documentos. Cada banco de dados MongoDB criado armazena coleções de documentos, como é apresentado na Figura 4.1.

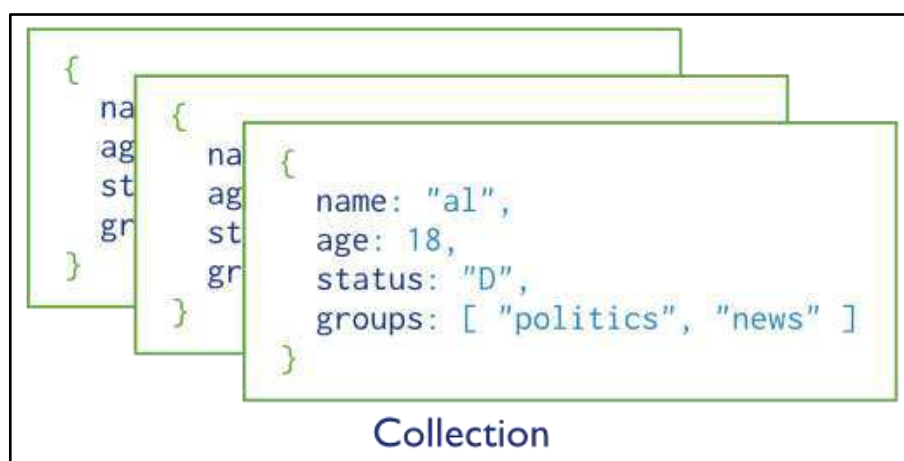


Figura 4.1 - Coleções de documentos armazenados em um banco de dados MongoDB

Os documentos são armazenados como documentos BSON¹³, que é uma representação binária de documentos JSON, que suporta a incorporação de documentos e *arrays* dentro de outros documentos e *arrays* [42].

4.1.1 Definição do contexto da pesquisa

¹² Disponível em: <https://www.mongodb.com/download-center>

¹³ Disponível em: bsonspec.org/

Em nosso trabalho foram configuradas duas *streams* filtradas, uma para a linguagem português e outra para a linguagem inglês, com o mesmo conjunto de palavras-chave. Para definir as palavras-chave foi necessário escolher um contexto de busca que retornasse mais resultados relevantes nas duas linguagens.

Em dezembro de 2013, uma pesquisa [43] apontou que a língua inglesa é a mais falada por usuários no Twitter com 34% dos usuários. A língua portuguesa assumiu o sexto lugar, com 6% dos usuários. Em junho de 2015, outro estudo [44] apontou a língua inglesa como a mais falada por usuários na internet, com 26% dos usuários. Português ficou em quinto lugar, com 4% dos usuários. Baseado nesses resultados, podemos deduzir que a coleta de dados na língua portuguesa supostamente retorna menos resultados que a na língua inglesa, em um contexto de pesquisa comum às duas linguagens e em um mesmo espaço de tempo.

Primeiramente, foram definidos objetos de pesquisa que fossem populares entre usuários falantes das duas linguagens. Portanto, títulos de filmes e jogos de *videogame* foram escolhidos, baseados em suas datas de lançamento e popularidade na mídia na época em que a coleta foi realizada. A primeira rodada de coletas foi realizada em português para testar o volume de dados coletado para esta linguagem com objetos de pesquisa populares. A Tabela 4.1 apresenta os resultados da primeira rodada de coletas realizada no período de abril a junho de 2015. Os resultados mostraram que esse tipo de objeto de pesquisa não retornaria volumes de dados significativos.

Título	Tipo	Lançamento	Número de Tweets
Mortal Kombat X	<i>Videogame</i>	14/04/2015	1710
Mad Max - Estrada da Fúria	Filme	14/05/2015	3974
Jurassic World	Filme	11/06/2015	2658
Vingadores - Era de Ultron	Filme	23/04/2015	6616

Tabela 4.1 - Resultado da primeira rodada de coletas

Em um segundo momento, foram definidos dois objetos de pesquisa na categoria séries de TV. A coleta foi realizada em português e em inglês. A Figura 4.2 apresenta a comparação dos volumes dos resultados obtidos no período de cerca de 5 meses de coleta.



Figura 4.2 - Comparação dos resultados das coletas em inglês e português

Título	Período de Exibição na TV	Número de tweets
<i>The Walking Dead</i>	outubro/2015 a abril/2016	(EN) 3.864.210
		(PT) 1.026.528
<i>Game of Thrones</i>	abril a junho de 2015	(EN) 70.552
		(PT) 22.127

Tabela 4.2 - Resultados das coletas nas duas linguagens e períodos de exibição dos títulos

É interessante observar que o período de exibição na TV dos títulos em questão é muito relevante nos resultados obtidos com as coletas realizadas. O título *Game of Thrones* encontrava-se fora de transmissão quando as coletas começaram, como podemos observar na

Tabela 4.2 e na Figura 4.2. Enquanto que o título *The Walking Dead* começou uma nova transmissão de temporada na TV durante o processo de coleta.

4.2 Preparação dos Dados

Uma vez que os *tweets* são coletados e armazenados em um banco de dados local, começa a etapa de preparação ou pré-processamento dos dados. O texto é essencialmente informal, muitos desafios devem ser levados em consideração, como erros de gramática, gírias, regionalismos e caracteres repetidos [30].

O conteúdo postado no Twitter é frequentemente criado a partir de aplicações móveis, em celulares e tablets, sendo assim, as sentenças são, em sua maioria, não tão bem formadas, apresentando erros sintáticos e gramaticais, e abreviações específicas ao contexto. Lidar com ruído é uma tarefa trabalhosa e difícil em PLN, mas considerando textos curtos esse problema é ainda mais significativa [17].

As postagens do Twitter, ou *tweets*, frequentemente são formadas por gírias específicas e abreviações que permitem a composição de sentenças mais curtas. Os usuários também escrevem muitas expressões dependentes da linguagem, podendo diferir demograficamente, o que torna ainda mais difícil a sua interpretação. Por exemplo, expressões próprias do inglês falado na Inglaterra podem diferir de expressões do inglês falado nos Estados Unidos.

Como já discutido no Capítulo 2, a literatura apresenta algumas propostas para lidar com os problemas citados anteriormente:

- Filtragem das postagens: remoção de URLs, nomes de usuários, palavras e termos reservados da linguagem do Twitter;
- Reconhecimento e/ou extração de entidades;
- Remoção de *stop words*;
- Marcação POS;
- *Stemming*: redução de um termo ao seu radical;
- Tratamento de *HashTags*: algumas *hashtags* contém letras maiúsculas e minúsculas indicando palavras distintas. As palavras são separadas de acordo com a presença de letras maiúsculas. Por exemplo, “hateMyDay” é decomposta em “hate My Day”.

A seguir, as etapas de preparação de dados de texto implementadas neste trabalho serão explicadas.

4.2.1 Definição dos conjuntos de dados

Como foi visto na seção 4.1 deste capítulo, as coletas de dados para o título *The Walking Dead* obtiveram um maior volume de *tweets*, e por esta razão foram escolhidas para serem utilizadas em nosso experimento. Primeiramente, como pode ser visualizado na Figura 4.3, extraímos um montante de 7.000 (sete mil) *tweets* de uma porção de 10.000 (dez mil) *tweets* da coleta de dados em inglês para trabalharmos em nosso primeiro experimento. Esse número foi escolhido por ser 70% do pedaço de coleta e conter uma quantidade suportável de linhas de texto para eventual tratamento manual dos dados.

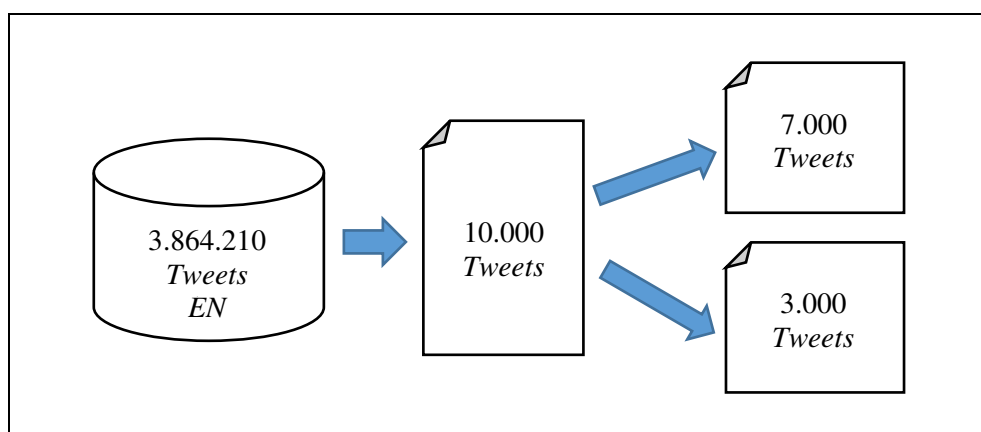


Figura 4.3 - Separação dos conjuntos de dados a serem utilizados nos experimentos

4.2.2 Etapas de tratamento dos dados

A primeira fase da preparação dos dados extraídos do Twitter é a correção da linguagem em que o *tweet* se encontra escrito, até onde for possível, o tratamento de termos próprios da rede social e o tratamento de *features* (transformação da escrita de certas *features*). O conjunto de dados passará por um processo de tradução automática, portanto, a transformação de certas *features* se torna necessária para que traduções não desejadas ocorram, como por exemplo, “The Walking Dead” ser traduzido para a língua portuguesa.

A linguagem utilizada por usuários nas mídias sociais é diferente da encontrada na mídia geral e a forma como as palavras são escritas nem sempre é a mesma encontrada no dicionário. Há uma gíria comum utilizada por usuários de plataformas de mídias sociais *online* que consiste em expressões abreviadas (“lol”, “omg”, “brb”, etc), emoticons e frequentemente, palavras com letras repetidas para enfatizar seu significado (“I just watched Fear the Walking Dead and I think it’s aweeeesooome!”) ou mesmo passar uma intensidade maior de sentimento (“this is soooooo nice”). Além disso, a plataforma Twitter apresenta uma linguagem própria para demarcar elementos no *tweet* de um usuário, que deve ser considerada na preparação dos dados para posterior classificação dos mesmos:

- Termos reservados como “via” e “RT” (*retweet*);
- Menções a usuários, denotadas por @ na forma @usuário;
- *Hashtags*, denotadas pelo símbolo #.

Tweet 1: Lol I'm diving into #TWD im over this shit. Bye.

Tweet 2: Excited for ahs more than the walking dead tbh

Tweet 3: lemme watch fear the walking dead abeg; this team is shite

Figura 4.4 - Exemplos de *tweets* coletados em inglês

A Figura 4.4 apresenta exemplos de três postagens de usuários da plataforma Twitter, extraídas dos dados coletados. Podem ser observados formas da linguagem inglesa erroneamente escritas, gírias, abreviações e expressões. Dessa maneira, a primeira etapa de tratamento do conjunto de dados escolhido em inglês, antes de traduzi-lo para português, é a correção de possíveis palavras mal escritas, remoção de termos reservados à própria linguagem do Twitter, URLs, tratamento de algumas *features* e *hashtags*. Como pode ser observado na Figura 4.5, após o processo de remoção de palavras e termos reservados da linguagem do Twitter (“RT”, “via” e menções a usuários) e URLs, o conjunto passou a conter 6.444 *tweets*.

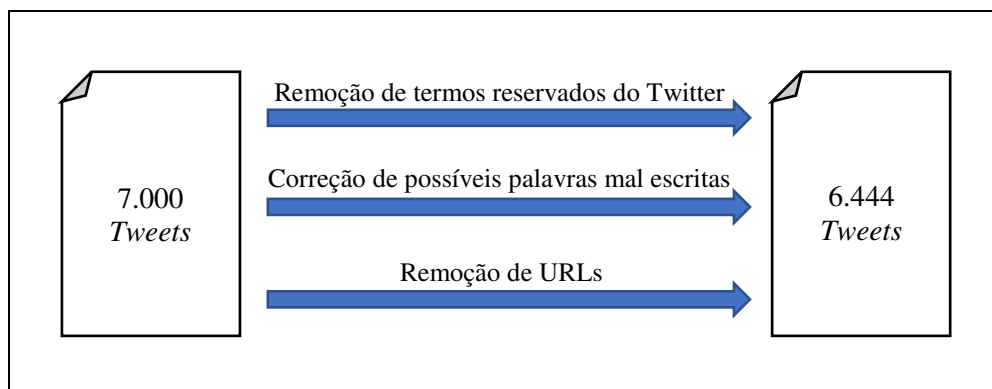


Figura 4.5 - Processos e resultado da primeira etapa de tratamento dos dados

No processo de correção das palavras mal escritas, seja com presença de letras repetidas ou erros de gramática, parte deste foi realizado com ajuda do dicionário Thesaurus¹⁴, de maneira que se um determinado termo presente em uma sentença (*tweet*) não apresentasse opção de correção, este seria corrigido manualmente, caso contrário, seria corrigido automaticamente junto às suas repetições no arquivo de dados. Adicionalmente, foi utilizado o Microsoft Bing Dictionary¹⁵, presente no aplicativo de edição de texto Microsoft Word¹⁶ em sua versão 2013, para correção de palavras. O termo não encontrado pelos dicionários seria verificado e corrigido manualmente, para uma verificação automática posterior de possíveis repetições do mesmo no conjunto de dados e consequente substituição pelo termo corrigido.

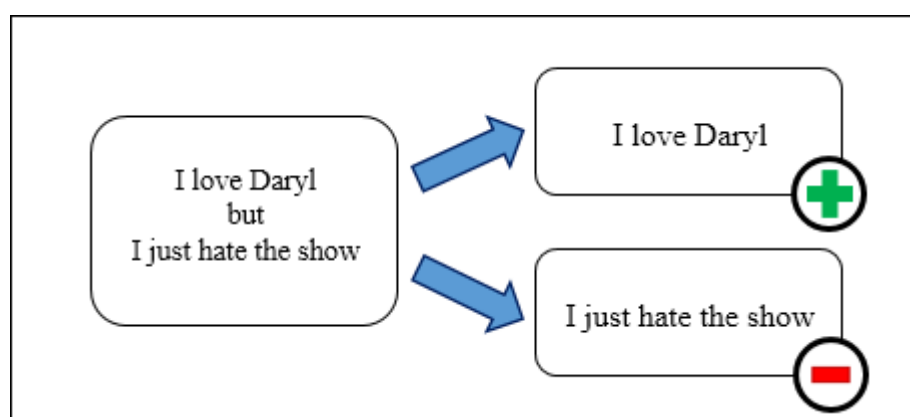


Figura 4.6 - Exemplo de separação de frases integrantes de uma sentença que apresentam polaridades distintas

¹⁴ Disponível em: <http://www.thesaurus.com/>

¹⁵ Disponível em: <https://www.microsoft.com/en-us/store/apps/microsoft-bing-dictionary-win10/>

¹⁶ Disponível: <https://products.office.com/en/word>

Algumas sentenças presentes no conjunto de dados apresentavam partes com polaridades finais distintas. O processo de classificação de sentimento utilizado neste trabalho é mais simples e utiliza apenas as polaridades positiva e negativa, portanto, tais sentenças precisam ter suas partes separadas. Um exemplo de uma sentença composta por frases com polaridades distintas é apresentada na Figura 4.6, onde uma frase presente na sentença possui claramente polaridade positiva e a outra frase presente, polaridade negativa.

4.2.2.1 Extração e substituição de features

Ainda antes do processo de tradução dos dados, é necessária a identificação de certas *features* para que não ocorram traduções indesejadas. Primeiramente a identificação de abreviações e a possível substituição destas pelas respectivas frases que as representam. Essas abreviações podem ser expressões ou mesmo nomes de séries de TV (como observado no conjunto de dados trabalhado).

Formas encontradas	Substituição
“ikr”, “I know right”	“ikr”
“idk”, “I don’t know”	“i don’t know”
“tbh”, “to be honest”	“tbh”
“bc”, “cos”, “coz”, “because”	“because”

Tabela 4.3 - Exemplos de *features* modificadas e/ou substituídas antes do processo de tradução

Títulos como o da série da TV americana “American Horror Story”, aparecem tanto em sua forma de escrita normal quanto na forma abreviada ou em sigla, neste caso, na forma de sigla, “ahs”. Mesmo concatenando o título para “americanhorrorstory”, sistemas de tradução automática como o Google Tradutor, adicionam os espaços retirados na concatenação e algumas vezes traduzem o título. Para não ocorrer este tipo de problema, a sigla do título foi utilizada, ou seja, todas as aparições de um dado título foram substituídas pela sua forma em sigla. Em alguns casos, como o do título “Arrow”, a substituição foi realizada através da concatenação <título> + <”serie”>. Neste caso, o termo final se encontrou na forma

“arrowserie”. O número de títulos de séries de TV, episódios, títulos de músicas, nomes de personalidades, nomes de escolas, etc., tratados em nosso conjunto de dados passa de 100 (cem) unidades.

Algumas *features* encontradas em forma de n-gramas representam termos polares em uma classificação de sentimento. Na Tabela 4.3 são apresentados quatro exemplos de n-gramas encontrados no conjunto de dados explorado. N-gramas como “to be honest” e “i know right” não representam significante polaridade em sentenças, sendo utilizados em sua maioria como complemento de sentenças, tanto negativas quanto positivas. Por exemplo, temos as sentenças “I hate that movie, to be honest” e “I think they’ve made a great job, to be honest”, que possuem polaridades negativa e positiva respectivamente, mostrando que o uso de “to be honest” não influencia na polaridade das mesmas. No caso de “I know right”, a forma sigla foi mantida como padrão pois no processo de tradução a palavra “right” poderia gerar uma tradução de polaridade positiva, quando na verdade, semelhante ao caso anterior, o n-grama não apresenta influência na polaridade de uma sentença. Em contrapartida, algumas siglas e abreviações precisam ser substituídas por suas formas em n-gramas, como nas linhas 2 e 4 da Tabela 4.3, para que o sentido da sentença não se perca no processo de tradução. No caso da linha 2, “I don’t know” representa um n-grama que pode influenciar na polaridade de uma sentença, portanto a substituição da sigla pela forma original em n-grama.

Sentença	Polaridade
“There’s no way you can survive that.”	Negativa
“nw, the twd staff is in my town!”	Positiva

Tabela 4.4 - Utilização da expressão “no way” com diferentes polaridades

Algumas *features* aparecem na forma de sigla e n-grama no conjunto de dados, em uma clara intenção de diferenciar o significado das mesmas em uma sentença, como é o caso de “no way” e “nw”, que não foram modificadas nem substituídas, pois observamos que os usuários das redes sociais diferenciam o significado da expressão pela forma como esta é escrita neste caso. A Tabela 4.4 apresenta duas sentenças onde a expressão “no way” representa diferentes polaridades. Apesar da presença da partícula de conhecida polaridade negativa “no”, a

expressão também representa uma interjeição positiva, como no caso da sentença da linha 2 da tabela citada. Neste caso, o tratamento já é realizado pelos próprios usuários das redes sociais, como observamos no conjunto de dados.

Após o processo de tradução automática dos dados através do Google Tradutor, passamos a ter a versão traduzida do conjunto em inglês. O conjunto de dados traduzido foi verificado manualmente, linha a linha, para correção de erros de tradução do processo automático. O bigrama “can’t wait” não poderia ser modificado no conjunto em inglês antes da tradução deste, pois poderia custar em traduções sem sentido em muitos casos. Tal bigrama pode se apresentar com polaridade positiva ou negativa, dependendo da sentença na qual se encontra, e desta forma, influenciar a polaridade final da dada sentença.

Sentença	Polaridade
“I can’t wait so long to see that.”	Negativa
“I just can’t wait to watch the next season”	Positiva

Tabela 4.5 - Utilização da expressão “can’t wait” com diferentes polaridades

Na primeira sentença apresentada na Tabela 4.5 podemos observar que o bigrama “can’t wait” influencia na polaridade final da sentença pela presença da negação “can’t”. Porém, na segunda sentença podemos observar que se trata de uma expressão de polaridade positiva, apesar da presença da negação. Dessa maneira, há a necessidade da modificação desse bigrama tanto no conjunto em inglês quanto em português, a fim de preservar a esperada polaridade final da sentença.

Sentença Original	“I can’t wait to watch the new episode”
Tradução EN-PT Google Tradutor	“Eu não posso esperar para ver o novo episódio”
Tradução EN-PT Microsoft Bing	“Eu não posso esperar para assistir ao novo episódio”
Tradução revisada manualmente	“Eu mal posso esperar para assistir ao novo episódio”

Tabela 4.6 - Comparação de traduções de sentenças contendo a expressão “can’t wait”

Os dois sistemas de tradução automática que utilizamos neste trabalho, Google Tradutor e Microsoft Bing, traduzem uma dada sentença que apresenta o bigrama “can’t wait” de polaridade positiva, com uma polaridade final negativa, como pode observado na Tabela 4.6. Isso se deve ao fato de que os processos de tradução automática não são tão capazes ainda de identificar o contexto das sentenças. Há a possibilidade de escolha da tradução mais plausível para uma determinada sentença pelo usuário, porém no processo de tradução automática de várias linhas de texto, essa verificação manual pelo usuário se torna muito custosa em tempo. Em nossos conjuntos de dados, os bigramas “can’t wait” e “não posso” foram concatenados nas sentenças onde a polaridade final esperada é positiva, de forma a evitar que os classificadores de sentimento identificassem as *features* “can’t” e “não” nessas sentenças.

Após todas as modificações no texto citadas e a correção do conjunto em português, os dois conjuntos foram passados para letras minúsculas, para não haver distinção de palavras por caixa alta no texto. O conjunto em português passou pela extração dos acentos das vogais.

4.2.2.2 Tratamento de negação

Há *features* que influenciam diretamente na classificação de sentimento de uma sentença, pois essas apresentam polaridades conhecida. Adjetivos como “bom” e “ruim” (“good” e “bad”, em inglês) e verbos como “odiar”, “gostar”, “amar” (“to hate”, “to like”, “to love”, em inglês) possuem polaridade conhecida e influenciam diretamente na polaridade final de uma sentença. Entretanto, algumas expressões podem mudar a polaridade destas palavras e se não forem tratadas, existe uma grande possibilidade de as sentenças serem classificadas erroneamente quanto à polaridade de sentimento.

Assim como acontece em operações com sinais da álgebra, onde $-(+1) = -1$ e $-(-1) = +1$, de maneira análoga acontece com a negação de palavras polares. Como pode ser observado na Tabela 4.7, palavras de polaridade positiva ao serem negadas passam a ter polaridade negativa. Em contrapartida, palavras que apresentam polaridade negativa passam a apresentar polaridade positiva ao serem negadas. Portanto, se torna necessário tratamento dessas *features* de forma a não comprometer a classificação final de polaridade das sentenças que as incluem. O processo consiste em transformar os termos negados com a adição de “not_”, de forma que “not bad”

passa a ser “not_bad”, por exemplo. O mesmo processo acontece com todas as outras palavras polares que sofrem negação.

<i>Feature</i>		Polaridade
Inglês	Português	
good	bom	Positivo
not good	não bom	Negativo
bad	ruim	Negativo
not bad	não ruim	Positivo
to like	gostar	Positivo
to not like	não gostar	Negativo

Tabela 4.7 - Exemplos de palavras de polaridade conhecida e a mudança de polaridade com a adição da negação

4.2.2.3 *Stemming, Remoção de Stop Words e Tokenização*

Os processos de *stemming*, remoção de *stop words* e tokenização do texto, explicados anteriormente no Capítulo 2 deste documento, são realizados através do *software* utilizado nos processos de classificação dos conjuntos de dados, Weka 3.8¹⁷, o qual disponibiliza diversas técnicas de aprendizado de máquina para classificação de dados de texto.

O *stemming* foi realizado através dos *stemmers* para as linguagens inglês e português implementados através da Snowball¹⁸, que é uma linguagem de processamento de texto utilizada na criação de algoritmos de *stemming* de texto para IR. SnowBall tem esse nome em homenagem à linguagem de manipulação de texto SNOBOL, criada em 1960 [45].

A transformação das sentenças em *tokens*, ou tokenização, é realizada através da função para transformação de *strings* em vetores presente no *software* Weka 3.8. Essa função separa os *tokens* seguindo algumas regras definidas pelo usuário. Em nossos experimentos, a tokenização

¹⁷ Disponível em: <http://www.cs.waikato.ac.nz/ml/index.html>

¹⁸ Disponível em: <http://snowballstem.org/algorithms/>

foi realizada através da presença de espaços entre palavras e sinais de pontuação (i.e., ponto final, vírgula, exclamação e etc).

Inglês	Português
I, me	eu, me, mim
you, u, yo, ya	voce, vc, c, tu, te, ti, lhe
he, him, she, her, it	ele, ela
we, us	nos, nois, a gente, vos, vois
they, them	eles, elas
the, a, an	o, a, os, as
this, that, dat, these, those	isso, esse, essa, isto, este, esta aquilo, aquele, aquela, esses, essas aqueles, aquelas
on, in, at, into, onto	em, na, no, nas, nos, nele, nela, neles, nelas, nisso, nesse, nessa, nesses, nessas nestes, nestas, nisto, neste, nesta naquilo, naquele, naquela
from, of, to, for, by, with	de, do, da, dos, das, para, pro, pra pros, pras, ao, aos, por, com

Tabela 4.8 - Parte das listas de *stop words* em inglês e português

A lista de *stop words*, em inglês e português, utilizada em nossos experimentos é constituída basicamente de pronomes, artigos, conjunções e preposições, como podemos observar na Tabela 4.8 que apresenta parte da lista utilizada neste trabalho. É interessante observar que devido à variação de gênero na língua portuguesa, a lista nesta linguagem é mais extensa. A lista completa de *stop words* pode ser visualizada no Apêndice B deste documento.

Capítulo 5

Classificação de Sentimento

Neste capítulo será apresentado o processo de classificação de sentimento abordado neste trabalho. Primeiramente será apresentado o método utilizado para a classificação dos conjuntos de dados quanto à subjetividade de suas sentenças. Por fim, será explanado o processo de classificação de sentimento desenvolvido para as duas linguagens utilizadas.

O processo de análise de sentimento de sentenças curtas pode ser tratado como um problema de PLN, ou mais especificamente, um problema de classificação de texto. A classificação de texto é uma tarefa definida pela designação de uma classe pré-definida a um dado texto, este pode ser um documento de texto completo, um parágrafo, uma sentença apenas, etc. Mais formalmente, classificar um dado texto significa encontrar a função de classificação F , Equação 5.1, que descreve como textos são associados às classes e também designa um texto $t_i \in T$ à uma categoria $c_j \in C$, onde $T = \{t_1, \dots, t_n\}$ é um domínio de textos e $C = \{c_1, \dots, c_n\}$ é um conjunto pré-definido de classes.

$$F: T \rightarrow C, f(t_i) = c_j \quad (5.1)$$

Em abordagens que utilizam técnicas de aprendizado de máquina, a classificação de texto começa com um conjunto de dados de texto para treino, ou conjunto de treino, $T = \{t_1, \dots, t_n\}$ que já estão marcados com uma classe $c_j \in C$ (positivo ou negativo, por exemplo). É construído um modelo de classificação, seguindo a função F , que é capaz de designar a classe correta a um novo texto $t_i \in T$.

Para medir a performance do modelo de classificação construído, uma parte do conjunto de textos pré-marcados com as classes de classificação não é utilizada como conjunto de treino, mas sim como conjunto de teste do modelo de classificação construído. As marcações de classe já presentes nos textos do conjunto de teste são comparadas com as estimadas pelo classificador, medindo a performance do modelo de classificação construído a partir do conjunto de treino. A

indicação mais comum é que cerca de 70% dos dados marcados sejam para treino do classificador e construção do modelo de classificação, e 30% dos dados seja para teste do modelo de classificação construído.

Antes de classificar uma sentença quanto à sua polaridade de sentimento, essa precisa apresentar opinião, ou seja, ser subjetiva. Para tanto, existem técnicas que identificam partes opnativas em sentenças, como a técnica de marcação POS, que identifica e marca adjetivos, mas só há algoritmos de marcação POS que apresentam resultados satisfatórios para a língua inglesa. Para a língua portuguesa não encontramos algoritmos compatíveis em eficácia. Portanto, como mencionado no Capítulo 3 deste documento, utilizaremos a técnica sugerida por Yu e Hatzivassiloglou [34] para classificar o texto quanto à sua subjetividade, utilizando algoritmos de classificação e sentenças previamente marcadas em objetiva ou subjetiva.

5.1 Marcação e classificação de subjetividade

O conjunto de dados em inglês foi marcado quanto à sua subjetividade, obedecendo a regra de que a presença de qualquer termo opnativo em uma dada sentença denota sua subjetividade. Analogamente, a falta de qualquer termo opnativo em uma dada sentença denota sua objetividade.

Construções verbais	Palavras
I prefer, I love , I like, I enjoy, I recommend	best, better, great, good, amazing
I believe, I think, I feel, I know	worse, worst, awful, bad, sad

Tabela 5.1 - Exemplos de palavras e construções verbais que expressam opinião ou emoção em uma sentença textual

Termos opnativos podem ser adjetivos ou verbos que expressam opinião, ou emoção, em relação a um certo tema. A Tabela 5.1 apresenta alguns exemplos de palavras e construções verbais que expressam opinião ou emoção e, conseqüentemente, podem indicar a subjetividade de uma sentença. A lista completa se encontra no Apêndice C deste documento.

Através de um processo automatizado, as sentenças do conjunto de dados em inglês, antes do mesmo ser traduzido para gerar o conjunto em português e após a correção da escrita das palavras, foram marcadas em “s” se subjetivas e “o” se objetivas, de acordo com a presença ou não das palavras opinativas. O conjunto de dados em inglês continha 6444 sentenças não marcadas, após a execução do algoritmo de marcação das sentenças de acordo com a presença ou não de palavras opinativas, o conjunto foi marcado em 5431 sentenças subjetivas e 1013 sentenças objetivas.

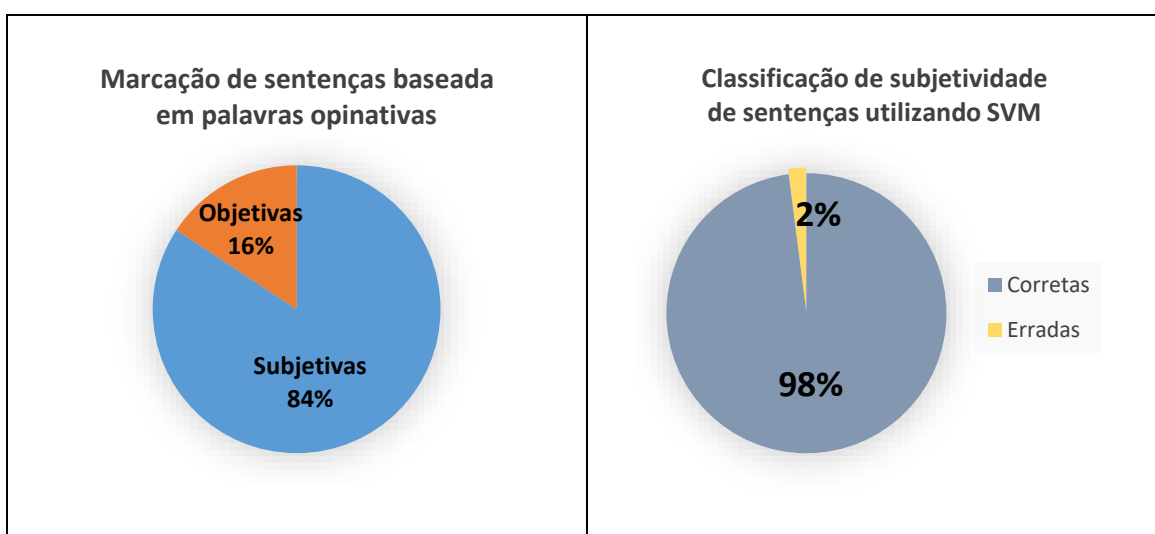


Figura 5.1 - Resultados das execuções dos algoritmos de marcação e classificação de subjetividade

A Figura 5.1 apresenta os resultados das execuções dos algoritmos de marcação e classificação de subjetividade das sentenças do conjunto de dados em inglês. O conjunto resultante da execução do primeiro algoritmo, constituído de sentenças marcadas em subjetivas e objetivas, vai entrar como conjunto de treino para a geração do modelo de classificação do algoritmo SVM. Depois que o modelo é gerado, o mesmo conjunto entra como conjunto de teste no classificador e as sentenças que forem classificadas diferente de sua marcação original serão eliminadas do conjunto. O processo se repete até que nenhuma sentença seja classificada diferente de sua marcação. O segundo gráfico da Figura 5.1 apresenta o resultado da execução do classificador SVM no conjunto de dados em inglês. Foram necessárias duas execuções do classificador, na segunda execução nenhuma sentença mais foi classificada erroneamente. Ao todo, 110 sentenças foram classificadas diferente de suas marcações e eliminadas do conjunto

de dados, restando 6.334 sentenças no conjunto de dados, marcadas em subjetivas e objetivas. Após a eliminação das sentenças marcadas como objetivas, os conjuntos de dados em inglês e em português passaram a conter, cada um, 5.321 sentenças de texto classificadas como subjetivas, ou seja, opinativas, que serão utilizadas na classificação de polaridade de sentimento.

5.2 Classificação prévia de polaridade de sentimento do conjunto de dados em inglês e marcação das sentenças

Após a definição e pré-processamento dos conjuntos de dados, é necessário que seja realizada a classificação e marcação prévia de polaridade de sentimento desses conjuntos, pois estes serão utilizados como conjunto de treino e teste em nosso experimento, como será melhor explicado mais à frente. Para essa classificação foi utilizado um conjunto de contexto geral com mais de um milhão de *tweets* em inglês [46], marcados com “0”, se negativos, e “1”, se positivos, para treino do classificador. Ao todo o conjunto de treino consiste de 552.112 sentenças marcadas como positivas e 492.402 sentenças marcadas como negativas, e é a junção de duas fontes de dados:

- *Sanders Analytics*¹⁹ - conjunto de dados para treino e teste de algoritmos de análise de sentimento. Consiste em *tweets* em inglês classificados manualmente. Os *tweets* pertencem a quatro tópicos [47].
- IMICH S1650²⁰ - competição de classificação de sentimento realizada no período de março a abril de 2011 pela Universidade de Michigan nos Estados Unidos. Os participantes recebiam um conjunto de treino com *tweets* em inglês marcados como positivos ou negativos e faziam suas submissões de dados classificados.

Para a classificação de sentimento do nosso conjunto em inglês, implementamos o algoritmo Naive Bayes simples. O *software* Weka não suportou bem o tamanho do arquivo do conjunto de treino utilizado, demorando mais de cinco horas para apenas abrir o arquivo. Após as etapas de tratamento de texto do conjunto de treino - remoção de *stop words*, remoção de palavras

¹⁹ Disponível em: <http://www.sananalytics.com/lab/twitter-sentiment/>

²⁰ Disponível em: <https://inclass.kaggle.com/c/si650winter11>

reservadas, remoção de URLs, tratamento de sinais e pontuação, e tokenização - serem realizadas, fizemos alguns testes com o conjunto de treino no classificador NB implementado e no *software* Weka.

Em nossa implementação NB, executamos o conjunto completo como treino e como teste, obtendo uma acurácia de 65%. Para executar testes em outros algoritmos de classificação com o conjunto no *software* Weka, dividimos o conjunto em 21 partes de 50 mil sentenças cada. Executamos todas as partes em *10-fold cross-validation* nos algoritmos Naive Bayes e Naive Bayes Multinomial. Observamos que o algoritmo Naive Bayes Multinomial obteve melhor média de acurácia, 73,75% contra 67,64% do Naive Bayes.

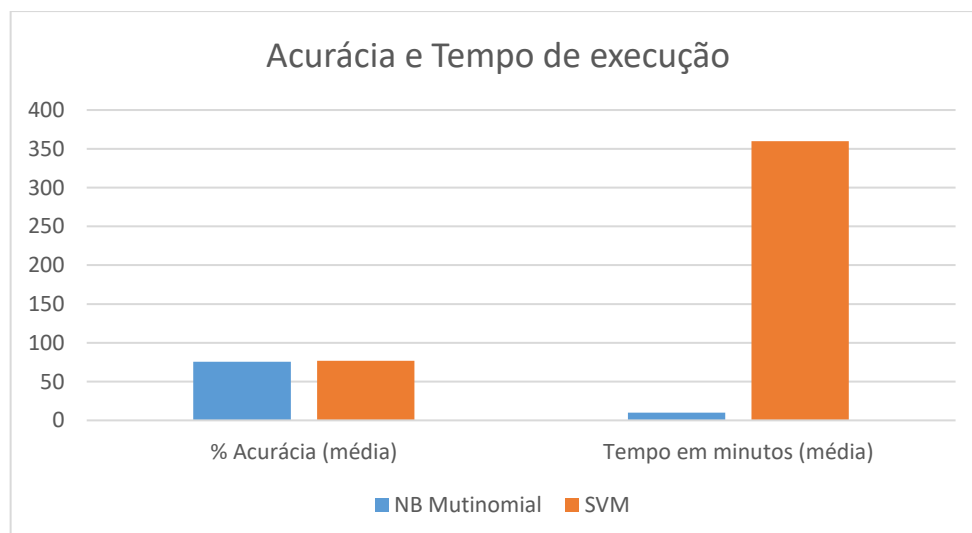


Figura 5.2 - Médias das acurácias X médias dos tempos de execução dos algoritmos NB Multinomial e SVM em conjuntos de 50 mil sentenças

As partes do conjunto que obtiveram melhor acurácia neste teste foram utilizadas para teste de tempo de execução e acurácia, dessa vez entre os algoritmos Naive Bayes Multinomial e SVM. A Figura 5.2 mostra os resultados das médias das acurácias X tempo de execução obtidos das execuções dos dois algoritmos. SVM obteve melhores acurácias, porém o tempo de execução aumenta bastante. Em contrapartida, Naive Bayes Multinomial consegue obter acurácias e tempos de execução satisfatórios. SVM obteve uma média de 77% de acurácia e Naive Bayes Multinomial obteve 75,6%. Já com relação à média de tempos de execução, o algoritmo Naive Bayes Multinomial executou em uma média de 10 minutos e o algoritmo SVM em 6 horas em

média. Para nossos experimentos subsequentes, os dois algoritmos serão utilizados pois o nosso conjunto de dados tem um tamanho bem reduzido de sentenças.

Nosso conjunto de dados foi previamente classificado em “0” para sentenças negativas e “1” para sentenças positivas. Como nossa implementação NB, por ser NB simples, obteve uma acurácia mais baixa que os outros algoritmos testados, a classificação foi revisada após a tradução dos dados e criação do conjunto de dados em português. Era necessária a revisão manual do conjunto de dados em português para que este estivesse o mais próximo possível da língua escrita por um ser humano, e a revisão da polaridade das sentenças foi realizada durante a revisão da escrita em português. O conjunto irmão em inglês teve suas polaridades de sentimento modificadas, caso necessário, após toda a revisão de escrita ter sido concluída.

Ao final dos processos de revisão e tratamento dos dados, nossos dois conjuntos, inglês e português, apresentaram 3225 sentenças marcadas como positivas e 2096 sentenças marcadas como negativas.

5.3 Métricas de avaliação da classificação de polaridade de sentimento

A efetividade de um processo de classificação de polaridade de sentimento não deve ser medida apenas através do valor da métrica acurácia, pois esta mede apenas a fração de classificações corretas. É relevante verificar as frações de falsas classificações corretas ou ainda, falsas classificações erradas. As métricas de precisão e *recall* são as mais frequentes e básicas na avaliação de efetividade em IR [48]. Para entendermos o cálculo das métricas de avaliação, temos os termos abaixo:

- *tp* - classificações verdadeiramente corretas para uma dada classe;
- *fp* - falsas classificações corretas para uma dada classe;
- *tn* - classificações verdadeiramente erradas para uma dada classe;
- *fn* - falsas classificações erradas para uma dada classe.

Se temos duas classes de classificação, *a* e *b*, para a classe *a*, o valor de *tp* será o número de elementos do conjunto de dados marcados como *a* e classificados como *a*, já o valor de *fp*

será o número de elementos do conjunto de dados marcados como b e classificados como a . Ainda para a classe a , o valor de fn será o número de elementos do conjunto de dados marcados como a e classificados como b . Já o valor de tn será o número de elementos marcados e classificados como b . Os mesmos valores para a classe b são calculados de maneira análoga aos da classe a .

$$Precisão = \frac{tp}{(tp + fp)} \quad (5.1)$$

$$Recall = \frac{tp}{(tp + fn)} \quad (5.2)$$

Nas Equações 5.1 e 5.2 podemos observar que a métrica precisão se concentra na fração de elementos de um conjunto que foram classificados em uma determinada classe. Já a métrica *recall* se concentra na fração de elementos verdadeiramente pertencentes à uma determinada classe.

$$F - measure = \frac{2 * (precisão * recall)}{(precisão + recall)} \quad (5.3)$$

A métrica *F-measure*, Equação 5.3, apresenta a média ponderada entre precisão e *recall*, realizando um balanceamento entre os valores destas métricas. Precisão, *recall* e *F-measure* variam entre 0 e 1.

$$acurácia = \frac{tp + tn}{tp + fp + fn + tn} \quad (5.4)$$

A métrica acurácia concentra-se na fração de elementos classificados corretamente para todas as classes, como pode ser observado na Equação 5.4.

Capítulo 6

Avaliação da Qualidade de Classificação de Sentimentos em dados com e sem tradução

Nosso trabalho tem o objetivo de analisar a validade da utilização de sistemas de tradução automática de texto em processos de classificação de sentimento, calculando se a classificação do texto traduzido corresponde à classificação dada ao texto original. O problema em questão recebeu nossa atenção ao depararmos com a necessidade de traduzir dados de texto em uma pesquisa anterior para obter resultados de opinião global de usuários. Em nossas pesquisas encontramos apenas alguns poucos trabalhos explorando o impacto dessas traduções automáticas, mencionados no Capítulo 3, porém estes trabalhos não investigaram o caso de traduções automáticas de texto para o idioma inglês, tampouco o impacto da tradução do idioma português nesses casos.

As pesquisas de Balahur e Turchi [36] apresentaram resultados satisfatórios em traduções automáticas da língua inglesa para as línguas de origem latina, espanhol e italiano. Estas línguas apresentam estrutura similar à da língua portuguesa; logo, esperávamos resultados semelhantes aos apresentados por esses autores.

6.1 Classificação de sentimento nos conjuntos de dados sem tradução

Nosso primeiro experimento consiste em realizar classificações de polaridade de sentimento utilizando apenas nossos conjuntos de dados irmãos, inglês e português. Nessa etapa, utilizamos apenas o software Weka e os algoritmos Naive Bayes Multinomial e SVM para a realização dos experimentos.

Como citado anteriormente, temos dois conjuntos de dados, um consistindo em sentenças em inglês, outro em português. O conjunto em português consiste das mesmas sentenças, na mesma ordem, do conjunto em inglês, traduzidas e revisadas manualmente para que fossem o mais próximo possível da língua portuguesa realmente falada pelos usuários das redes sociais. O conjunto em inglês foi classificado e teve suas sentenças marcadas com suas respectivas polaridades de sentimento, “1” caso a sentença apresentasse polaridade de sentimento positiva, e “0” caso apresentasse polaridade de sentimento negativa. Após revisões manuais dessas marcações, o conjunto em português teve suas sentenças marcadas com as mesmas polaridades presentes nas sentenças do conjunto em inglês. Por isso, chamamos estes conjuntos de irmãos, são iguais, apenas diferem na linguagem na qual suas sentenças se encontram escritas.

Primeiramente, antes de realizar o primeiro experimento com dados traduzidos em sistemas automáticos de tradução, validamos os dois conjuntos através das métricas de avaliação - acurácia, precisão, *recall* e *F-measure* - em *10-fold cross-validation* nos algoritmos de classificação Naive Bayes Multinomial e SVM.

O processo de *k-fold cross-validation* consiste na separação do conjunto de dados de treinamento em subconjuntos, onde um desses subconjuntos será utilizado como treinamento e os demais como teste. A cada iteração *k* um novo subconjunto será utilizado como treinamento, como pode ser observado na Figura 6.1 que apresenta o processo com *10-fold*, mantendo assim uma rotatividade de subconjuntos de treinamento e teste.

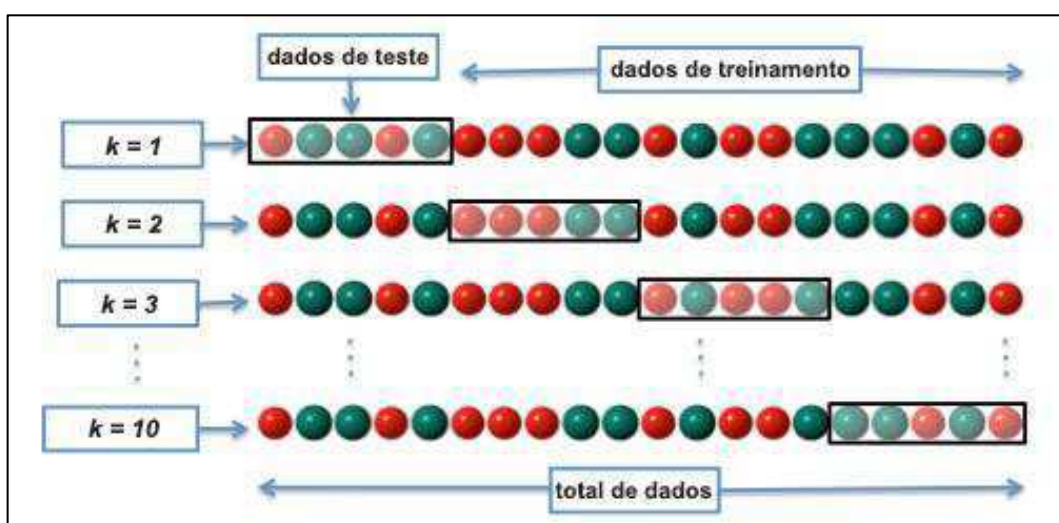


Figura 6.1 - Funcionamento do processo *k-fold cross-validation* com $k = 10$

Os valores das métricas de IR coletados através das classificações realizadas pelos algoritmos citados para os dois conjuntos de dados mostraram que estes conjuntos apresentam acurácias satisfatórias, assim como as médias das demais métricas de avaliação, como pode ser observado na Tabela 6.1. Apesar de serem irmãos, a diferença na língua escrita pode explicar a variação de acurácia entre os dois conjuntos em cada algoritmo de classificação utilizado. Contudo, essa variação de valores nas métricas nos dois conjuntos, para um mesmo algoritmo de classificação, é pequena.

		Conjuntos			
		Inglês		Português	
Métricas	Classificadores	NBM	SVM	NBM	SVM
		Acurácia	80,62%	82,91%	79,09%
	Precisão	0,811	0,828	0,796	0,819
	Recall	0,806	0,829	0,791	0,820
	F-measure	0,807	0,827	0,792	0,819

Tabela 6.1 - Resultados das classificações em *10-fold cross-validation* através dos classificadores NB Multinomial e SVM nos conjuntos de dados irmãos

Os dois conjuntos de dados foram divididos em duas partes, uma para treino dos classificadores e outra para teste. A escolha das sentenças presentes nas duas partes, treino e teste, foi feita aleatoriamente e igual nos dois conjuntos, obedecendo a regra de 70% das sentenças marcadas como positivas e 70% das sentenças marcadas como negativas nos conjuntos de dados seriam incluídas no subconjunto de treino, e o resto das sentenças seriam incluídas no subconjunto de teste. A divisão foi feita dessa forma para evitar que os subconjuntos ficassem desbalanceados em número de sentenças marcadas como positivas e negativas. Ao final da divisão, os subconjuntos em inglês e português, de treino e teste, continham 3725 e 1595 sentenças, respectivamente. A Tabela 6.2 apresenta os resultados das classificações realizadas com os subconjuntos de treino e teste criados para as duas línguas. É notável a queda nos valores

das métricas em relação à classificação em *cross validation* apresentada anteriormente, entretanto, os resultados obtidos ainda são satisfatórios, principalmente os obtidos através do classificador SVM.

		Conjuntos			
		Inglês		Português	
		NBM	SVM	NBM	SVM
Classificadores	Métricas				
	Acurácia	77,47%	80,5%	74,46%	79,42%
	Precisão	0,780	0,803	0,752	0,792
	Recall	0,775	0,805	0,745	0,794
	F-measure	0,776	0,803	0,747	0,792

Tabela 6.2 - Resultados das classificações através dos os algoritmos NB Multinomial e SVM com os subconjuntos de treino e subconjuntos de teste construídos

6.1.1 Análise do Primeiro Experimento

Em nosso primeiro experimento, preparamos um conjunto de dados de texto na língua inglesa que foi traduzido para a língua portuguesa, gerando dois conjuntos, um com sentenças escritas em inglês, outro com sentenças escritas em português. Ambos os conjuntos foram revisados em sua escrita e pré-processados para o processo de classificação de polaridade de sentimento.

A validação dos conjuntos através de *10-fold cross-validation* e os algoritmos de classificação Naive Bayes Multinomial e SVM, mostrou que o conjunto de dados em inglês oferecia uma acurácia de 80% a 83%, já o conjunto de dados em português oferecia uma acurácia de 79% a 82%. A pequena diferença nas classificações nos levam a crer que sejam resultado da distância entre às duas línguas utilizadas.

6.2 Classificação de sentimento de dados de teste submetidos à tradução automática

Os subconjuntos de teste nos dois idiomas, inglês e português, foram submetidos aos processos de tradução automática, Google Tradutor e Microsoft Bing Tradutor, gerando quatro subconjuntos de teste: dois traduzidos inglês-português e dois traduzidos português e inglês. A Figura 6.2 demonstra o processo de divisão dos conjuntos de dados em subconjuntos de treino e teste, e a tradução destes subconjuntos de teste em dois subconjuntos, um traduzido pelo Microsoft Bing Tradutor e o outro traduzido pelo Google Tradutor.

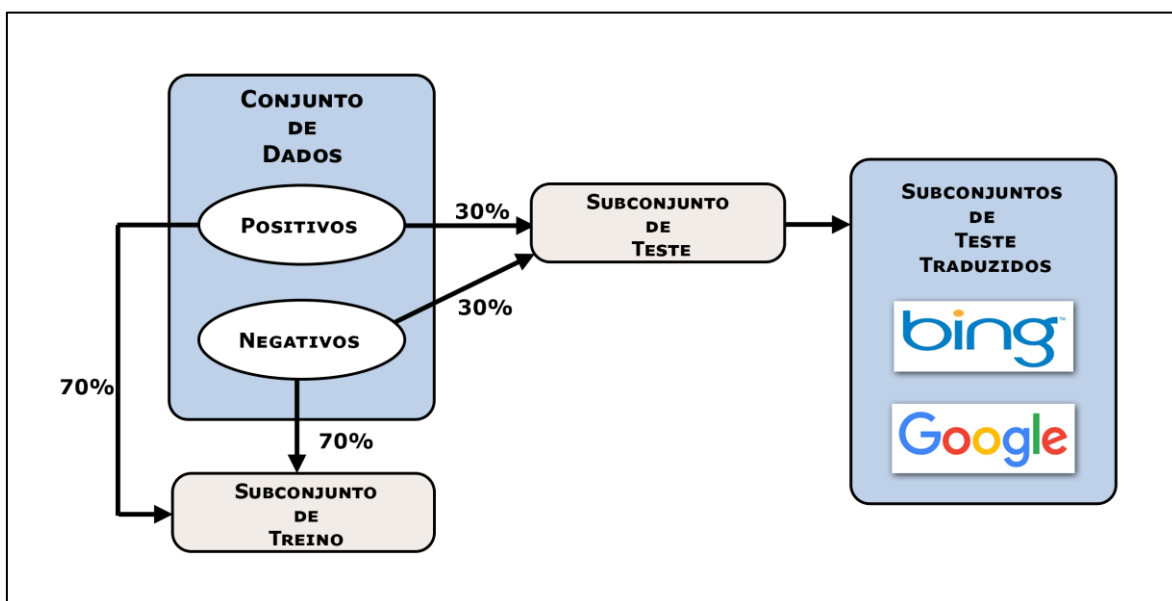


Figura 6.2 - Divisão dos conjuntos em subconjuntos de treino e teste e subsequente tradução dos conjuntos de teste.

Os subconjuntos de teste traduzidos não passaram por revisão manual da tradução. Todos os subconjuntos passaram pelas mesmas etapas de pré-processamento automáticas realizadas anteriormente nos dois conjuntos de dados originais.

Nosso primeiro experimento consiste no treinamento dos seguintes classificadores, utilizando os subconjuntos de treino gerados em inglês e português:

- Classificador Naive Bayes Multinomial para inglês
- Classificador Naive Bayes Multinomial para português
- Classificador SVM para inglês
- Classificador SVM para português

A Figura 6.3 apresenta os diagramas com os processos de treinamento e classificação realizados. Os modelos de classificação, Naive Bayes Multinomial e SVM, construídos a partir do subconjunto de treino em inglês, foram utilizados na classificação dos subconjuntos de teste traduzidos de português para inglês (PT-EN). Conseqüentemente, os modelos de classificação, Naive Bayes Multinomial e SVM, construídos a partir do subconjunto de treino em português, foram utilizados na classificação dos subconjuntos de teste traduzidos de inglês para português (EN-PT). Completando ao todo, oito processos de classificação de polaridade de sentimento.

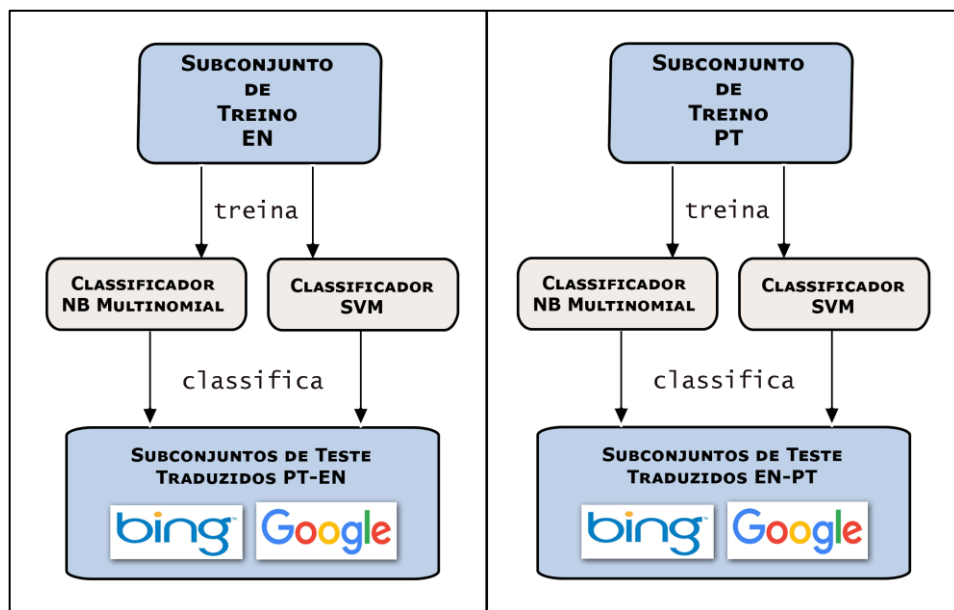


Figura 6.3 - Processos de treinamento dos classificadores e classificação dos subconjuntos traduzidos

Os resultados das classificações executadas podem ser observados nas Tabelas 6.3 e 6.4. Podemos observar que tanto nos processos realizados com classificadores para inglês, quanto nos processos com classificadores para português, com estes subconjuntos de dados, os processos de tradução automática não apresentaram grandes diferenças entre si. Na tradução EN-PT os dois sistemas de tradução automática contribuíram para métricas bem próximas em

seus valores. É notório, assim como nas métricas colhidas anteriormente e na literatura da área, que classificadores baseados no algoritmo SVM conseguem melhores acurácias em classificações binárias de texto.

		Subconjuntos de teste traduzidos PT-EN			
		Bing Tradutor		Google Tradutor	
		NBM	SVM	NBM	SVM
Classificadores	Métricas				
	Acurácia	75,09%	78,16%	75,78%	77,85%
	Precisão	0,758	0,780	0,765	0,777
	Recall	0,751	0,782	0,758	0,779
	F-measure	0,753	0,780	0,760	0,777

Tabela 6.3 - Resultados das classificações nos classificadores NB Multinomial e SVM com subconjunto de treino em inglês e subconjuntos de teste traduzidos PT-EN

		Subconjuntos de teste traduzidos EN-PT			
		Bing Tradutor		Google Tradutor	
		NBM	SVM	NBM	SVM
Classificadores	Métricas				
	Acurácia	73,02%	77,16%	73,02%	77,10%
	Precisão	0,736	0,769	0,735	0,769
	Recall	0,730	0,772	0,730	0,771
	F-measure	0,732	0,769	0,732	0,768

Tabela 6.4 - Resultados das classificações nos classificadores NB Multinomial e SVM com subconjunto de treino em português e subconjuntos de teste traduzidos EN-PT

Observamos que nas métricas colhidas anteriormente através das classificações dos dois conjuntos de dados originais, a classificação de textos escritos em inglês tem uma tendência a obter melhores resultados que a realizada em textos escritos em português. Com a tradução

automática dos textos, essa tendência continua sendo válida no caso estudado. Observando os dados obtidos apenas das classificações realizadas pelo classificador SVM, podemos notar que as métricas precisão e *recall*, fração do que foi classificado nas duas classes e fração do que foi classificado e realmente pertencia às duas classes, respectivamente, se mantêm em um intervalo de valores muito próximos, ou seja, estão bem balanceadas. Os valores das *F-measure* medidos se mostraram bastante satisfatórios.

Apesar dos valores das métricas medidas serem satisfatórios, é notório que as acurácias caíram após os processos de tradução no caso estudado, quando comparamos esses valores com os obtidos nas classificações dos subconjuntos de teste antes destes serem traduzidos. Contudo, apesar da queda nos valores, estes ainda se apresentaram satisfatórios, o que implica na necessidade de uma investigação mais profunda com volumes de dados maiores.

6.2.1 Análise do Segundo Experimento

Os dois conjuntos usados no primeiro experimento foram então divididos de maneira balanceada em subconjuntos de treino e teste, de forma que a quantidade de sentenças marcadas com as classes positiva e negativa fossem distribuídas em 70% de cada classe para o subconjunto de treino e 30% de cada classe para o conjunto de teste. Ao final das divisões, tínhamos um subconjunto de treino e um subconjunto de teste em inglês, e um subconjunto de treino e um subconjunto de teste em português.

Os subconjuntos de teste gerados foram submetidos à tradução automática em dois sistemas: Google Tradutor e Microsoft Bing Tradutor. As traduções não foram revisadas manualmente para que seus resultados fossem preservados.

Classificadores Naive Bayes Multinomial e SVM foram treinados com os dois subconjuntos de treino criados, obtendo classificadores para inglês e português. Os subconjuntos de teste traduzidos foram classificados pelos modelos de classificação criados. Observamos que para o caso apresentado, os dois sistemas de tradução automática utilizados não apresentaram diferença significativa na acurácia das classificações executadas. Contudo, acreditamos que torna-se necessária uma investigação com um maior volume de dados para averiguar se esse resultado tende a perdurar em outros casos.

As classificações realizadas pelos classificadores treinados com o subconjunto de dados em português, em subconjuntos de teste traduzidos de inglês para português, apresentaram acurácias entre 73% a 77%. Apesar de ser um resultado abaixo do obtido através da *cross validation* do conjunto em português, são valores satisfatórios de métricas de avaliação.

Já as classificações realizadas pelos classificadores treinados com o subconjunto de dados em inglês, nos subconjuntos de teste traduzidos português para o inglês, apresentaram acurácias entre 75% a 78%. Assim como as acurácias obtidas pelos classificadores em português, estes obtiveram em média 5 pontos abaixo dos valores obtidos através da *cross validation*, o que pode ser atribuído ao processo de tradução em si, gerando palavras que podem não estar presentes no conjunto de treino. Tal observação necessita de uma investigação mais aprofundada para avaliar em conjuntos de treino mais robustos se o decremento na acurácia continua a ocorrer ou não.

Outro ponto importante a ser mencionado é o ruído presente em dados extraídos de redes sociais *online* que possivelmente alteram os resultados nestes dados traduzidos automaticamente. O ruído se configura em gírias, abreviações, siglas, etc, que, mesmo com um processo extenso de tratamento e extração, ou até modificação destes termos, não há garantia que todo o ruído seja eliminado de maneira automática, sem intervenção humana. Por esse motivo, acreditamos que a diferença entre as acurácias pode também ser consequência da presença de palavras não traduzidas ou simplesmente não presentes no conjunto de treino, como consequência do processo de tradução automática. Entretanto, no caso estudado, os resultados se mostraram animadores com relação à quantidade de sentenças classificadas corretamente para as duas classes de sentimento, e em todas as métricas de avaliação coletadas.

6.3 Classificação de sentimento de conjuntos não previamente marcados e submetidos à tradução automática

Em nosso segundo experimento, utilizamos porções de dados aleatórias, em inglês e português, extraídas dos dados de nossa coleta, no mesmo contexto já trabalhado. O conjunto de treino dos classificadores são os dois conjuntos irmãos originais, em inglês e português, já trabalhados.

Os conjuntos a serem classificados passaram pelos mesmos passos de tratamento de texto, explicados no Capítulo 4 deste documento, que os conjuntos de treino originais. Nenhum conjunto teve sua escrita manualmente revisada nesse experimento (com exceção dos conjuntos de treino que foram revisados no experimento anterior). Sendo os dois classificadores c_1 e c_2 e duas línguas l_1 e l_2 , o experimento consiste em:

- Treinar um classificador c_1 na língua l_1 ;
- Classificar um conjunto de dados de texto escritos na língua l_1 ;
- Treinar um classificador c_2 na língua l_2 ;
- Traduzir o conjunto de dados de texto escrito em l_1 , e classificado por c_1 , para a língua l_2 ;
- Classificar o conjunto de dados traduzido para l_2 em c_2 ;
- Comparar as classificações do conjunto original em c_1 e do conjunto traduzido em c_2 .

Neste experimento optamos por utilizar apenas o algoritmo de classificação SVM, pois obteve maiores acurácias em nosso experimento anterior, e o sistema de tradução automática Microsoft Bing Translate, por ser mais simples de ser executado e verificamos que a mudança de sistema de tradução automática não acarretou em uma diferença significativa nas métricas coletadas em nosso primeiro caso estudado. A língua l_1 do conjunto inicial é português (PT) e a língua l_2 para a qual o conjunto foi traduzido é inglês (PT-EN), como pode ser observado na Figura 6.4.

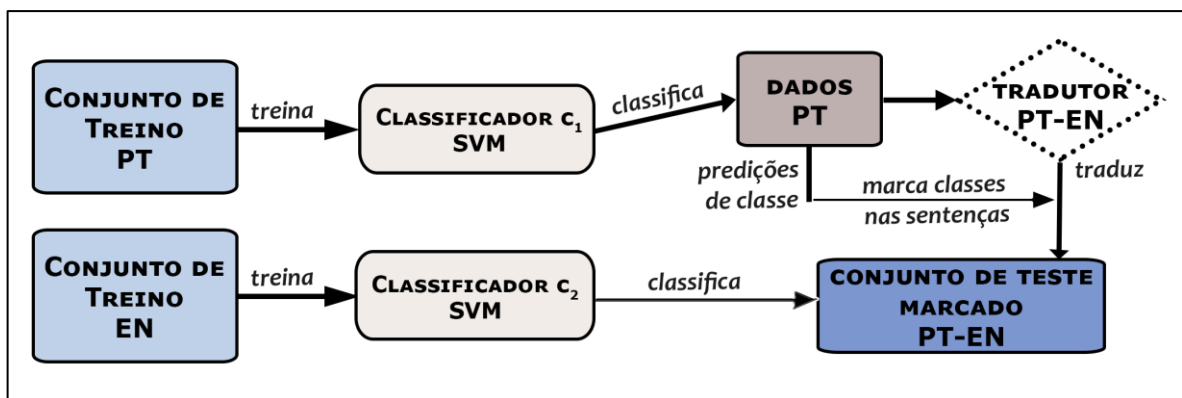


Figura 6.4 - Diagrama do funcionamento do experimento

O conjunto de treino em português foi utilizado para treinar um classificador SVM c_1 , ao passo que o conjunto de treino em inglês foi utilizado para treinar um classificador SVM, c_2 . O conjunto de dados em português, PT, sem marcação de classes é classificado por c_1 , em seguida o mesmo conjunto é traduzido através do sistemas de tradução automática Bing Tradutor, como resultado temos o conjunto traduzido de dados de português para inglês, PT-EN. As predições de classes das sentenças do conjunto PT são armazenadas e utilizadas para marcar as sentenças do conjunto PT-EN com suas correspondentes classes. O conjunto PT-EN marcado com classes é utilizado como conjunto de teste no classificador c_2 .

A classificação do conjunto PT realizada pelo classificador c_1 resultou em 1381 sentenças positivas e 1058 sentenças negativas. Os valores das médias das métricas de avaliação coletadas podem ser observados na Tabela 6.5. Além das métricas, alguns valores relacionados às marcações das sentenças e suas classificações são interessantes de serem mencionados:

- Positivos classificados como positivos = 1159;
- Positivos classificados como negativos = 222;
- Negativos classificados como positivos = 198;
- Negativos classificados como negativos = 860.

Acurácia	Precisão	Recall	F-measure
82,78%	0,828	0,828	0,828

Tabela 6.5 - Valores das médias das métricas de avaliação calculadas após classificação do conjunto PT-EN pelo classificador c_2

Apesar dos valores animadores, neste experimento as marcações das sentenças do conjunto de teste não tem garantia de estarem de fato corretas. Porém, o objetivo é comparar a classificação de um mesmo conjunto antes e após tradução automática sem revisão manual da mesma. A fração de sentenças classificadas igualmente nos dois conjuntos foi alta, quase cinco vezes o número de sentenças classificadas erroneamente.

6.3.1 Análise do Terceiro Experimento

Nosso último experimento realizado objetivou a comparação entre classificações de dados brutos em português extraídos de redes sociais *online* e estes mesmos dados traduzidos para inglês. Os conjuntos de treino utilizados neste experimento foram os primeiros criados no primeiro experimento, em inglês e português, revisados manualmente. O experimento consistiu no treinamento de dois classificadores SVM, um em português, outro em inglês, e na classificação de dados brutos em português e destes traduzidos para inglês.

Os dados brutos em português não passaram por revisão manual, passando apenas pelas etapas de pré-processamento de dados comuns a todos os conjuntos de dados em português. Esses dados de texto não foram marcados previamente com as classes de sentimento correspondentes às suas sentenças e foram classificados pelo classificador de sentimento em português, que gerou as predições das classificações das sentenças do conjunto em questão.

O conjunto de dados em português foi traduzido pelo tradutor automático Microsoft Bing para a língua inglesa. As marcações de classes de sentimento do conjunto em português foram aplicadas ao conjunto traduzido, gerando um conjunto de teste. Este conjunto também não foi revisado manualmente.

Os resultados da classificação, realizada pelo classificador de sentimento em inglês, dos dados de texto traduzido, mostraram uma alta acurácia de 82% e valores de outras métricas de avaliação igualmente altos.

Neste caso, podemos observar que o aumento do volume do conjunto de treino para a classificação nos dois idiomas pode ter uma consequência positiva no aumento da acurácia das classificações. É interessante observar que as predições realizadas pelos dois classificadores, sem intervenção humana nas marcações prévias de classes de sentimento das sentenças, mostraram bons resultados comparados entre si.

Capítulo 7

Considerações finais

Neste capítulo apresentaremos, primeiramente, as considerações finais do nosso trabalho, considerando as expectativas anteriores às execuções dos experimentos desenvolvidos, relativas aos resultados já obtidos em alguns trabalhos relacionados apresentados anteriormente, como também às conclusões que podemos extrair dos nossos resultados obtidos. Em seguida, apresentaremos os trabalhos futuros baseados nos resultados e conclusões do presente trabalho.

7.1 Considerações finais

Os experimentos realizados em nosso trabalho mostraram resultados satisfatórios em métricas de avaliação e comparados aos resultados obtidos em trabalhos de outros autores em experimentos semelhantes de tradução de dados de texto e classificação de polaridade de sentimento. Contudo, apesar dos resultados animadores, não podemos afirmar que estes são constantes para todos os casos, devido ao volume de dados utilizado em nossos experimentos não ser tão grande. Julgamos necessária uma investigação mais aprofundada relacionada tanto ao processo de tradução automática quanto à presença de ruído decorrente desses processos de tradução.

Podemos concluir que os sistemas de tradução automática utilizados neste trabalho apresentam uma tendência de traduções equiparadamente eficientes, mostrando que esses sistemas evoluíram bastante nos últimos anos. Quanto à classificação de dados de texto traduzidos automaticamente, através dos dois sistemas de tradução abordados neste trabalho, podemos dizer que, a partir dos resultados obtidos, a tradução automática de texto pode apresentar bons resultados para alguns casos. Porém, há a necessidade de experimentação com volumes de dados de treino mais abrangentes nas duas línguas estudadas neste documento.

Como principais contribuições desse trabalho podemos destacar:

- implementação de um *crawler* personalizado para coleta de dados da rede social Twitter;
- implementação de um classificador *Naive Bayes* simples para classificação de texto com a possibilidade de utilização de grandes volumes de dados;
- a avaliação da qualidade da tradução automática de texto de português para inglês e inglês para português, utilizando os sistemas de tradução Google Tradutor e Microsoft Bing Tradutor em alguns cenários de classificação de polaridade de sentimento;
- a avaliação do impacto da utilização de dados traduzidos de inglês para português e de português para inglês em alguns cenários de classificação de polaridade de sentimento;
- a publicação de um trabalho inicial [13] que apresentou um estudo simples sobre identificação de tendências no mercado de *smartphones* entre os usuários de redes sociais utilizando dados extraídos da rede social Twitter, onde pudemos verificar a validade desses dados perante pesquisas realizadas no mundo real.
- códigos das ferramentas desenvolvidas disponíveis em repositório *online* [49].

Devido às limitações de tempo e a necessidade de utilização de revisão manual dos dados em alguns casos, os resultados obtidos, apesar de satisfatórios, não podem ser encarados como conclusivos para o problema.

7.2 Trabalhos Futuros

A partir dos resultados obtidos e conclusões do presente trabalho, podemos dar uma continuidade ao nosso primeiro estudo publicado sobre tendências no mercado de *smartphones* aplicando agora a classificação de polaridade de sentimento para calcular uma tendência opinativa através das postagens dos usuários das redes sociais. Considerando também os resultados obtidos nos trabalhos relacionados, onde algumas línguas latinas com estruturas semelhantes ao português (italiano e espanhol) também apresentaram tendências à acurácias satisfatórias em classificações de polaridade de sentimento, podemos incluir essas línguas no conjunto de idiomas abordados em nossa pesquisa. Infelizmente, para uma análise global dos dados, julgamos necessário um estudo de impacto de traduções automáticas abordando mais idiomas. Finalmente, nosso primeiro estudo poderá ter continuidade com a abordagem de certos idiomas, sendo estes os estudados neste trabalho e nos relacionados.

Como destacamos anteriormente a necessidade de um maior volume de dados de treino para os classificadores de sentimento, pretendemos continuar a preparação de dados de texto, com o intuito de formar bases de dados mais robustas e confiáveis para futuros estudos.

Além da continuidade dos estudos já realizados, pretendemos realizar análises de sentimento explorando mais emoções e analisando o contexto para a identificação mais confiável de opinião em dados de texto.

Bibliografia

- [1] Tumasjan, Andranik, Timm O. Sprenger, Philipp G. Sandner, and Isabell M. Welp. "Predicting elections with twitter: What 140 characters reveal about political sentiment." In Proceedings of the fourth international aaai conference on weblogs and social media, pp. 178-185. 2010.
- [2] "Strategies for Effective Tweeting: A Statistical Review", in www.salesforce.com/marketing-could, pp.06-07, 2012.
- [3] Erika Jurisová. "The impact of social networking on business and business ethics", 2013.
- [4] Sitaram Asur, Bernardo A. Huberman, "Predicting the Future with Social Media," *wiat*, vol. 1, pp.492-499, 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2010.
- [5] David Alfred Ostrowski. "Identification of Trends in Consumer Behavior through Social Media", 2013.
- [6] David Alfred Ostrowski. "Social Network Analysis for Consumer Behavior Prediction", 2012.
- [7] <http://www.ibtimes.com/android-market-share-nears-52-percent-apple-iphone-still-most-popular-device-us-723349>, acessado em março de 2015.
- [8] <http://www.wpcentral.com/its-official-windows-phone-third-most-popular-smartphones>, acessado em março de 2015.
- [9] <http://dev.twitter.com/docs/api/1.1/get/search/tweets>, acessado em março de 2015.
- [10] Wison T., Wiebe J., Hoffman P., "Recognizing Contextual Polarity in phrase-level Sentiment Analysis", 2005.
- [11] <http://www.twitter.com>, acessado em março de 2015.
- [12] Sarita Yard, Daniel Romero, Grant Schoenebeck, Danah Boyd, "Detecting Spam in a Twitter Network", *First Monday – Peer-reviewed Journal on the Internet*, Vol.15, 2010
- [13] Evelyn Farias, Reinaldo Gomes, "Social Network Analysis for Market Trends Identification: A Preliminary Study", *WWW/INTERNET International Conference*, 2014.
- [14] Alexander Pak, Patrick Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining", 2010
- [15] <http://www.facebook.com>, acessado em março de 2015.
- [16] <http://www.tumblr.com>, acessado em março de 2015.

- [17] <http://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/>
- [18] M. Baumgarten, M.D. Mulvenna, N.Rooney, “Keyword-Based Sentiment Mining using Twitter”, *International Journal of Ambient Computing and Intelligence*, 5(2), 56-69, April-June, 2013
- [19] Ajit Pai, “The government wants to study ‘social pollution’ on Twitter”, 2014.
- [20] Erika Fry, “CONTAGION — From Justin Bieber to data scientists, how Twitter got hot in the academy”, 2014.
- [21] Enrique Rivero, “Twitter ‘Big data’ can be used to monitor HIV and drug-related behavior”, 2014.
- [22] I-Hsien Ting, Shyue-Liang Wang, Hsing-Miao Chi, Jyun-Sing Wu, “Content matters: a study of hate groups detection based on social networks analysis and web mining”, *ASONAM '13 Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 1196-1201, New York, 2013
- [23] Aibek Makazhanov, Davood Rafiei, “Predicting Political preferences of Twitter Users”, *ASONAM '13 Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 298-305, 2013
- [24] Lu Chen, Ingmar Weber, Adam Okulicz-Kozaryn, “U.S. Religious Landscape on Twitter”, *the 6th International Conference on Social Informatics (SocInfo 2014)*, 2014
- [25] Dr. M. Mohamed Sathik, A. Abdul Rasheed, “Discovering Communities in Social Networks Through Mutual Accessibility”, *(IJCSE) International Journal on Computer Science and Engineering*, Vol. 02, No. 04, pp. 1423-1428, 2010
- [26] Mrinmaya Sachan, Danish Contractor, Tanveer A. Faruque, L.Venkata Subramaniam, “Using Content and Interactions for Discovering Communities in Social Networks”, *WWW 2012 - Session: Community Detection in Social Networks*, April 16-20, Lyon, France, 2012
- [27] Alexandra Balahur, Marco turchi, “Improving Sentiment Analysis in Twitter Using Multilingual Machine Translated Data”, 2013.
- [28] Alexander Pak, Patrick Paroubek, “Twitter as a Corpus for Sentiment Analysis and Opinion Mining”, 2010
- [29] Eduardo Duarte, “Sentiment Analysis on Twitter for the Portuguese Language”, 2013.

- [30] André Luiz Firmino Alves, Cláudio de Souza Baptista, Anderson Almeida Firmino , Maxwell Guimarães de Oliveira , Anselmo Cardoso de Paiva, “A Spatial and Temporal Sentiment Analysis Approach Applied to Twitter Microtexts”, 2015.
- [31] Yiou Lin, Hang Lei, Jia Wu, Xiaoyu Li, “An Empirical Study on Sentiment Classification of Chinese Review using Word Embedding”, 2015.
- [32] Navanath Saharia, “Detecting Emotion from Short Messages on Nepal Earthquake”, International Conference of Speech, Technology and Human – Computer Dialogue (SpeD), 2015
- [33] Alexandra Balahur, Marco Turchi, “Multilingual Sentiment Analysis using Machine Translation?”, 2012.
- [34] Hong Yu, Vasileios Hatzivassiloglou, “Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences”, 2003.
- [35] B. Liu, “Sentiment Analysis and Opinion Mining”, 2012.
- [36] Walaa Medhat, Ahmed Hassan, Hoda Korashy, “Sentiment analysis algorithms and applications: a survey”, 2014.
- [37] Thorsten Joachims, “Text Categorization with support vector machines: learning with many Relevant Features”, 1998.
- [38] <https://oauth.net>, acessado em 25 de junho de 2016.
- [39] https://en.wikipedia.org/wiki/Representational_state_transfer, acessado em 25 de junho de 2016.
- [40] <https://class.coursera.org/nlp/lecture>, acessado em 10 de junho de 2016.
- [41] Rafaela Pozzebon, “Qual a diferença entre robô, spider e crawler”, 2011.
- [42] <http://bsonspec.org/>, acessado em 25 de junho de 2016.
- [43] <http://mashable.com/2013/12/17/twitter-popular-languages/>, acessado em 02 de julho de 2016.
- [44] <http://www.statista.com/statistics/262946/share-of-the-most-common-languages-on-the-internet/>, acessado em 02 de julho de 2016.
- [45] <http://snowballstem.org/>, acessado em 02 de julho de 2016.
- [46] <http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/>, acessado em 02 de julho de 2016.
- [47] <http://www.sananalytics.com/lab/twitter-sentiment/>, acessado em 02 de julho de 2016.
- [48] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, “Introduction to Information Retrieval”, Cambridge University Press. 2008.

[49] <https://github.com/evelynsf?tab=repositories>, acessado em 21 de setembro de 2016.

Apêndice A

Código Fonte do *crawler* de dados do Twitter

Código Fonte A.1: Classe que implementa o crawler de dados do Twitter

```
1 import java.io.BufferedReader;
2 import java.io.FileNotFoundException;
3 import java.io.IOException;
4 import java.io.InputStreamReader;
5 import java.net.ConnectException;
6
7 import crawler.storage.TweetCollection;
8 import crawler.util.*;
9 import twitter4j.FilterQuery;
10 import twitter4j.TwitterException;
11 import twitter4j.TwitterStream;
12 import twitter4j.TwitterStreamFactory;
13
14 /**
15  * Classe construtora do crawler
16  * @author Evelyn
17  *
18  */
19 public class Crawler {
20
21     private TwitterStream twitterStream;
22     private TweetListener listener;
23     private Query query;
24
25     /*
26     * Crawlers constructor
27     */
28     public Crawler() throws IllegalStateException, TwitterException,
29     FileNotFoundException, IOException{
30         this.start();
31     }
32
33     /*
34     * Start connection via OAuth to Twitter server, tweets stream
35     filtered by a query
36     */
37     private void start() throws IllegalStateException, TwitterException,
38     FileNotFoundException, IOException {
39
40         OAuthUser oauth = new OAuthUser();
41
42         listener = new TweetListener        ();
43         twitterStream = new
44         TwitterStreamFactory(oauth.build()).getInstance();
45         twitterStream.addListener(listener);
46
47         //Stream of all flow of tweets (no filter query)
48         //twitterStream.sample();
```



```

49
50         query = new Query();
51         if(query.isEmpty()){
52             System.out.println("Twitter SAMPLE stream is
53 running...");
54             twitterStream.sample();
55         }
56         else{
57
58             //FilterQuery(int count, long[] follow,
59 java.lang.String[] track, double[][] locations, java.lang.String[]
60 language)
61             twitterStream.filter(new FilterQuery(0, null,
62 query.getKeywords(), null, query.getLanguages() ));
63
64             System.out.println("Twitter stream is running...");
65         }
66         onInputExit(); //stops the streamming
67     }
68
69     /*
70     * Stop streamming if user puts exit on keyboard
71     */
72     public void onInputExit() throws IOException{
73
74         BufferedReader br = new BufferedReader(new
75 InputStreamReader(System.in));
76         String input = br.readLine();
77
78         if (input.toLowerCase().equalsIgnoreCase("exit")){
79             this.stop();
80         }
81     }
82
83     /*
84     * Stop the streaming
85     */
86     private void stop(){
87         System.out.println("The Twitter stream is shutting down...!\n"
88 +
89             listener.getColected() + " tweets coletados");
90         twitterStream.shutdown();
91     }
92
93
94
95 }

```

Código Fonte A.2: Classe que implementa o *listener* da API do Twitter

```

1 package crawler.util;
2
3 import java.io.File;
4 import java.io.FileWriter;

```

```

5  import java.io.IOException;
6  import java.io.PrintWriter;
7  import crawler.storage.TweetCollection;
8  import twitter4j.StallWarning;
9  import twitter4j.Status;
10 import twitter4j.StatusDeletionNotice;
11 import twitter4j.StatusListener;
12
13 /**
14  * Listener for twitter stream
15  * @author Evelyn
16  *
17  */
18 public class TweetListener implements StatusListener {
19
20
21     private TimeStamp runtime;
22     private Tweet tweet;
23     private File file;
24     private int collected;
25     private int stored;
26     private ConfigurationFile results;
27     private TweetCollection tweetCollection;
28
29
30     public TweetListener() throws IOException{
31
32         tweetCollection = new TweetCollection("Tweets", "test");
33         runtime = new TimeStamp(); //catch the current daytime
34         this.setCollected(0);
35         this.setStored(0);
36         setResults(new ConfigurationFile("output/" + runtime.getTimeFormat()
37 + ".txt"));
38     }
39
40     public void onStatus(Status status) {
41
42         tweet = new Tweet(status.getText(),status.getUser().getName(),
43 status.getUser().getScreenName(),
44             status.getCreatedAt().toString(),
45 status.getUser().getLocation());
46
47         tweetCollection.insert(status);
48         //System.out.println(tweet.toString());
49
50         try {
51
52             file = new File("tweets.txt");
53             PrintWriter writer = new PrintWriter(new FileWriter(file,
54 true));
55             writer.append(tweet.toString() + "\n");
56             stored++;
57             writer.close();
58
59
60         } catch ( IOException e ) {
61             e.printStackTrace();
62         }
63         this.setCollected(this.getCollected() + 1);
64         try {
65             results.writeProperties(getCollected(), stored);

```

```

66         } catch (IOException e) {
67             // TODO Auto-generated catch block
68             e.printStackTrace();
69         }
70
71
72         /*
73             System.out.println("[ " +
74                 //new GregorianCalendar().getTime().toString()
75                 time.getTimeFormat()
76                 + " ] "
77                 + status.getUser().getName() + " : " +
78 status.getText());
79         */
80     }
81
82     public void onDeleteNotice(StatusDeletionNotice statusDeletionNotice)
83     {
84         //System.out.println("Got a status deletion notice id:" +
85 statusDeletionNotice.getStatusId());
86     }
87
88     public void onTrackLimitationNotice(int numberOfLimitedStatuses) {
89         //System.out.println("Got track limitation notice:" +
90 numberOfLimitedStatuses);
91     }
92
93     public void onException(Exception ex) {
94         ex.printStackTrace();
95     }
96
97     @Override
98     public void onScrubGeo(long userID, long upToStatusID) {
99         System.out.println("Got scrub_geo event userID:" + userID + "
100 upToStatusId:" + upToStatusID);
101     }
102
103     @Override
104     public void onStallWarning(StallWarning stallWarning) {
105         System.out.println("Got stall warning:" + stallWarning);
106     }
107
108
109     public ConfigurationFile getResults() {
110         return results;
111     }
112
113     public void setResults(ConfigurationFile results) {
114         this.results = results;
115     }
116
117     public int getStored() {
118         return stored;
119     }
120
121     public void setStored(int stored) {
122         this.stored = stored;
123     }
124
125     public int getCollected() {
126         return collected;

```

```

127     }
128
129     public void setColected(int colected) {
130         this.colected = colected;
131     }
132
133     public TimeStamp getRuntime() {
134         return runtime;
135     }
136
137     public void setRuntime(TimeStamp runtime) {
138         this.runtime = runtime;
139     }
140
141 }

```

Código Fonte A.3: Classe que implementa uma *query* de busca de dados do Twitter

```

1  package crawler.util;
2
3  import java.io.FileInputStream;
4  import java.io.FileNotFoundException;
5  import java.io.IOException;
6  import java.util.Properties;
7
8  /**
9   * Twitter query class
10  * @author Evelyn
11  *
12  */
13  public class Query{
14
15      public final static String[] LANGUAGES = {"pt"};
16
17      private String[] languages;
18      private String[] keywords;
19      private boolean isSample = false;
20
21      private Properties properties;
22
23      /*
24       * Constructor
25       */
26      public Query() throws FileNotFoundException, IOException{
27          this.languages = null;
28          this.keywords = null;
29          this.build();
30      }
31
32
33      /*
34       * Loads the query properties from a input config file
35       */
36      private Properties load() throws FileNotFoundException, IOException{
37

```

```

38         properties = new Properties();
39
40         try{
41             properties.load(new
42 FileInputStream("input/twitterQuery.config"));
43         }
44         catch(FileNotFoundException fnfe){
45             System.out.println("Twitter Query configuration - file
46 not found." + fnfe.getMessage());
47             isSample = true;
48         }
49         catch(IOException ioe){
50             System.out.println("Twitter Query Configuration - Error
51 reading file" + ioe.getMessage());
52         }
53
54         return properties;
55     }
56
57     /*
58      * Read string transforming it in vector of strings
59      */
60     private String[] readString(Properties p, String prop){
61
62         String[] output = null;
63
64         String aux = p.getProperty(prop);
65         if ((aux.equalsIgnoreCase(null) ||
66 aux.trim().equalsIgnoreCase(""))){
67
68             if(this.languages != null &&
69 prop.equalsIgnoreCase("keywords")){
70
71                 System.out.println(prop.toUpperCase() + " - fix it
72 in twitterQuery.config file."
73 + "\nApplication stopped");
74                 System.exit(0);
75             }
76             else{
77                 System.out.println(prop.toUpperCase() + " - none");
78             }
79         }
80         else{
81             output = aux.split(",");
82             for (int i = 0; i < output.length; i++){
83                 output[i] = output[i].trim();
84             }
85             System.out.println          (prop.toUpperCase() + " = "
86 + propString(output));
87         }
88
89         return output ;
90     }
91
92     private String propString(String[] in){
93         String out = in[0];
94         if (in.length > 1){
95             for (int i = 1; i < in.length; i++){
96                 out += ", " + in[i];
97             }
98         }

```

```

99         return out;
100     }
101
102     /*
103     * Build a query with its settings
104     */
105     public Query build() throws FileNotFoundException, IOException{
106
107         properties = this.load();
108         System.out.println("JTwitterCrawler - Search Query Settings");
109
110         this.languages = readString(properties, "languages");
111         this.keywords = readString(properties, "keywords");
112
113         return this;
114     }
115
116     public String[] getLanguages() {
117         return languages;
118     }
119
120     public void setLanguages(String[] languages) {
121         this.languages = languages;
122     }
123
124     public String[] getKeywords() {
125         return keywords;
126     }
127
128     public void setKeywords(String[] keywords) {
129         this.keywords = keywords;
130     }
131
132
133     public boolean isSample() {
134         return isSample;
135     }
136
137
138     public void setSample(boolean isSample) {
139         this.isSample = isSample;
140     }
141
142     public boolean isEmpty(){
143         boolean flag;
144         if( this.languages == null && this.keywords == null)
145             flag = true;
146         else flag = false;
147         return flag;
148     }
149 }
150

```

Código Fonte A.4: Classe que implementa o arquivo de configuração do *crawler* de dados do Twitter

```

1  package crawler.util;
2
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.FileNotFoundException;
6  import java.io.FileWriter;
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import java.util.Properties;
10
11  /**
12   * Configuration file class - object
13   * Manipulates files - create, write, read
14   * @author Evelyn
15   *
16   */
17  public class ConfigurationFile {
18
19      private String filename;
20      private File file;
21      private Properties properties;
22      //private int run;
23
24      public ConfigurationFile(String namePath) throws IOException{
25          this.setFilename(namePath);
26          createFile();
27          loadProperties();
28      }
29
30      //read number_of_runs
31      // if number_of_runs == 0
32      // increment number_of_runs AND create file results1
33      // else create file
34
35      private void createFile() throws IOException{
36
37          this.file = new File(getFilename());
38          if (!file.exists()){
39              try{
40                  file.createNewFile();
41                  System.out.println("File " + file.getName() + "
42 created successfully.");
43              }
44              catch(IOException ioe){
45                  System.out.println("File cannot be created." +
46 ioe.getMessage());
47                  System.exit(0);
48              }
49          }
50          else{
51              System.out.println("File " + file.getName() + " found.");
52          }
53      }
54
55      private void loadProperties() throws FileNotFoundException,
56 IOException{
57
58          properties = new Properties();

```

```

59         try{
60             properties.load(new
61 FileInputStream(this.file.getPath()));
62             System.out.println("Loading properties from file...");
63             if (properties.isEmpty()){
64                 System.out.println("No properties found.");
65                 System.out.println("Setting properties and writing
66 in file.");
67
68                 //properties.put("Number_of_Runs", "0");
69                 //this.run = 0;
70                 properties.put("Colected_Tweets", "");
71                 properties.put("Stored_Tweets", "");
72
73                 PrintWriter ptw = new PrintWriter(new
74 FileWriter(file), true);
75                 properties.store(ptw, "Properties");
76                 ptw.close();
77             }
78             else{
79                 //this.run =
80 Integer.parseInt(properties.getProperty("Number_of_Runs"));
81                 //this.setRun(this.run + 1);
82                 //properties.put("Colected_Tweets" + getRun(), "");
83                 //properties.put("Stored_Tweets" + getRun(), "");
84             }
85         }
86         catch(NullPointerException npe){
87             System.out.println("Not possible to get properties from
88 the file " + this.file.getName());
89             System.exit(0);
90         }
91         catch(FileNotFoundException fnfe){
92             System.out.println("File not found." +
93 fnfe.getMessage());
94             System.exit(0);
95         }
96     }
97
98     public void writeProperties(int colected, int stored) throws
99 IOException{
100
101         //colected = colected +
102 Integer.parseInt(this.properties.getProperty("Colected_Tweets"));
103         //String numero = "" + this.run;
104         //if (this.run == 0) numero = "";
105
106         //this.properties.setProperty("Number_of_Runs", "" + this.run);
107         //this.properties.setProperty("Colected_Tweets" + numero, "" +
108 colected);
109         //this.properties.setProperty("Stored_Tweets" + numero, "" +
110 stored);
111
112         this.properties.setProperty("Colected_Tweets", "" + colected);
113         this.properties.setProperty("Stored_Tweets", "" + stored);
114         PrintWriter ptw = new PrintWriter(new FileWriter(file), false);
115         properties.store(ptw, "Properties");
116         ptw.close();
117     }
118 }
119

```



```

120     public File getFile() {
121         return file;
122     }
123
124     public void setFile(File file) {
125         this.file = file;
126     }
127
128     public Properties getProperties() {
129         return properties;
130     }
131
132     public void setProperties(Properties properties) {
133         this.properties = properties;
134     }
135
136     public String getFilename() {
137         return filename;
138     }
139
140     public void setFilename(String filename) {
141         this.filename = filename;
142     }
143
144 }

```

Código Fonte A.5: Classe que implementa a autenticação OAuth no crawler de dados do Twitter

```

1  package crawler.util;
2
3  import java.io.FileInputStream;
4  import java.io.FileNotFoundException;
5  import java.io.IOException;
6  import java.util.Properties;
7  import twitter4j.conf.Configuration;
8  import twitter4j.conf.ConfigurationBuilder;
9
10 /**
11  * OAuth user class - OAuth authentication
12  *
13  * @author Evelyn
14  *
15  */
16 public class OAuthUser {
17
18     private Properties loadProperties() throws FileNotFoundException,
19     IOException{
20
21         Properties properties = new Properties();
22         try{
23             properties.load(new
24             FileInputStream("input/oauthKeys.config"));

```

```

25         }
26         catch(FileNotFoundException fnfe){
27             System.out.println("Oauth keys configuration - file not
28 found." + fnfe.getMessage());
29             System.exit(0);
30         }
31         catch(IOException ioe){
32             System.out.println("Oauth Keys Configuration - Error
33 reading file" + ioe.getMessage());
34             System.exit(0);
35         }
36     }
37     return properties;
38 }
39
40 //public Configuration build(String consumerKey, String
41 consumerSecret, String accessToken, String accessSecret) throws
42 FileNotFoundException, IOException {
43     public Configuration build() throws FileNotFoundException,
44 IOException {
45
46         ConfigurationBuilder cb = new ConfigurationBuilder();
47
48         Properties properties = loadProperties();
49
50         cb.setDebugEnabled(true)
51             .setOAuthConsumerKey(properties.getProperty("consumerKey"))
52
53             .setOAuthConsumerSecret(properties.getProperty("consumerSecret"))
54             .setOAuthAccessToken(properties.getProperty("accessToken"))
55
56             .setOAuthAccessTokenSecret(properties.getProperty("accessTokenSecret"
57 ));
58
59         return cb.build();
60     }
61 }

```

Código Fonte A.6: Classe que implementa um *tweet* no *crawler* de dados do Twitter

```

1 package crawler.util;
2
3 import twitter4j.GeoLocation;
4 import twitter4j.HashtagEntity;
5 import twitter4j.MediaEntity;
6 import twitter4j.Place;
7 import twitter4j.Scopes;
8 import twitter4j.Status;
9 import twitter4j.SymbolEntity;
10 import twitter4j.URLEntity;
11 import twitter4j.User;
12 import twitter4j.UserMentionEntity;
13
14 /**

```

```

15  * Tweet object implementation
16  * @author Evelyn
17  *
18  */
19  public class Tweet{
20
21      private String tweet;
22      private String userName;
23      private String userScreenName;
24      private TimeStamp timeStamp;
25      private String location;
26      private String timeCreated;
27
28
29      public Tweet(){
30          this.setTweet(null);
31          this.setUserName(null);
32          this.setUserScreenName(null);
33          this.setTimeStamp(null);
34          this.setLocation(null);
35      }
36
37
38      public Tweet(String twt, String name, String screenName, TimeStamp
39  timestamp,
40          String local){
41
42          this.setTweet(twt);
43          this.setUserName(name);
44          this.setUserScreenName(screenName);
45          this.setTimeStamp(timestamp);
46          this.setLocation(local);
47      }
48
49      /*
50       * Uses the status.createdAt() method
51       */
52      public Tweet(String twt, String name, String screenName, String
53  timestamp,
54          String local){
55
56          this.setTweet(twt);
57          this.setUserName(name);
58          this.setUserScreenName(screenName);
59          this.setTimeCreated(timestamp);
60          this.setLocation(local);
61      }
62
63      public String toString(){
64          return
65              //this.timeStamp.getDayFormatBR() + " "
66              //+ this.timeStamp.getHourFormat()
67              this.getTimeCreated()
68              + " " + this.getLocation() + " "
69          + this.getUserName() + " "
70          + this.getUserScreenName() + " "
71          + this.getTweet();
72      }
73
74      public String getTweet() {
75          return tweet;

```

```

76     }
77
78     public void setTweet(String tweet) {
79         this.tweet = tweet;
80     }
81
82     public String getUsername() {
83         return userName;
84     }
85
86     public void setUsername(String userName) {
87         this.userName = userName;
88     }
89
90     public String getUserScreenName() {
91         return userScreenName;
92     }
93
94     public void setUserScreenName(String userScreenName) {
95         this.userScreenName = userScreenName;
96     }
97
98     public Timestamp getTimeStamp() {
99         return timeStamp;
100    }
101
102    public void setTimeStamp(Timestamp timeStamp) {
103        this.timeStamp = timeStamp;
104    }
105
106
107    public String getLocation() {
108        return location;
109    }
110
111    public void setLocation(String location) {
112        this.location = location;
113    }
114
115    public String getTimeCreated() {
116        return timeCreated;
117    }
118
119    public void setTimeCreated(String timeCreated) {
120        this.timeCreated = timeCreated;
121    }
122
123
124
125 }

```

Código Fonte A.7: Classe que representa a marcação de tempo em um *tweet*

```

1 package crawler.util;
2

```

```

3 import java.text.SimpleDateFormat;
4 import java.util.Calendar;
5
6 /**
7  * Time stamp configuration
8  * @author Evelyn
9  *
10 */
11 public class TimeStamp {
12
13     public static final String TIME_FORMAT = "yyyy-MM-dd_HH-mm-ss";
14     public static final String DAY_FORMAT = "yyyy-MM-dd";
15     public static final String DAY_FORMAT_BR = "dd-MM-yyyy";
16     public static final String HOUR_FORMAT = "HH:mm:ss";
17     public static final String DAY_WEEK_MONTH_YEAR = "EEE MMM d yyyy";
18
19     private Calendar calendar;
20     private SimpleDateFormat date;
21
22     /*
23     * Constructor
24     */
25     public TimeStamp(){
26         calendar = Calendar.getInstance();
27     }
28
29     private String getDate(String format){
30         date = new SimpleDateFormat(format);
31         return date.format(calendar.getTime());
32     }
33
34     public String getTimeFormat(){
35         return getDate(TIME_FORMAT);
36     }
37
38     public String getDateFormat(){
39         return getDate(DAY_FORMAT);
40     }
41
42     public String getDayFormatBR(){
43         return getDate(DAY_FORMAT_BR);
44     }
45
46     public String getHourFormat(){
47         return getDate(HOUR_FORMAT);
48     }
49
50     public String getDayWeekFormat(){
51         return getDate(DAY_WEEK_MONTH_YEAR);
52     }
53 }

```

Código Fonte A.8: Classe que implementa a persistência dos dados coletados pelo crawler de dados do Twitter, através da integração com a biblioteca MongoDB

```

package crawler.storage;

import java.net.UnknownHostException;
import org.bson.Document;

import twitter4j.Status;

import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
/**
 * Implements a data collection of tweets -
 * @author Evelyn
 *
 */
public class TweetCollection{

    private String collectionName;
    private String dbname;
    private MongoClient client;
    private MongoDatabase tweetDB;
    private MongoCollection<Document> collection;

    public TweetCollection(String collection, String db) throws
UnknownHostException{

        this.collectionName = collection;
        this.dbname = db;

        this.client = new MongoClient();

        this.tweetDB = client.getDatabase(dbname);
        this.collection =
this.tweetDB.getCollection(collectionName);
    }

    /**
     * inserts full status object
     */
    public void insertFullStatus(Status status){
        Document doc = new Document().append("status", status);
        this.collection.insertOne(doc);
    }

    public void insert(Status status){

        Document document = new
Document().append("id", status.getId())
                .append("username",
status.getUser().getName())
                .append("userScreenName",
status.getUser().getScreenName())
                .append("timestamp",
status.getCreatedAt().toString())
                .append("text", status.getText())
                .append("language", status.getLang())
                .append("location",
status.getUser().getLocation())
                /*
                .append("geoLocation",
status.getGeoLocation())

```

```

                .append("place", status.getPlace()) //Place
has many methods (including getCountry() )
                .append("user", status.getUser()) //Object
user
                .append("accessLevel",
status.getAccessLevel())
                .append("contributors",
status.getContributors())
                .append("favoritedCount",
status.getCurrentUserRetweetId())
                .append("inReplyToScreenName",
status.getInReplyToScreenName())
                .append("inReplyToStatusID",
status.getInReplyToStatusId())
                .append("inReplyToUserID",
status.getInReplyToUserId())
                .append("retweetCount",
status.getRetweetCount())
                .append("hashCode", status.hashCode())
                .append("source", status.getSource())
                .append("isFavorited", status.isFavorited())
                .append("isPossiblySensitive",
status.isPossiblySensitive())
                .append("isRetweet", status.isRetweet())
                .append("isRetweeted", status.isRetweeted())
                .append("isRetweetedByMe",
status.isRetweetedByMe())
                .append("isTruncated", status.isTruncated())
                .append("hashtagEntities",
status.getHashtagEntities())
                .append("scopes", status.getScopes())
                .append("URLEntities",
status.getURLEntities())
                .append("symbolEntities",
status.getSymbolEntities())
                .append("userMentionEntities",
status.getUserMentionEntities())
                .append("mediaEntities",
status.getMediaEntities())
                .append("extendedMediaEntities",
status.getExtendedMediaEntities())
                */
                ;
                this.collection.insertOne(document);
            }

}

```

Apêndice B

Listas de *Stop Words* e N-gramas

Na etapa de pré-processamento do conjunto de dados (*tweets*) temos as *stop words* definidas a partir do vocabulário da linguagem e embasada na literatura na área. Alguns n-gramas precisam ser transformados para que seu valor não seja comprometido durante o processo de classificação.

B.1 Lista de *Stop Words*

A lista de *stop words*, na tabela abaixo em inglês e português, contém basicamente pronomes, artigos, e palavras de pouco, ou nenhum, impacto na polaridade final de uma sentença textual. Essas listas são passadas ao software Weka para a execução do processo de classificação.

Inglês	Português
I, me	eu, me, mim
you, u, yo, ya	voce, vc, c, tu, te, ti, lhe
he, him, she, her, it	ele, ela
we, us	nos, nois, a gente, vos, vois
they, them	eles, elas
my, mine	meu, minha
your, yours. ur, urs	teu, tua, teus, tuas seu, sua, seus, suas vosso, vossa
his, hers, its	dele, dela
our, ours	nosso, nossa nossos, nossas da gente vossos, vossas
their, theirs	deles, delas
the, a, an	o, a, os, as

<p>this, that, dat, these, those</p>	<p>isso, esse, essa isto, este, esta aquilo, aquele, aquela esses, essas aqueles, aquelas</p>
<p>on, in, at into, onto</p>	<p>em, na, no, nas, nos nele, nela, neles, nelas nisso, nesse, nessa nesses, nessas nestes, nestas nisto, neste, nesta naquilo, naquele, naquela</p>
<p>from, of to, for by with</p>	<p>de, do, da dos, das para, pro, pra pros, pras ao, aos por com</p>
<p>as like as such</p>	<p>assim como tal</p>
<p>where what, wut, wat, which who, whose, whom when why, y because, cause, cos, cuz</p>	<p>onde, aonde que, q, oq, qual, quais quem quando, qdo, qd pq, porque</p>

B.2 Lista de *n-gramas* transformados em unigramas e unigramas modificados

N-gramas como nomes de cidades, séries de TV, episódios de séries de TV, títulos de músicas, videogames, nomes de pessoas, e etc, são transformados em unigramas. Alguns unigramas encontrados nos *tweets* correspondem a *N-gramas* diversos, como pode ser constatado na tabela abaixo, portanto todos devem ser transformados em um unigrama único.

n-grama ou 1-grama original	1-grama resultante
the wire	thewire
bb	breakingbad
breaking bad	breakingbad
tbbt	bigbangtheory
big bang theory	bigbangtheory
super girl	supergirl
scream queens	screamqueens
grey's anatomy greys anatomy	greysanatomy
flash	theflash
ahs american horror story	americanhorrorstory
tvd vampire diaries	vampirediaries
house	houseserie
lost	lostserie
glee	gleeserie
eureka	eurekaserie
pb prison break	prisonbreak
person of interest	personofinterest
hannibal	hannibalserie
empire	empireserie
blacklist	theblacklist
gothan	gothanserie
game of thrones got	gameofthrones
blue bloods	bluebloods
scandall	scandallsSerie
sons of anarchy	sonsofanarchy
blue mountain state	bluemountainstate
weeds	weedsSerie
skins	skinsSerie

swamp people	swamppeople
rick morty rick and morty rick n morty	rickmorty
once upon a time ouat	onceuponatime
pretty little liars pretty lil liars ppl	prettylittleliars
arrow	arrowserie
falling skies	fallingskies
downtown Abbey	downtownabbey
revenge	revengeserie
batesmotel	batesmotel
park avenue	parkavenue
heroes	heroesserie
the wolf among us	thewolfamongus
orange is the new black oitnb	orangeisthenewblack
my mad fat diary mmfd	mymadfatdiary
one three hill	onethrrehill
the originals	theoriginals
shameless	shamelessserie
scrubs	scrubserie
the office	theoffice
doctor who dw	doctorwho
friends	friendsserie
burn notice	burnnotice
davinci's demons davincis demons	davincisdemons
twolf teen wolf	teenwolf
new girl	newgirl
grimm	grimmserie

himyn how I met your mother	himym
agents of shield	agentsofshield
supernatural spn	supernaturalserie
gossipgirl	gossipgirl
black sails	blacksails
criminal minds	criminalminds
silicon valley	siliconvalley
z nation zn	znation
until down	untildown
death note	deathnote
fma fullmetal full metal alchemist	fma
flash rebirth	flashrebirth
civil war	civilwar
scysf so close yet so far	scysf
old a man logan	oldmanlogan
left for dead left 4 dead	leftfordead
borgias	Borgias
behind the scenes	behindthescenes
johnny depp	johnnydepp
penny dreadful	pennydreadful
la los angeles	losangeles
new York nyc	newyork
Hollywood studios horror nights	Hollywoodstudioshorrornights
universal studios	universalstudios
talking dead	talkingdead
the walking dead	twd
fear the walking dead fear twd	ftwd

night club casa noturna	nightclub casanoturna
reino unido	uk
arco e flecha	arcoflecha
nova iorque new york	nyc

Apêndice C

Listas de palavras polares

Palavras polares são utilizadas na identificação de emoção ou sentimento em um dado texto, podendo transmitir emoção de polaridade negativa ou positiva. Nas classificações prévias de subjetividade realizadas neste trabalho, as palavras em questão foram aplicadas, sendo a presença de alguma destas em um dado texto um indicativo de subjetividade deste e a ausência de qualquer palavra polar, um indicativo de objetividade do dito texto. A composição da lista de palavras polares foi possível através da observação e identificação manual de sentenças subjetivas no contexto dos dados trabalhados. Algumas palavras são expressas em sua forma integral, outras pelo seu radical, de forma a identificar todas as formas possíveis de uma dada palavra polar.

mean	best	meh, bleh	rather
prefer	better	bomb	finally
believe	great	savage	nw
think	good	luck	most, must
feel	bad	terrible	more, less
know	worst	ass	
enjoy	worse	dead	
recommend	like	crazy	
need	love	paranoi	
hope	hate	brilliant	
want	amaz(ing, azed)	addict	
miss	awesome	sweet	
doubt	cool	fascinat	
suck	nice	tired	
promis	fine	solid	
interest	well	drug	

Apêndice D

Resultados das classificações no software Weka

As saídas detalhadas do software Weka para todas as classificações realizadas nos experimentos comentados no documento.

D.1 Resultado da *10-fold cross-validation* com o classificador Naive Bayes Multinomial e conjunto em inglês

```
Correctly Classified Instances      4289          80.6203 %
Incorrectly Classified Instances    1031          19.3797 %
Kappa statistic                    0.6005
Mean absolute error                0.2004
Root mean squared error            0.4095
Relative absolute error            41.9722 %
Root relative squared error        83.8183 %
Total Number of Instances         5320

=== Detailed Accuracy By Class ===

           TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
           0.811   0.200   0.862     0.811   0.835     0.602   0.880    0.912    pos
           0.800   0.189   0.733     0.800   0.765     0.602   0.880    0.821    neg
Weighted Avg.   0.806   0.196   0.811     0.806   0.807     0.602   0.880    0.876

=== Confusion Matrix ===

  a    b  <-- classified as
2614  611 |  a = pos
 420 1675 |  b = neg
```

D.2 Resultado da *10-fold cross-validation* com o classificador Naive Bayes Multinomial e conjunto em português

```

Correctly Classified Instances      4208          79.0977 %
Incorrectly Classified Instances    1112          20.9023 %
Kappa statistic                    0.5695
Mean absolute error                 0.2169
Root mean squared error             0.4268
Relative absolute error             45.4381 %
Root relative squared error         87.3615 %
Total Number of Instances          5320

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                0.796   0.217   0.849      0.796   0.822      0.571    0.864    0.903    pos
                0.783   0.204   0.714      0.783   0.747      0.571    0.864    0.805    neg
Weighted Avg.   0.791   0.212   0.796      0.791   0.792      0.571    0.864    0.864

=== Confusion Matrix ===

  a    b  <-- classified as
2568  657 |  a = pos
 455 1640 |  b = neg

```

D.3 Resultado da *10-fold cross-validation* com o classificador SVM e conjunto em inglês

```

Correctly Classified Instances      4411          82.9135 %
Incorrectly Classified Instances    909           17.0865 %
Kappa statistic                    0.6363
Mean absolute error                 0.1709
Root mean squared error             0.4134
Relative absolute error             35.7868 %
Root relative squared error         84.602 %
Total Number of Instances          5320

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                0.889   0.263   0.839      0.889   0.863      0.638    0.813    0.813    pos
                0.737   0.111   0.812      0.737   0.773      0.638    0.813    0.702    neg
Weighted Avg.   0.829   0.203   0.828      0.829   0.827      0.638    0.813    0.769

=== Confusion Matrix ===

  a    b  <-- classified as
2867  358 |  a = pos
 551 1544 |  b = neg

```

D.4 Resultado da *10-fold cross-validation* com o classificador SVM e conjunto em português


```

Correctly Classified Instances      4365          82.0489 %
Incorrectly Classified Instances    955          17.9511 %
Kappa statistic                    0.6186
Mean absolute error                0.1795
Root mean squared error            0.4237
Relative absolute error            37.5978 %
Root relative squared error        86.7162 %
Total Number of Instances         5320

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.878   0.269   0.834     0.878   0.856     0.620   0.805    0.807    pos
                0.731   0.122   0.796     0.731   0.762     0.620   0.805    0.688    neg
Weighted Avg.   0.820   0.211   0.819     0.820   0.819     0.620   0.805    0.760

=== Confusion Matrix ===

  a    b  <-- classified as
2833 392 |  a = pos
 563 1532 |  b = neg

```

D.5 Resultado da classificação com o classificador NB Multinomial, subconjunto de treino em Inglês e subconjunto de teste traduzido PT-EN pelo Microsoft Bing Tradutor

```

Correctly Classified Instances      1197          75.0941 %
Incorrectly Classified Instances    397          24.9059 %
Kappa statistic                    0.4894
Mean absolute error                0.2546
Root mean squared error            0.4761
Relative absolute error            53.3287 %
Root relative squared error        97.4256 %
Total Number of Instances         1594

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.755   0.255   0.820     0.755   0.786     0.492   0.814    0.868    pos
                0.745   0.245   0.664     0.745   0.702     0.492   0.814    0.719    neg
Weighted Avg.   0.751   0.251   0.758     0.751   0.753     0.492   0.814    0.810

=== Confusion Matrix ===

  a    b  <-- classified as
 729 237 |  a = pos
 160 468 |  b = neg

```

D.6 Resultado da classificação com o classificador SVM, subconjunto de treino em Inglês e subconjunto de teste traduzido PT-EN pelo Microsoft Bing Tradutor

```
Correctly Classified Instances      1246          78.1681 %
Incorrectly Classified Instances    348          21.8319 %
Kappa statistic                    0.5371
Mean absolute error                0.2183
Root mean squared error            0.4672
Relative absolute error            45.7214 %
Root relative squared error        95.6237 %
Total Number of Instances         1594

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
          0.843   0.312   0.806     0.843   0.824     0.538   0.765    0.774    pos
          0.688   0.157   0.740     0.688   0.713     0.538   0.765    0.632    neg
Weighted Avg.   0.782   0.251   0.780     0.782   0.780     0.538   0.765    0.718

=== Confusion Matrix ===

  a  b  <-- classified as
814 152 |  a = pos
196 432 |  b = neg
```

D.7 Resultado da classificação com o classificador NB Multinomial, subconjunto de treino em Inglês e subconjunto de teste traduzido PT-EN pelo Google Tradutor

Correctly Classified Instances	1208	75.7842 %							
Incorrectly Classified Instances	386	24.2158 %							
Kappa statistic	0.5031								
Mean absolute error	0.2526								
Root mean squared error	0.4721								
Relative absolute error	52.9097 %								
Root relative squared error	96.6166 %								
Total Number of Instances	1594								
=== Detailed Accuracy By Class ===									
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.762	0.248	0.825	0.762	0.792	0.505	0.817	0.870	pos
	0.752	0.238	0.672	0.752	0.710	0.505	0.817	0.722	neg
Weighted Avg.	0.758	0.244	0.765	0.758	0.760	0.505	0.817	0.812	
=== Confusion Matrix ===									
a	b	<-- classified as							
736	230	a = pos							
156	472	b = neg							

D.8 Resultado da classificação com o classificador SVM, subconjunto de treino em Inglês e subconjunto de teste traduzido PT-EN pelo Google Tradutor

Correctly Classified Instances	1241	77.8545 %							
Incorrectly Classified Instances	353	22.1455 %							
Kappa statistic	0.5317								
Mean absolute error	0.2215								
Root mean squared error	0.4706								
Relative absolute error	46.3783 %								
Root relative squared error	96.3082 %								
Total Number of Instances	1594								
=== Detailed Accuracy By Class ===									
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.835	0.309	0.806	0.835	0.821	0.532	0.763	0.773	pos
	0.691	0.165	0.732	0.691	0.711	0.532	0.763	0.627	neg
Weighted Avg.	0.779	0.252	0.777	0.779	0.777	0.532	0.763	0.716	
=== Confusion Matrix ===									
a	b	<-- classified as							
807	159	a = pos							
194	434	b = neg							

D.9 Resultado da classificação com o classificador NB Multinomial, subconjunto de treino em Português e

subconjunto de teste traduzido EN-PT pelo Microsoft Bing Tradutor

Correctly Classified Instances	1164	73.0238 %							
Incorrectly Classified Instances	430	26.9762 %							
Kappa statistic	0.4438								
Mean absolute error	0.2685								
Root mean squared error	0.4876								
Relative absolute error	56.2262 %								
Root relative squared error	99.7896 %								
Total Number of Instances	1594								
=== Detailed Accuracy By Class ===									
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.748	0.298	0.795	0.748	0.771	0.445	0.794	0.840	pos
	0.702	0.252	0.645	0.702	0.672	0.445	0.794	0.700	neg
Weighted Avg.	0.730	0.280	0.736	0.730	0.732	0.445	0.794	0.785	
=== Confusion Matrix ===									
a	b	<-- classified as							
723	243	a = pos							
187	441	b = neg							

D.10 Resultado da classificação com o classificador SVM, subconjunto de treino em Português e subconjunto de teste traduzido EN-PT pelo Microsoft Bing Tradutor

Correctly Classified Instances	1230	77.1644 %							
Incorrectly Classified Instances	364	22.8356 %							
Kappa statistic	0.5131								
Mean absolute error	0.2284								
Root mean squared error	0.4779								
Relative absolute error	47.8235 %								
Root relative squared error	97.7972 %								
Total Number of Instances	1594								
=== Detailed Accuracy By Class ===									
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.845	0.341	0.792	0.845	0.818	0.515	0.752	0.763	pos
	0.659	0.155	0.734	0.659	0.695	0.515	0.752	0.618	neg
Weighted Avg.	0.772	0.268	0.769	0.772	0.769	0.515	0.752	0.706	
=== Confusion Matrix ===									
a	b	<-- classified as							
816	150	a = pos							
214	414	b = neg							

D.11 Resultado da classificação com o classificador NB Multinomial, subconjunto de treino em Português e subconjunto de teste traduzido EN-PT pelo Google Tradutor

```

Correctly Classified Instances      1164          73.0238 %
Incorrectly Classified Instances    430          26.9762 %
Kappa statistic                    0.4428
Mean absolute error                0.2721
Root mean squared error            0.4921
Relative absolute error            56.983 %
Root relative squared error        100.7203 %
Total Number of Instances          1594

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                0.752   0.303   0.793     0.752   0.772     0.444   0.792    0.838    pos
                0.697   0.248   0.646     0.697   0.671     0.444   0.792    0.708    neg
Weighted Avg.   0.730   0.281   0.735     0.730   0.732     0.444   0.792    0.787

=== Confusion Matrix ===

  a  b  <-- classified as
726 240 |  a = pos
190 438 |  b = neg

```

D.12 Resultado da classificação com o classificador SVM, subconjunto de treino em Português e subconjunto de teste traduzido PT-EN pelo Google Tradutor

```

Correctly Classified Instances      1229          77.1016 %
Incorrectly Classified Instances    365          22.8984 %
Kappa statistic                    0.5088
Mean absolute error                0.229
Root mean squared error            0.4785
Relative absolute error            47.9549 %
Root relative squared error        97.9315 %
Total Number of Instances          1594

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                0.855   0.358   0.786     0.855   0.819     0.512   0.748    0.760    pos
                0.642   0.145   0.742     0.642   0.688     0.512   0.748    0.617    neg
Weighted Avg.   0.771   0.274   0.769     0.771   0.768     0.512   0.748    0.704

=== Confusion Matrix ===

  a  b  <-- classified as
826 140 |  a = pos
225 403 |  b = neg

```

