**UNIVERSIDADE REGIONAL DE BLUMENAU**

**CENTRO DE CIÊNCIAS EXATAS E NATURAIS**

**CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO**

# BRAND FEELING: UM SISTEMA PARA ANALISAR O SENTIMENTO DOS USUÁRIOS EM RELAÇÃO A UMA MARCA

**LUIZ CÉSAR COPPI JUNIOR**

**BLUMENAU**
**2017**

**LUIZ CÉSAR COPPI JUNIOR**

# BRAND FEELING: UM SISTEMA PARA ANALISAR O SENTIMENTO DOS USUÁRIOS EM RELAÇÃO A UMA MARCA

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Profª. Andreza Sartori, Doutora - Orientadora

**BLUMENAU**
**2017**

# BRAND FEELING: UM SISTEMA PARA ANALISAR O SENTIMENTO DOS USUÁRIOS EM RELAÇÃO A UMA MARCA

Por

**LUIZ CÉSAR COPPI JUNIOR**

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

| | |
|---|---|
| Presidente: | Profª. Andreza Sartori, Doutora – Orientadora, FURB |
| Membro: | Profª. Joyce Martins, Mestre – FURB |
| Membro: | Prof. Roberto Heinzle, Doutor – FURB |

Blumenau, 27 de Outubro de 2017

## AGRADECIMENTOS

Agradeço primeiramente a minha família, que sempre me apoiou e incentivou meus estudos.

Agradeço especialmente meu pai por sempre me buscar durante o curso, não deixando o cansaço da volta para casa prejudicar meus estudos.

À minha orientadora Andreza, que me guiou desde o início da monografia, se dedicando para me auxiliar nas questões textuais e nas tecnologias usadas.

Aos meus colegas de curso, que deixaram esta jornada menos árdua e mais agradável.

E a todos os professores que me auxiliaram durante o curso, compartilhando o seu conhecimento.

# RESUMO

Este trabalho apresenta o desenvolvimento de um sistema para a análise de sentimento dos usuários do Twitter sobre uma determinada marca. A partir de uma interface onde o usuário informa uma marca é possível visualizar o sentimento dos usuários do Twitter sobre a mesma. Ao ser informada a marca, o sistema realiza o processo de busca e análise dos dados, que se inicia com a mineração dos dados provenientes da busca avançada do Twitter. Então, estes dados são filtrados por meio de técnicas para isolar o texto de cada postagem e do uso da biblioteca Stanford Log-linear Part-Of-Speech (POS) Tagger para definir as classes gramaticais de cada termo. Na sequência, é feita a análise de sentimentos sobre estes termos utilizando para isso a biblioteca MIT Java WordNet Interface (JWI) em conjunto com o WordNet para reduzir cada termo ao seu radical, e o SentiWordNet para verificar qual o valor de sentimento que cada termo possui. Durante o desenvolvimento deste sistema foi observada a necessidade de adicionar um filtro de categoria para as buscas, visando suprir uma limitação existente no Twitter, mas ao mesmo tempo aumentando o número de termos em cada busca. Como resultado, o sistema se mostrou capaz de analisar uma grande quantidade de dados provenientes do Twitter, em um tempo de execução satisfatório para o usuário final. O trabalho proposto obteve sucesso em atingir os objetivos elencados, adaptando-se as limitações encontradas ao longo do seu desenvolvimento. Este trabalho se mostra relevante pois contribui tanto para a área de análise de sentimentos, ao disponibilizar uma aplicação que evidencia o potencial desta área para o público em geral, bem como no âmbito social, através de uma ferramenta acessível para o usuário comum.

Palavras-chave: Análise de sentimentos. Mineração de dados. Filtragem de dados. WordNet. SentiWordNet.

# ABSTRACT

This work presents the development of a system to analyze the sentiment of Twitter's users about a particular brand. This system provides an interface where the users can inform a brand and visualize the information related to the corresponding sentiment. When a brand is informed to the system, begins the process of data searching and analysis, starting with the data mining of the advanced search of Twitter. These data are then filtered through techniques to isolate the text of each post, and the application of the Stanford Log-linear Part-Of-Speech (POS) Tagger library, to define the grammatical classes of each term. Afterwards, the MIT Java WordNet Interface (JWI) library along with WordNet is used in order to reduce each term to its radical, as well as the SentiWordNet to verify the value of each term's sentiment. During the development of this work, a category filter to the search was added, aiming to solve an existing limitation on Twitter and to increase the number of terms in each search. As a result, the system was able to analyze a large amount of data coming from Twitter at a satisfactory execution time to the end user. This work has successfully achieved the defined aims, as well as was able to adapt to the limitations encountered throughout the development. This work contribute on the sentiment analysis area throught an application that highlights the potential of this area to the general public, as well as, providing an accessible tool for the regular user.

Key-words: Sentiment analysis. Data mining. Data filtering. WordNet. SentiWordNet.

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

HTML – HyperText Markup Language

JWI – MIT Java WordNet Interface

PLN – Processamento de Linguagem Natural

POS – Part Of Speech

RMI – Remote Method Invocation

SOAP – Protocolo Simples de Acesso a Objetos

SVM – Support Vector Machine (Máquina de Vetores de Suporte)

UML – Unified Modeling Language

URL – Localizador Uniforme de Recursos

# SUMÁRIO

# 1 INTRODUÇÃO

Atualmente vive-se numa sociedade na qual cerca de 4.2 bilhões de pessoas estão conectadas as redes sociais, sendo que estas possuem um grande impacto na opinião e, consequentemente, nas decisões que as pessoas tomam em suas vidas (MANGUKIYA, 2016). Em paralelo a isso, empresas buscam novos métodos para se aproximar dos usuários bem como aumentar seu lucro, “a opinião on-line tem se tornado um tipo de moeda virtual que pode fazer um produto falhar ou ganhar o mercado” (WRIGHT, 2009, tradução nossa, p. 1).

A partir desta problemática, surgiram soluções para tornar essa massa de dados disponível nas redes sociais em algo útil. Dentre elas, a abordagem que se destacou foi a mineração de dados. Segundo Memon et al. (2010, p. 5, tradução nossa), “o impacto desta nova área de mineração de dados em redes sociais como papel para suportar a manutenção do conhecimento e tomada de decisão [...] junto do seu impacto em usuários, organizações e na sociedade ainda deve ser encontrado”.

Com isso, uma nova abordagem para análise desses dados vem ganhando força, a análise de sentimentos, que é um campo de pesquisa que possui como característica principal a extração do sentimento expresso em uma frase, classificando as sentenças como positivas, negativas ou neutras. Apesar de parecer simples, esta área trabalha com o Processamento de Linguagem Natural (PLN), o que acaba envolvendo problemas com figuras de linguagem, como sarcasmo e ironia, por exemplo (FERSINI et al., 2016, p. 5-8).

Diversos estudos surgiram para descobrir qual a melhor combinação destas áreas, usando os algoritmos para análise de sentimento em conjunto com a mineração de dados de redes sociais. “A explosão de dados disponíveis na web tem tornado a pesquisa e catalogação automática de textos cada vez mais interessante, assim como a extração de informações e de sentimentos” (FORNACCIARI; MORDONINI; TOMAUIOLO, 2015, p. 2-3, tradução nossa). Porém aplicações que utilizam da combinação de análise de sentimentos e mineração de dados para validar a opinião do público sobre uma marca são somente encontradas de forma paga, e num escopo menor em ferramentas de inteligência empresarial. Essas ferramentas são sistemas que possuem dados de uma determinada empresa, visando disponibilizar gráficos e informações para a gestão da mesma.

Diante deste cenário, foi desenvolvido um sistema capaz de analisar o sentimento do público em geral sobre uma determinada marca, permitindo ao usuário fornecer uma marca como entrada e visualizar o sentimento predominante sobre a mesma. A partir da informação de entrada, o sistema é capaz de efetuar a busca dos dados do Twitter, a filtragem destes

dados e, por fim, a análise de sentimentos sobre os mesmos, gerando como resultado os termos citados sobre esta marca e o sentimento do público sobre a mesma.

## 1.1 OBJETIVOS

Este trabalho tem como objetivo desenvolver um sistema para analisar automaticamente o sentimento dos usuários em relação a uma marca.

Os objetivos específicos são:

a) realizar uma pesquisa de opinião em tempo real com usuários no Twitter em sua versão em inglês sobre uma marca escolhida;
b) efetuar um filtro sobre os dados vindos do Twitter para deixar apenas informações válidas para o algoritmo de análise de sentimentos;
c) aplicar um algoritmo de análise de sentimentos visando medir automaticamente a opinião relativa ao sentimento positivo ou negativo dos usuários em relação a marca pesquisada;
d) disponibilizar uma interface intuitiva para que o usuário possa realizar a pesquisa de opinião sobre uma marca.

## 1.2 ESTRUTURA

O presente trabalho está organizado em quatro capítulos: introdução, fundamentação teórica, desenvolvimento e conclusões. No segundo capítulo são apresentados os fundamentos básicos para a realização deste trabalho, assim como os trabalhos correlatos. O terceiro capítulo descreve o desenvolvimento do sistema, incluindo os requisitos propostos e diagramas, as técnicas, ferramentas e tecnologias utilizadas, bem como os principais trechos de código, as telas do sistema e a análise de resultados. Por fim, o quarto capítulo expõe as conclusões obtidas a partir dos resultados deste trabalho, assim como possíveis extensões para o mesmo.

# 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os principais assuntos necessários para realização do trabalho. Desta forma, este capítulo foi dividido em 5 partes, onde a seção 2.1 apresenta os conceitos de processamento de linguagem natural, seus componentes e áreas, bem como a filtragem de dados. A seção 2.2 conceitua a análise de sentimentos, sua definição e principais técnicas utilizadas. A seção 2.3 apresenta a mineração de dados e o seu processo para execução. Na seção 2.4 é apresentada uma explanação sobre redes sociais e a problematização da quantidade de dados gerados diariamente. Por fim, na seção 2.5 são descritos três trabalhos correlatos.

## 2.1 PROCESSAMENTO DE LINGUAGEM NATURAL

O PLN é uma área da inteligência artificial que trata da compreensão e interpretação de textos na linguagem natural humana pelas máquinas. Esta área correspondente ao desenvolvimento de modelos computacionais que devem realizar tarefas dependentes de entradas ou analisar textos que são expressos em alguma forma literal (PEREIRA, 2013).

De acordo com Müller (2003), a análise dos dados de entrada num sistema de PLN é geralmente dividida em quatro etapas, sendo elas: análise morfológica; análise sintática; análise semântica e análise pragmática. O processamento começa pela análise morfológica, que trata da redução de cada palavra ao seu radical. Esta etapa visa unificar o dicionário de palavras utilizado, agrupando-as por um radical comum.

Então, aplica-se a análise sintática dos dados, que analisa as palavras de uma frase visando encontrar a sua estrutura gramatical e obter como saída uma representação da frase que revela a relação entre as palavras (LIDDY, 2001). A análise semântica busca representar o significado exato da sentença de acordo com o dicionário, podendo, para isso, buscar tanto o significado de uma frase como um todo, como combinações de palavras definidas na análise anterior. Por fim, aplica-se a análise pragmática, que trata do uso objetivo da linguagem buscando obter o contexto existente em um texto, com o objetivo de interpretar o significado real do mesmo sem que este significado esteja descrito nele (LIDDY, 2001, p. 9).

Uma das técnicas que podem ser utilizadas na fase de análise sintática dos dados é a técnica de Part Of Speech Tagger (POS Tagger), onde as partes da linguagem se baseiam na ideia da divisão gramatical das palavras entre classes pré-definidas, como: substantivo, verbo, artigo, pronome, preposição, advérbio e conjunção. Nesta técnica é realizada uma análise de cada frase, onde cada palavra recebe um classificador, indicando qual a classe gramatical pertence (VOUTILAINEN, 2005).

A partir do PLN surgiram várias aplicações e até mesmo subáreas que trabalham com partes específicas dentro desta área, como por exemplo, diversas técnicas para filtragem de dados, que é detalhado na seção 2.1.1. Outros exemplos são as áreas que tratam da comunicação do computador com as pessoas, como a sumarização automática de um texto, e a tradução de textos entre diferentes linguagens. O PLN também é aplicado na área do reconhecimento de voz, onde é realizada a conversão do áudio para um formato textual e na área de análise de sentimentos, que é o foco deste trabalho e é detalhada na sequência.

### 2.1.1 FILTRAGEM DE DADOS

Ao trabalhar com o PLN para obter informações é necessário fazer uma filtragem sobre os dados originais. Santos (2010, p. 22) afirma que a etapa de filtragem dos dados "é mais onerosa, devido a quantidade de técnicas diferentes que podem ser aplicadas, sendo que não existe uma técnica considerada superior que as outras, pois cada uma se adequa melhor em determinado contexto". Dentre as várias técnicas existentes, são detalhadas as que são utilizadas neste trabalho, sendo elas: a derivação, a lematização, a tokenização, as StopWords e por fim, a remoção dos caracteres da Linguagem de Marcação de Hipertexto (HyperText Markup Language - HTML).

Um dos procedimentos que pode ser utilizado é o da redução das palavras a sua forma original, isto significa remover as conjugações das palavras visando obter o radical das mesmas. Para isso, existem duas técnicas: derivação e lematização. A derivação é utilizada na extração de informações e tem como característica a redução de uma palavra qualquer a sua menor forma, sendo que neste caso esta forma não respeita as POS da linguagem, podendo unir palavras de diferentes classes gramaticais (JURŠIC et al., 2010, p. 4).

A lematização se baseia também na redução da palavra a sua menor forma, porém neste caso são respeitadas as classes gramaticais das palavras, ou seja, as palavras são agrupadas por cada POS, gerando assim os chamados "*lemas*" de uma palavra base em cada classe (JURŠIC et al., 2010, p. 4). Na Figura 1 é mostrado um exemplo de lematização, separando as palavras base de sufixos na língua inglesa, onde na primeira palavra é retirado o número e na quinta palavra é alterado o tempo verbal de passado para presente.

Figura 1 – Exemplo de lematização de palavras

| | Wordform | Lemma | (WordformSuffix --> LemmaSuffix) |
|---|---|---|---|
| English | | | |
| 1 | dogs | dog | ("s" --> " ") |
| 2 | wolves | wolf | ("ves" --> "f") |
| 3 | sheep | sheep | (" " --> " ") |
| 4 | looking | look | ("ing" --> " ") |
| 5 | took | take | ("ook" --> "ake") |

Fonte: Juršic (2010).

A escolha entre as técnicas se faz com base em quais linguagens serão tratadas na aplicação, e também baseado no aproveitamento das POS na aplicação. Neste trabalho foi utilizada a lematização, pois a POS é importante para realizar a análise dos dados na aplicação. Outra técnica utilizada é a tokenização, que consiste na "separação de um texto em suas unidades mínimas, ou seja, cada palavra presente no texto é separada, bem como a pontuação" (SANTOS, 2010, p. 23).

A técnica de StopWords corresponde a palavras que são utilizadas em demasia numa linguagem e acabam perdendo seu significado para a aplicação, assim como palavras que apenas prejudicam a precisão da aplicação, como números ou pronomes, por exemplo. Para tratar estas palavras é criada uma lista, chamada de StopList, onde são indicadas as palavras a serem ignoradas pela aplicação. Na Figura 2 é mostrada a lista de StopWords que devem ser utilizadas ao trabalhar com a língua inglesa segundo Schütze (2008).

Figura 2 – Lista de StopWords ao trabalhar com a língua inglesa

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| a | an | and | are | as | at | be | by | for | from |
| has | he | in | is | it | its | of | on | that | the |
| to | was | were | will | with | | | | | |

Fonte: Schütze (2008).

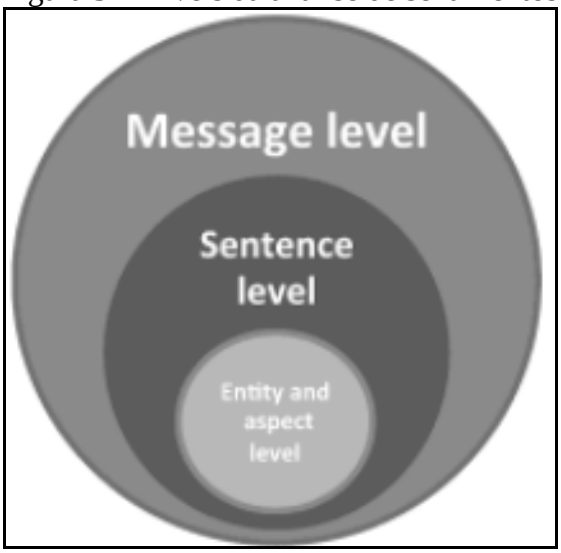
Quando é realizada a mineração de dados da web os dados são extraídos no formato da linguagem HTML, ou seja, junto dos dados existe uma marcação responsável por formatá-los na visualização do site. Por este motivo é necessário realizar um tratamento dos dados, visando remover todos os marcadores HTML para isolar apenas os dados do processamento, evitando assim, a interferência de ruídos na análise dos dados. Também é necessário remover links contidos nos dados, já que apenas o texto é utilizado na análise posterior, assim como tratar caracteres específicos que podem estar dentro do texto, como, por exemplo, a sequência de caracteres utilizada para a quebra de uma linha.

## 2.2 ANÁLISE DE SENTIMENTOS

A análise de sentimentos, também chamada de mineração de opinião, tem como característica principal a extração dos sentimentos contidos em um texto. Segundo Fersini et al. (2016, p. 1, tradução nossa), a análise de sentimentos é descrita como a definição de "ferramentas automáticas que conseguem extrair informações subjetivas de textos em linguagem natural [...] visando criar conhecimento estruturado e que possa ser usado por um sistema de decisão".

Fersini et al. (2016) descreve que a análise de sentimentos pode trabalhar em três níveis, conforme apresentado na Figura 3. No nível de mensagem (*message level*), a análise é realizada com base em cada um dos textos, analisando o texto como um todo. No nível de sentença (*sentence level*), o texto é dividido e a análise é realizada sobre cada uma de suas frases. Por fim, o nível de entidade e aspecto (*entity and aspect level*) trabalha com o conceito de que uma emoção é sempre sobre uma entidade ou aspecto específico, ou seja, apenas sobre as palavras chaves de um texto.

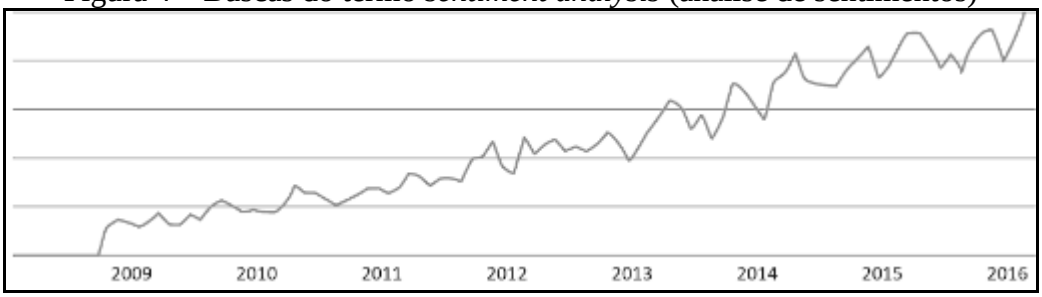
Figura 3 – Níveis da análise de sentimentos




Fonte: Fersini et al (2016).

O gráfico mostrado na Figura 4 retrata as buscas do termo *sentiment analysis* (análise de sentimentos) no Google. A partir dele pode-se visualizar que o interesse na área vem aumentando consideravelmente. Como consequência, diversas abordagens para a utilização da análise de sentimentos surgiram. As principais são a polaridade, a identificação e classificação da objetividade, subjetividade ou neutralidade de um texto, e o método de análise de características dos textos, que são aplicadas neste trabalho.

Figura 4 – Buscas do termo *sentiment analysis* (análise de sentimentos)




Fonte: Fersini et al (2016).

A polaridade busca representar se um sentimento é positivo, negativo ou neutro. O sentimento positivo é representado por todas as emoções positivas, como alegria, entusiasmo,

esperança, isto é, termos que expressam positividade. O sentimento negativo é representado por todas as emoções negativas como tristeza, raiva e medo (MOHAMMAD, 2015). Segundo Benevenuto, Ribeiro e Araújo (2015, p. 3), “Alguns métodos tratam a polaridade como um resultado discreto binário (positivo ou negativo) ou ternário (positivo, negativo ou neutro)”. Algumas abordagens se baseiam na intensidade de um sentimento, onde é estimado, com base numa variação máxima pré-definida, a intensidade de um determinado sentimento no texto em relação aos demais sentimentos, sendo aplicado tanto para sentimentos positivos quanto para negativos.

A classificação do texto entre objetivo ou subjetivo, é outra abordagem da análise de sentimentos. Uma vez identificado que um texto é objetivo não é necessário realizar nenhum tratamento sobre o mesmo, já que neste caso o texto não possui opinião e deve ser classificado como neutro. O texto subjetivo geralmente representa sentenças que buscam expressar opinião sobre algo, na qual é necessário aplicar a polaridade, isto é positivo ou negativo, para que seja possível identificar o sentimento do mesmo (FERSINI et al., 2016, p. 5-6).

Já o método de análise de características dos textos, chamado de *Feature Selection*, busca diminuir o número de características analisadas e aumentar a eficiência da análise. Para isso, é realizado um ranqueamento de todas as características disponíveis com base em quanto cada característica contribui para análise. A partir deste ranking é definido uma pontuação de corte, onde as características abaixo desta pontuação são removidas da análise (NICHOLLS; SONG, 2010, p. 287).

A análise de sentimentos trabalha com o PLN e, consequentemente, os problemas que afetam esta área também afetam a análise de sentimentos. Um exemplo é a interpretação de sarcasmo e ironia, na qual o sentido do conteúdo da frase ou texto é o contrário do seu real significado. No âmbito da análise de sentimentos isto quer dizer que quando uma sentença usa uma destas figuras de linguagem é classificada como emocionalmente positiva por um algoritmo enquanto na verdade é negativa, e vice-versa.

## 2.3 MINERAÇÃO DE DADOS

A mineração de dados corresponde ao processo de identificar padrões ou de extrair informações de uma grande massa de dados que pode ter como origem uma base de dados, a web ou qualquer outro repositório de dados (HAN; PEI; KAMBER, 2011). Segundo Witten et al. (2016, p. 6, tradução nossa), “Mineração de dados é definida como o processo de descobrir padrões em dados. [...] Os padrões descobertos devem ser significativos e levar para alguma vantagem”.

O processo por trás da mineração de dados geralmente acontece através de uma sequência de passos. Primeiro os dados são limpos, visando remover ruídos e dados inconsistentes, seguido da integração dos dados para os casos de mineração de várias fontes. Então, são realizadas a seleção dos dados relevantes, para a finalidade da aplicação, e a transformação desses dados em tabelas ou estruturas, onde são aplicados métodos e algoritmos para extrair os dados para a aplicação com o intuito de facilitar a mineração. Por fim, são retiradas destes dados as informações necessárias, que são tratadas e apresentadas ao usuário final da aplicação (HAN; PEI; KAMBER, 2011, p. 7-8).

A mineração de dados pode ser executada sobre diversas fontes, onde a mais comum é através de banco de dados. Os bancos de dados podem ser estruturados de duas maneiras: relacional, sendo que neste caso os dados são estruturados em tabelas; não relacional, onde os dados não são estruturados em tabelas, mas com diferentes técnicas, como arquivos por exemplo. Ao efetuar a mineração dos bancos de dados é possível realizar consultas específicas para analisar estes dados, buscar padrões e até prever dados futuros. Também é possível aplicar as técnicas de mineração de dados em dados ordenados, dados estruturados em grafos, dados de *network* ou até dados guardados em arquivos de texto (HAN; PEI; KAMBER, 2011).

A mineração de dados também pode ser pesquisada da web, através dos marcadores HTML existentes nas páginas. Uma vez identificados quais marcadores possuem as informações que são necessárias para a aplicação, a mineração irá então retirar os dados contidos nestes (HAN; PEI; KAMBER, 2011, p. 7-8). Também é uma prática comum na mineração de dados da web utilizar uma Interface de Programação de Aplicativos (Application Programming Interface - API), que é um “artefato de acesso por meio de linguagens de programação” (GOLDSCHMIDT; PASSOS, 2015). Este serviço retorna para as aplicações que acessam o mesmo informações pré-definidas pelo site de origem, ou seja, apenas dados que os autores desejam compartilhar com terceiros.

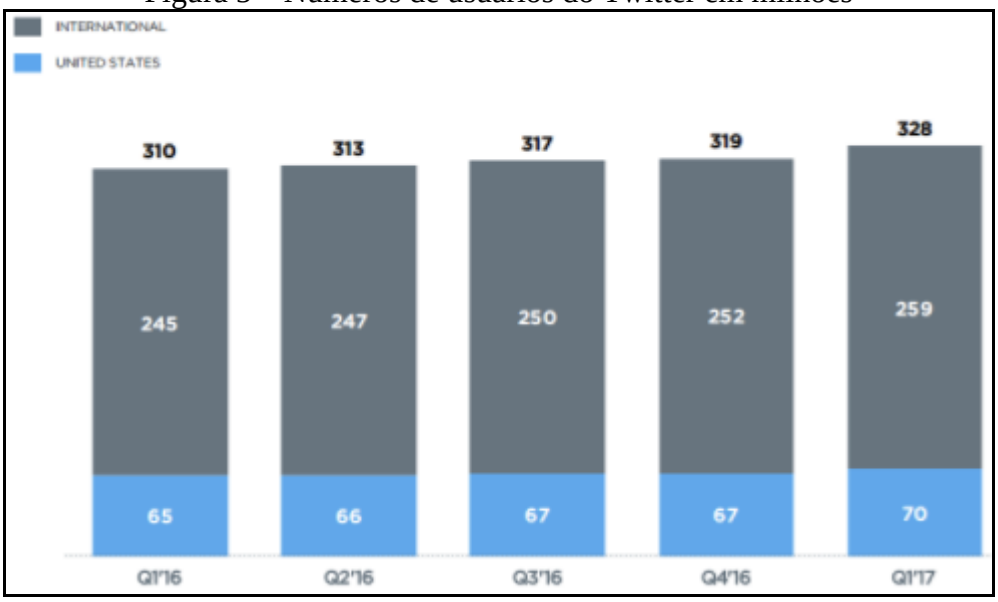
## 2.4 REDE SOCIAL

Uma rede social é uma plataforma on-line que permite aos seus usuários se conectarem a outras pessoas, com base em suas amizades, interesses profissionais, atividades dentre outros. Estas redes geralmente fornecem uma forma para que estes usuários se comuniquem, seja por meio de mensagens privadas ou publicações e comunicados no geral. Nos últimos anos, esta área “tem explodido como uma categoria de comunicação on-line onde as pessoas

criam conteúdo, compartilham, marcam e se interligam num ritmo extraordinário" (ASUR; HUBERMAN, 2010, p. 1, tradução nossa).

Algumas redes sociais funcionam em formato de micro blogs, onde os usuários se comunicam através de mensagens curtas. Estes sites se tornaram uma fonte de informação variada, por permitir aos usuários, que são de diferentes países, classes sociais e interesses distintos, expressarem rapidamente suas opiniões sobre temas variados incluindo assuntos relacionados ao seu dia-a-dia (AGARWAL et al., 2011). O Twitter[1] se destaca dentre estas redes sociais como a que gera mais conteúdo, por conta do grande número de usuários ativos, conforme demonstrado na Figura 5. A Figura 5 destaca o crescimento do número de usuários do Twitter ao longo do ano de 2016 nos Estados Unidos e mundialmente. Considerando uma divisão de 4 trimestres, o número de usuário parte de 310 milhões no primeiro trimestre de 2016 e chega a 328 milhões no começo de 2017.

Figura 5 – Números de usuários do Twitter em milhões




Fonte: Hutchinson (2017).

A análise de sentimentos tradicional acontece sobre uma grande base de dados, textos ou informações de documentos no geral, onde estas informações utilizam uma linguagem formal. No caso das redes sociais, especialmente micro blogs, o conteúdo gerado representa a opinião do público em geral sobre diversos assuntos, e na maioria dos casos estas opiniões são expressas numa linguagem informal, necessitando de tratamento para a análise de sentimentos (GO; BHAYANI; HUANG, 2009, p. 1).

[1] O Twitter pode ser acessado no site https://twitter.com/

## 2.5 TRABALHOS CORRELATOS

Nesta seção são apresentados três trabalhos correlatos a este trabalho. Na seção 2.1 é destacado o artigo científico Sentiment strength detection for the social web (THELWALL; BUCKLEY; PALTOGLOU, 2012). A seção 2.2 descreve o trabalho Protótipo para mineração de opinião em redes sociais: estudo de casos selecionados usando o Twitter (SANTOS, 2010). Por fim, na seção 2.3 é apresentado o artigo Twitter, MySpace, Digg: Unsupervised Sentiment Analysis in Social Media (PALTTOGLOU; THELWALL, 2012).

### 2.5.1 SENTIMENT STRENGTH DETECTION FOR THE SOCIAL WEB

O trabalho apresentado por Thelwall, Buckley e Paltoglou (2012) tem como objetivo verificar se o algoritmo SentiStrength, que usa uma abordagem não supervisionada de aprendizagem de máquina, é uma opção viável para análise de sentimentos em redes sociais. Uma comparação de desempenho com outros algoritmos para análise de sentimentos que utilizam técnicas de aprendizado de máquina é realizada, a fim de validar se o SentiStrength tem um resultado mais preciso.

Para fazer a comparação, foram desenvolvidas aplicações utilizando o algoritmo SentiStrength juntamente com outros algoritmos. Os algoritmos utilizados para comparação são: regressão logística (SLOG), que corresponde ao algoritmo de regressão apropriado para quando a variável utilizada é binária, ou seja, possui apenas duas variações; e o *ADA Bost*, que, segundo Schapire (2013, p. 1, tradução nossa) “uma abordagem de aprendizado de máquina baseada na ideia de criar uma regra de predição muito precisa combinando diversas regras relativamente fracas e não tão precisas”.

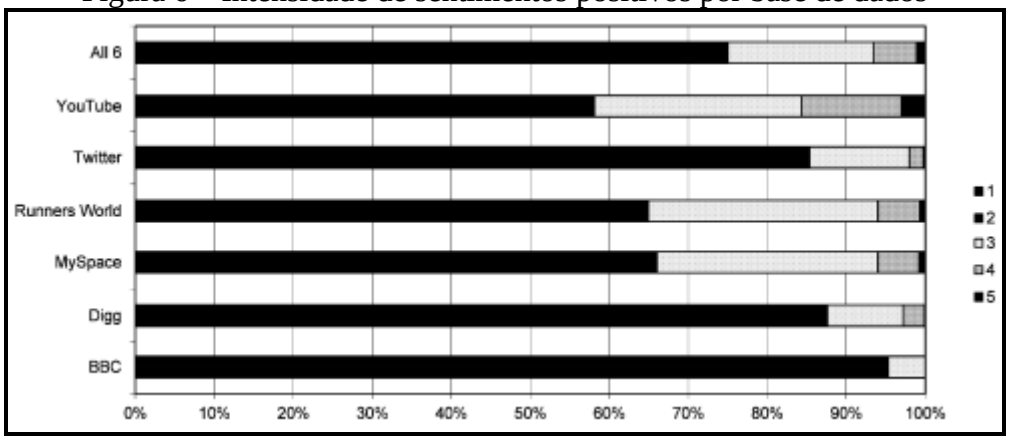
Estas aplicações são então executadas sobre os dados oriundos dos fóruns da BBC, Digg.com e Runners World, além dos comentários do Youtube, Myspace e Twitter. A Tabela 1 apresenta a quantidade média de caracteres (*mean chars*), palavras (*mean words*), e textos (*texts*) existentes em cada base de dados. A Figura 6 e a Figura 7 mostram que estas bases possuem concentrações de sentimentos positivos e negativos diferentes, sendo que o índice (de 1 à 5) indica a porcentagem de intensidade do sentimento.

Tabela 1 – Comparativo das bases de dados utilizadas

| | Mean chars | Mean words | Texts |
|---|---|---|---|
| BBC | 356.44 | 62.54 | 1000 |
| Digg | 183.32 | 31.49 | 1077 |
| MySpace | 101.91 | 20.08 | 1041 |
| Runners World | 335.42 | 65.13 | 1046 |
| Twitter | 94.55 | 15.35 | 4218 |
| Youtube | 91.18 | 17.12 | 3407 |
| All Six Combined | 146.05 | 26.18 | 11790 |

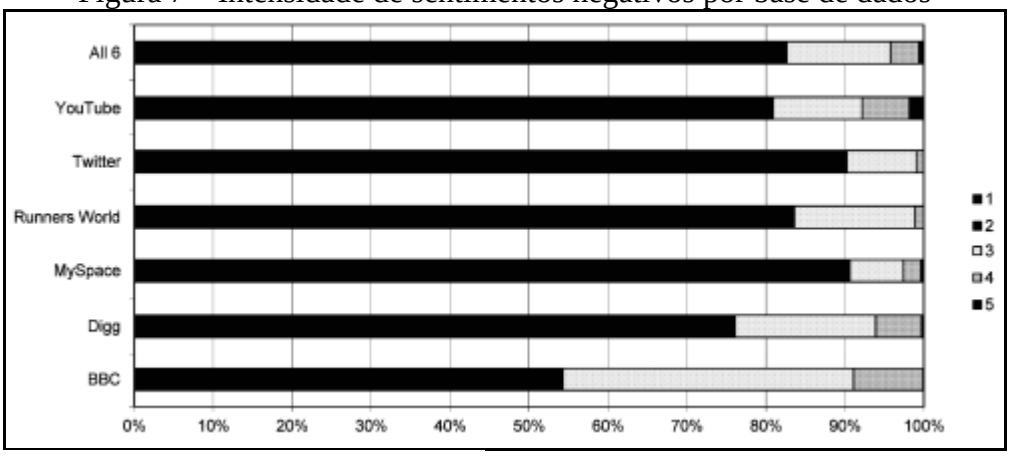
Fonte: Thelwall, Buckley e Paltoglou (2012).

Figura 6 – Intensidade de sentimentos positivos por base de dados




Fonte: Thelwall, Buckley e Paltoglou (2012).

Figura 7 – Intensidade de sentimentos negativos por base de dados




Fonte: Thelwall, Buckley e Paltoglou (2012).

Os resultados mostram que o algoritmo SentiStrength obteve correlação positiva em relação a performance dos demais algoritmos para as bases de dados utilizadas. Os autores citam que alguns dados acabaram sendo descartados por conta de ruídos, como propagandas ou links para outros sites. Observaram também que os algoritmos que utilizam técnicas como aprendizagem de máquina para refinar seus dados, acabaram tendo um desempenho melhor que o SentiStrength.

Por fim, os autores reportam que o SentiStrength pode ainda evoluir em ambientes onde é utilizada linguagem informal, assim como adicionar termos indiretos de afeto ao algoritmo. Estes resultados mostram também que algoritmos para análise de sentimentos ainda podem

melhorar ao trabalhar com redes sociais através da filtragem de dados, onde é possível extrair informações mais precisas desta fonte de dados informais.

### 2.5.2 PROTÓTIPO PARA MINERAÇÃO DE OPINIÃO EM REDES SOCIAIS: ESTUDO DE CASOS SELECIONADOS USANDO O TWITTER

O trabalho de Santos (2010) tem como objetivo desenvolver um protótipo para analisar a opinião das pessoas na web, utilizado para isto o Twitter como base das informações. Para isso, os dados foram obtidos através de mineração de dados e do algoritmo de Máquina de Vetores de Suporte (Support Vector Machine - SVM) para criar um protótipo que não só fizesse a análise desta fonte, mas também que possibilitasse a expansão do sistema no futuro.

O autor detalha a execução do trabalho através de um processo estruturado onde devem ser seguidos os passos mostrados na Figura 8. Primeiro é realizada a coleta de dados, onde foram retirados dados de arquivos textuais contidos no disco rígido do usuário, de tabelas em bancos de dados e da web. Para extrair os dados da web foi utilizado um *crawler*, que são robôs programados para percorrer a web visando coletar dados específicos.

Figura 8 – Etapas do processo de mineração de texto (*text mining*)


Fonte: Santos (2010).

Como segundo passo, o protótipo passa por etapas para tratar os dados disponíveis e prepará-los para a utilização na análise. No pré-processamento são utilizadas técnicas como a separação do texto em palavras, a normalização e a remoção de palavras específicas com o objetivo de limpar os dados utilizados. Antes da mineração, os dados são indexados pelas palavras contidas em cada texto, visando otimizar a performance da mesma. A etapa de mineração é realizada em três passos: primeiro são agrupados os textos previamente pesquisados em conjuntos de dados similares, chamados de *clusters*; segundo é feito a

sumarização, que busca resumir o conteúdo dos textos de forma automática; e, por fim, são extraídas as informações contidas nestes dados previamente minerados.

Na última etapa é executada a análise de sentimentos utilizando um algoritmo de aprendizado de máquina desenvolvido pelo autor. Este algoritmo efetua a análise sobre as informações geradas pelas etapas anteriores, separando estes dados em classes pré-definidas pelo autor. Como exemplo é citada a separação dos dados pela polaridade, onde o algoritmo deve dividir os textos entre sentimentos positivos, negativos e neutros.

A Figura 9 apresenta uma das etapas do protótipo desenvolvido pelo autor. Esta etapa visa mostrar a polaridade da opinião dos usuários sobre o termo, ou seja, mostrar o percentual de sentimentos positivos e negativos sobre o mesmo. Segundo Santos (2010, p. 82), "O protótipo desenvolvido demonstrou-se funcional, pois foi capaz de realizar o processo de análise de sentimentos requisitado pelo usuário de maneira razoável". Apesar de ser sólido na análise de textos, foi observado que o protótipo não obteve sucesso na obtenção dos dados do Twitter, por não ter considerado e retirado mensagens específicas como notícias, *retweets* e afins, que, por fim, acabaram atrapalhando a análise.

Figura 9 – Resultados por polaridade da opinião dos usuários sobre um termo


Fonte: Santos (2010).

### 2.5.3 TWITTER, MYSPACE, DIGG: UNSUPERVISED SENTIMENT ANALYSIS IN SOCIAL MEDIA

O trabalho de Paltoglou e Thelwall (2012) é focado nas diversas formas de comunicação via texto na web. Propõe uma abordagem intuitiva utilizando algoritmos não supervisionados, baseando-se num dicionário de palavras para estimar o nível de intensidade emocional contido em um texto, e com base neste nível retirar a polaridade (positiva ou

negativa) contida, caso ela exista. Para isto, são utilizadas três bases de dados nos testes, que foram escolhidas visando englobar as diferentes maneiras de comunicação via web.

A primeira base de dados utiliza o Twitter, que é dividida em duas seções: uma utiliza os dados provenientes da API disponibilizada pelo Twitter, enquanto a segunda seção contém dados extraídos via mineração de dados. A segunda base de dados é extraída do site de notícias Digg, onde a linguagem dos usuários é mais formal. A última é proveniente do site MySpace onde os dados são gerados através de conversas informais entre os usuários.

Para validar o desempenho do algoritmo é realizada uma comparação com outros algoritmos de análise de sentimentos existentes no mercado: o *Naive Bayes (NB)*, que corresponde a uma série de algoritmos supervisionados de aprendizado de máquina, que se baseiam na teoria de que todas as características são independentes; *Maximum Entropy* que é uma técnica usada para estimar as probabilidades das entradas de um algoritmo de modo que estas entradas não precisam ser conhecidas (PENFIELD, 2003, p. 1), e SVM que são técnicas utilizadas para analisar um conjunto de dados e reconhecer padrões. Cada um destes algoritmos, junto do algoritmo proposto pelos autores, é executado sobre cada uma das bases de dados previamente criadas, gerando assim um grande número de resultados para a validação da abordagem deste trabalho.

Na Tabela 2 e na Tabela 3 são descritos os dados da análise no site Digg. A Tabela 2 apresenta a classificação por polaridade, dividida entre palavras que possuem sentimentos positivos e negativos, já a Tabela 3 por objetividade e por subjetividade. Em ambas as tabelas são apresentados os resultados de desempenho da abordagem utilizando um dicionário de palavras, bem como os resultados de desempenho dos outros algoritmos citados anteriormente para a comparação. Para comparação são utilizados o cálculo da precisão do algoritmo baseado em categorias (Pr.), da fração de informações relevantes extraídas, ou *Recall* (R.) e o medidor de precisão sobre o *Recall* (F1). Os resultados deste trabalho, conforme apresentado na terceira linha da Tabela 2, demonstram que a abordagem escolhida pelos autores foi capaz de ser mais efetiva que os algoritmos baseados em aprendizado de máquina na classificação por polaridade, já na classificação por subjetividade em alguns casos o algoritmo também foi melhor que os concorrentes.

Tabela 2 – Resultados da classificação por polaridade

| | Positive | | | Negative | | | Avg. |
|---|---|---|---|---|---|---|---|
| | Pr. | R. | F1 | Pr. | R. | F1 | F1 |
| Majority | 0.0 | 0.0 | 0.0 | 67.4 | 100 | 80.5 | 40.3 |
| Random | 32.6 | 50.0 | 39.4 | 67.4 | 50.0 | 57.4 | 48.4 |
| **Lexicon** | **60.0** | **87.7** | **71.9** | **93.3** | **70.8** | **80.5** | **76.2** |
| SVMpr | 60.8 | 68.2 | 64.3 | 83.7 | 78.7 | 81.1 | 72.7 |
| SVMtf | 59.8 | 68.2 | 63.8 | 83.5 | 77.8 | 80.6 | 72.2 |
| NBpr | 53.7 | 73.8 | 62.2 | 84.5 | 69.2 | 76.1 | 69.2 |
| NBtf | 69.5 | 38.8 | 49.4 | 75.5 | 91.9 | 82.9 | 66.2 |
| MaxEnt | 56.1 | 69.2 | 91.9 | 83.2 | 73.8 | 78.2 | 70.1 |

Fonte: Thelwall e Paltoglou (2012).

Tabela 3 – Resultados da classificação por subjetividade

| | Positive | | | Negative | | | Avg. |
|---|---|---|---|---|---|---|---|
| | Pr. | R. | F1 | Pr. | R. | F1 | F1 |
| Majority | 79.2 | 100 | 88.4 | 0.0 | 0.0 | 0.0 | 44.2 |
| Random | 79.2 | 50.0 | 61.3 | 20.8 | 50.0 | 29.3 | 45.3 |
| **Lexicon** | **93.2** | **92.5** | **92.9** | **67.1** | **69.5** | **68.3** | **80.6** |
| SVMpr | 87.4 | 93.5 | 90.3 | 66.2 | 48.6 | 56.0 | 73.2 |
| SVMtf | 86.2 | 90.8 | 88.4 | 56.0 | 79.9 | 49.7 | 69.1 |
| NBpr | 88.4 | 89.3 | 88.8 | 57.4 | 55.2 | 56.3 | 72.6 |
| NBtf | 83.9 | 98.0 | 90.4 | 78.9 | 28.6 | 42.0 | 66.2 |
| MaxEnt | 83.9 | 91.0 | 87.3 | 49.3 | 33.3 | 39.8 | 63.6 |

Fonte: Thelwall e Paltoglou (2012).

# 3 DESENVOLVIMENTO DO SISTEMA

Este capítulo demonstra as etapas de desenvolvimento da aplicação. Na seção 3.1 são apresentados os requisitos funcionais e não funcionais do sistema. A seção 3.2 possui a especificação do sistema, com os diagramas da Unified Modeling Language (UML), de casos de uso, classes e atividades. A seção 3.3 apresenta a implementação, com as técnicas e tecnologias utilizadas, os principais trechos de código do sistema, além das tecnologias utilizadas e da interface do usuário.

## 3.1 REQUISITOS

O sistema descrito neste trabalho deve atender aos Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) descritos no Quadro 1 e Quadro 2 respectivamente, sendo que os requisitos funcionais estão vinculados aos casos de usos apresentados na Figura 10.

Quadro 1 – Requisitos funcionais

| **Requisitos funcionais (RF)** | **Casos de uso (UC)** |
|---|---|
| RF01: exibir os dados das melhores marcas já classificadas | UC05 |
| RF02: permitir ao usuário visualizar os dados de uma marca ou realizar uma pesquisa | UC01, UC02, UC04, UC06 |
| RF03: permitir ao usuário selecionar duas marcas e fazer uma comparação entre elas | UC03 |
| RF04: efetuar a pesquisa de opiniões em tempo real sobre uma marca no Twitter | UC01, UC02, UC04 |
| RF05: filtrar os dados vindos do Twitter visando remover ruídos e otimizar sua qualidade | UC02 |
| RF06: utilizar um algoritmo de análise de sentimentos sobre os dados gerados na pesquisa para medir a opinião dos usuários | UC02 |
| RF07: salvar as pesquisas num banco de dados para evitar buscas duplicadas num curto período de tempo | UC02 |
| RF08: efetuar as pesquisas com base em uma das categorias: diária, semanal, quinzenal ou mensal. | UC07 |

Fonte: elaborado pelo autor.

Quadro 2 – Requisitos não funcionais

| **Requisitos não funcionais (RNF)** |
|---|
| RNF01: apresentar uma interface responsiva para ser acessado tanto de computadores quanto de dispositivos móveis (Requisito Não Funcional - RNF) |
| RNF02: utilizar a técnica de votação em seus servidores, onde caso um erro aconteça em algum dos servidores este seja identificado e a informação correta retornada |
| RNF03: possuir seus servidores programados em Java e a interface em PHP e AngularJs para utilizar os recursos disponíveis nestas linguagens |
| RNF04: efetuar a análise de sentimentos utilizando como base a emoção das palavras definidas no SentiWordNet |
| RNF05: utilizar as técnicas de remoção de caracteres HTML, tokenização e de StopWords na filtragem de dados |

Fonte: elaborado pelo autor.

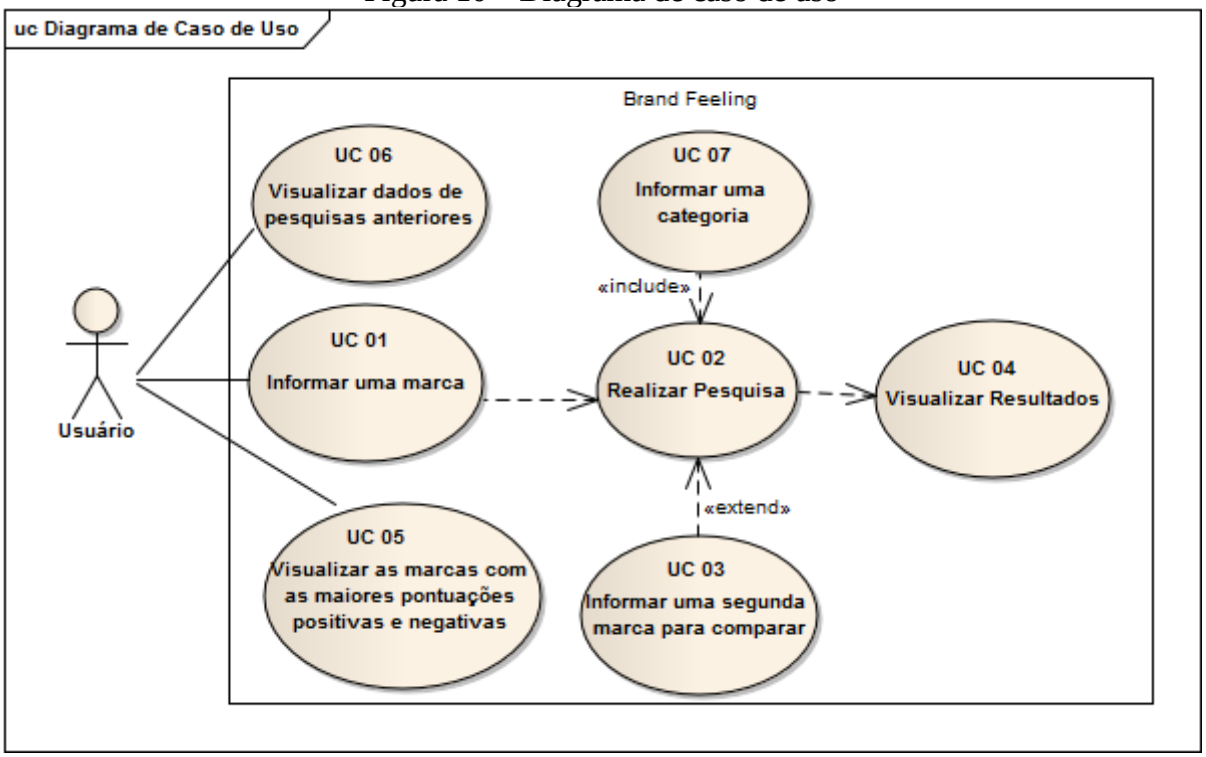
## 3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação do sistema. Esta especificação foi feita seguindo os padrões da UML, sendo os diagramas desenvolvidos através da ferramenta Entreprise Architet.

### 3.2.1 Diagrama de caso de uso

A Figura 10 mostra o diagrama de caso de uso do sistema, onde o usuário é o único ator envolvido, ele tem a opção de efetuar a pesquisa de uma marca (UC01 e UC02), selecionar uma categoria (UC07) e visualizar os resultados desta pesquisa (UC04), podendo adicionar a esta uma segunda marca para comparação (UC03). Além disso, o usuário pode também visualizar a listagem das marcas que possuem as maiores pontuações positivas e negativas (UC05), assim como visualizar pesquisas anteriores realizadas (UC06) sobre uma marca.

Figura 10 – Diagrama de caso de uso



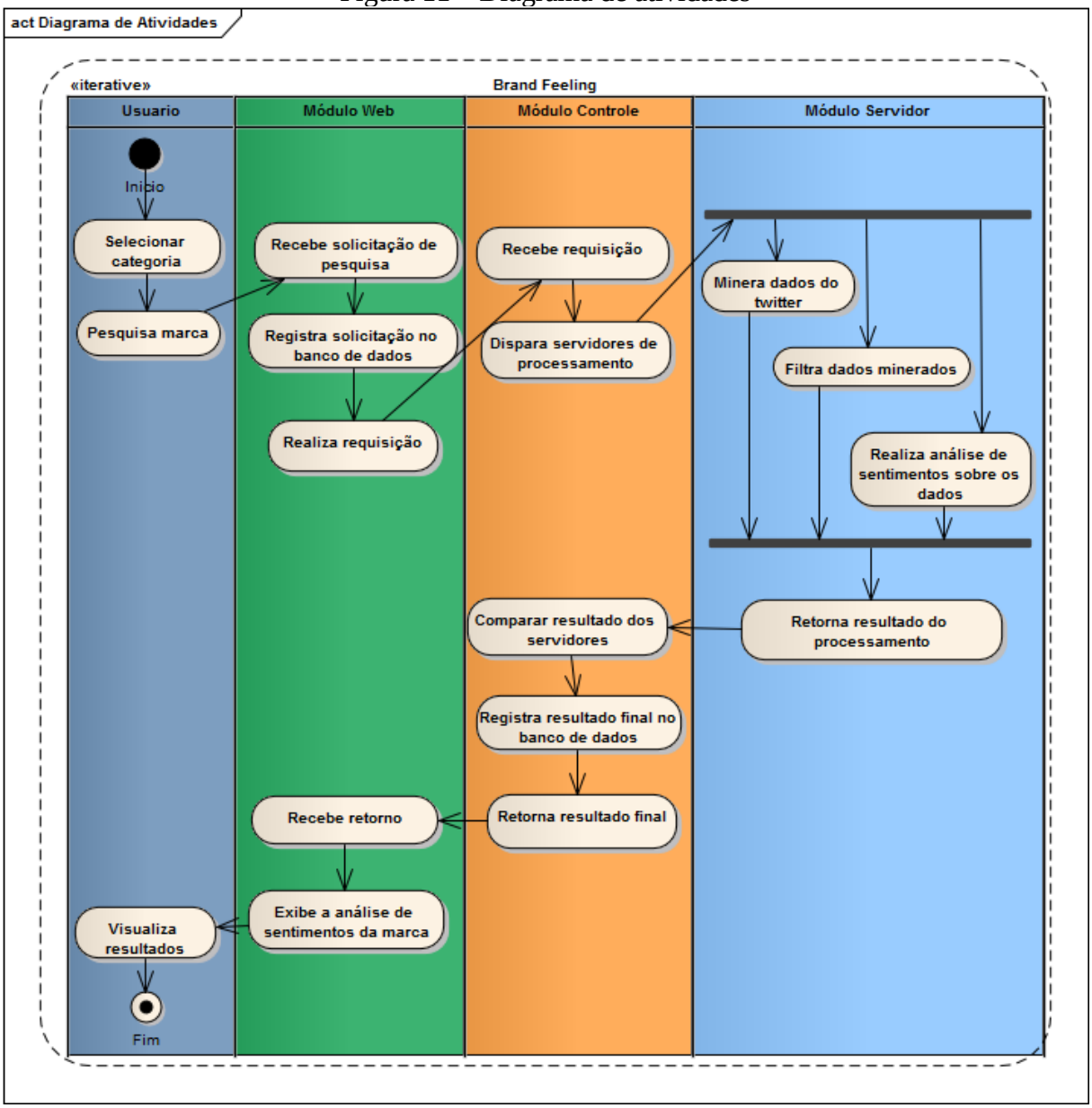

Fonte: elaborado pelo autor.

### 3.2.2 Diagrama de atividades

Nesta seção é apresentado o diagrama de atividades. A Figura 11 mostra o processo para a realização da pesquisa de uma marca. Neste processo o usuário deve realizar três ações simples, pesquisar uma marca, selecionar uma das categorias, que correspondem ao período de tempo sobre o qual a pesquisa será feita, e ao final do processo visualizar os resultados que

o sistema irá retornar. Após a requisição do usuário, o `Módulo Web` registra a solicitação no banco de dados e realiza a requisição da pesquisa ao `Módulo Controle`. O `Módulo Web` também é responsável por receber o retorno da pesquisa e mostrar os resultados em tela para o usuário no final do processo.

Figura 11 – Diagrama de atividades




Fonte: elaborado pelo autor.

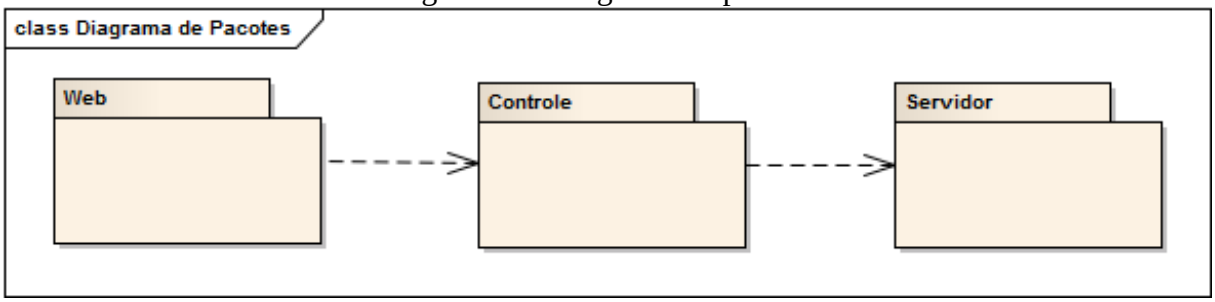
Ao receber a requisição o `Módulo Controle` se responsabiliza inicialmente por iniciar os servidores de processamento dos dados. Cada um destes servidores irá realizar o mesmo processo e no final retornar seu respectivo resultado ao `Módulo Controle`, que se encarrega de validar os resultados, registrar os mesmos no banco de dados e, por fim, de retorná-los ao `Módulo Web`.

Por fim o Módulo Servidor inicializa três ações simultâneas, onde cada uma delas espera existir o primeiro conteúdo para então realizar seu processo: a mineração dos dados do Twitter, a filtragem dos dados que foram minerados e, a realização da análise de sentimentos sobre estes dados. Uma vez finalizadas todas estas ações, ou seja, não existindo mais dados a serem processados, cada servidor irá retornar seus resultados ao Módulo Controle.

### 3.2.3 Diagrama de pacotes

Nesta seção é apresentado o diagrama de pacotes, assim como o diagrama de classes que cada pacote contém. O objetivo da utilização de pacotes foi o de organizar as diferentes classes do sistema pelo módulo em que elas se encontram, bem como separar as linguagens utilizadas no sistema, PHP e Java, em partes distintas. A Figura 12 apresenta o diagrama de pacotes do sistema.

Figura 12 – Diagrama de pacotes



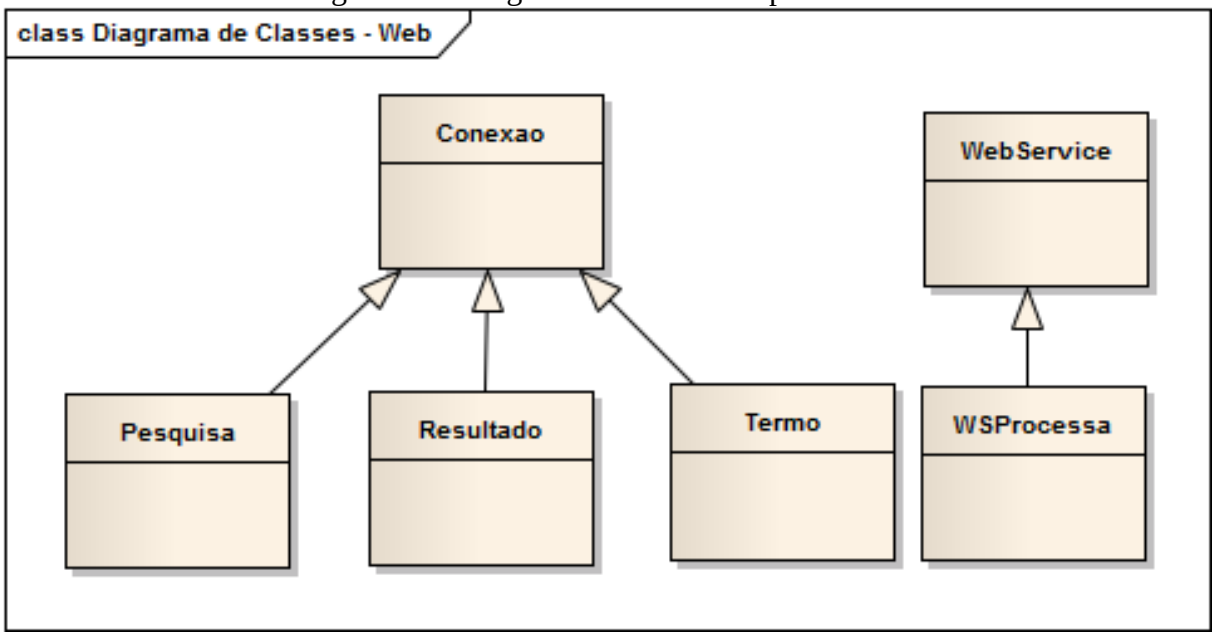

Fonte: elaborado pelo autor.

#### 3.2.3.1 Pacote web

O pacote web contém as classes correspondentes ao Módulo Web do sistema, programadas com a linguagem PHP. A Figura 13 apresenta estas classes e suas relações.

A classe base do Módulo Web é a Conexao, que se responsabiliza pela conexão com o banco de dados MySQL. A partir dela as demais classes realizam as consultas na base de dados. A classe Pesquisa se responsabiliza por buscar os dados de uma pesquisa, enquanto a classe Resultado busca os resultados finais de todas as buscas e a classe Termo realiza a busca dos termos de uma pesquisa. Para a realização da pesquisa é utilizada a classe WSProcessa, que possui o método específico para fazer a requisição de pesquisa através do protocolo de comunicação Protocolo Simples de Acesso a Objetos (SOAP), sendo que a comunicação em si é feita pela classe WebService, onde a requisição é enviada para o Módulo Controle.

Figura 13 – Diagrama de classes do pacote `Web`




Fonte: elaborado pelo autor.

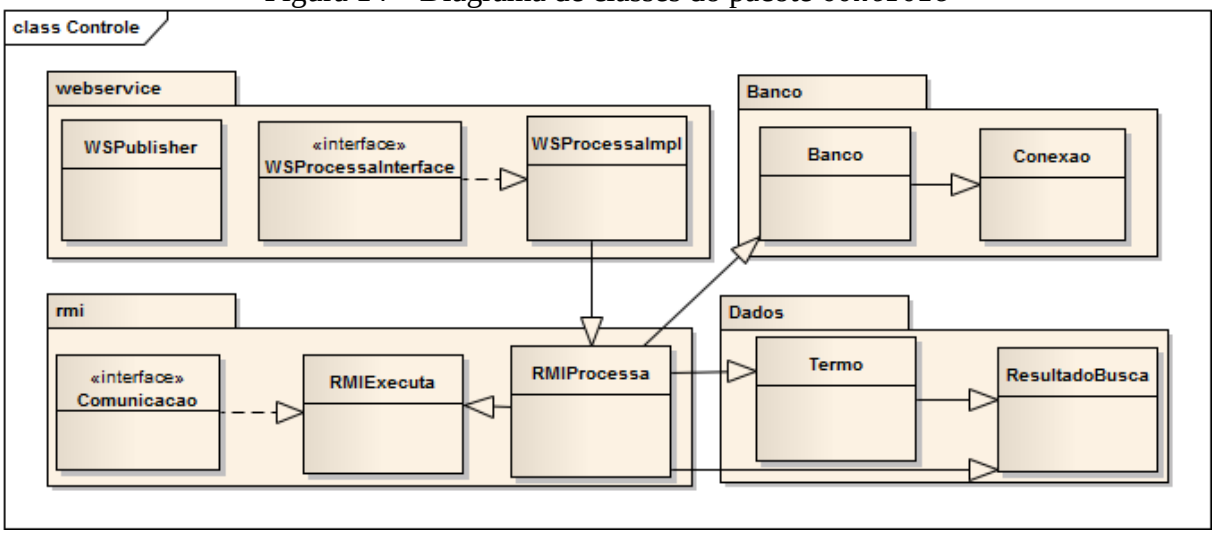
#### 3.2.3.2 Pacote `Controle`

O pacote de `Controle` contém as classes desenvolvidas em Java, que são responsáveis por controlar e repassar as requisições dos usuários ao servidor. A Figura 14 apresenta as classes divididas nos pacotes `webservice`, `rmi`, `banco` e `dados`.

O pacote `webservice` contém as classes que implementam os métodos da criação do webservice em Java que será acessado pelo `Módulo Web`. A classe `WSPublisher` disponibiliza um Localizador Uniforme de Recursos (URL) para acesso ao serviço, já a classe `WSProcessaImpl` faz a implementação dos métodos definidos na interface de comunicação com a tecnologia SOAP, definida na interface `WSProcessaInterface`. O pacote `Banco` agrupa as classes de conexão com o banco de dados, sendo que a classe `Banco` possui os métodos específicos a serem executados, enquanto que a classe `Conexao` se responsabiliza por estabelecer uma conexão e de ter os métodos básicos de SQL.

O pacote `rmi` se encarrega da implementação da comunicação via Remote Method Invocation (RMI). Nele, a interface `Comunicacao` define os métodos existentes na interface RMI, enquanto que a classe `RMIExecuta` faz as chamadas destes métodos e repassa o resultado para a classe `RMIProcessa`. A classe `RMIProcessa` se encarrega de gerir os três servidores de mineração, fazer a validação da votação e gravar o resultado na base. Por fim, o pacote de `Dados` contém a classe de `Termo` e de `ResultadoBusca`, utilizadas apenas para validação da pesquisa e para salvar os dados na base.

Figura 14 – Diagrama de classes do pacote `Controle`




Fonte: elaborado pelo autor.

#### 3.2.3.3 Pacote `Servidor`

O pacote `Servidor` contém as classes que fazem a mineração e a análise de dados, apresentadas na Figura 15. Dentro deste pacote existem subdivisões das classes que são: `rmi leitura, filtragem, analise, e dados.`

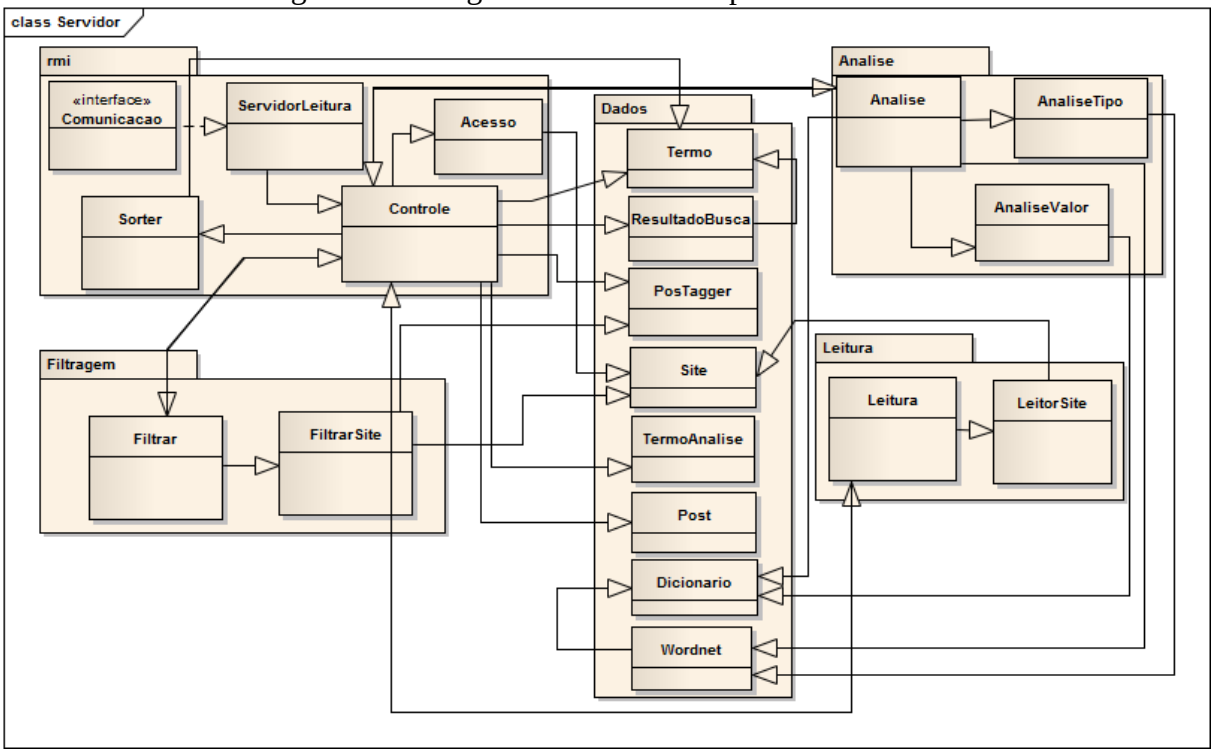
O pacote `rmi` contém as classes que criam o servidor RMI que é acessado pelo `Controle`. Este pacote possui a interface de comunicação RMI `Comunicacao`, a classe `ServidorLeitura` que implementa a interface, a classe `Sorter` que faz a ordenação dos resultados no fim, a classe `Acesso` que contém os sites a serem usados e suas configurações e a classe de `Controle`, responsável por fazer o controle interno do `Servidor`. No pacote de `Leitura` estão as classes que fazem a leitura dos sites, na qual a classe `Leitura` controla as *threads*, e a classe `LeitorSite` é executada na forma de várias *threads* e que varre o conteúdo dos sites.

O pacote `Filtragem` possui as classes que efetuam o filtro sobre os dados minerados. A classe `Filtrar` é responsável por iniciar e gerir as *threads* de filtro para cada site, e a classe `FiltrarSite` é responsável por efetuar os métodos de filtro. O pacote de `Analise` se encarrega de fazer a análise sobre os dados filtrados anteriormente, onde a classe `Analise` contém as *threads* de análise, a `AnaliseTipo` se responsabiliza por achar o radical de um termo inicial e determinar sua categoria sintática dentro da postagem, enquanto que a `AnaliseValor` se encarrega de verificar qual o valor do sentimento de cada termo.

Por fim, o pacote de `Dados` possui todas as classes existentes que representam algum dado no sistema. A classe `Termo` representa cada palavra e sua análise, a classe `ResultadoBusca` representa o resultado final da busca para cada marca pesquisada, a classe

`PosTagger` chama as funções da biblioteca de POS Tagger. A classe `Site` possui os dados de um site enquanto que a classe `TermoAnalise` representa um termo que ainda não foi analisado no processo, já a classe `Post` representa uma postagem minerada. Por sua vez, a classe `Dicionario` contém o dicionário do SentiWordNet a ser utilizado no sistema, e, por fim, a classe `WordNet` se comunica com a biblioteca JWI para encontrar o radical das palavras.

Figura 15 – Diagrama de classes do pacote `Servidor`




Fonte: elaborado pelo autor.

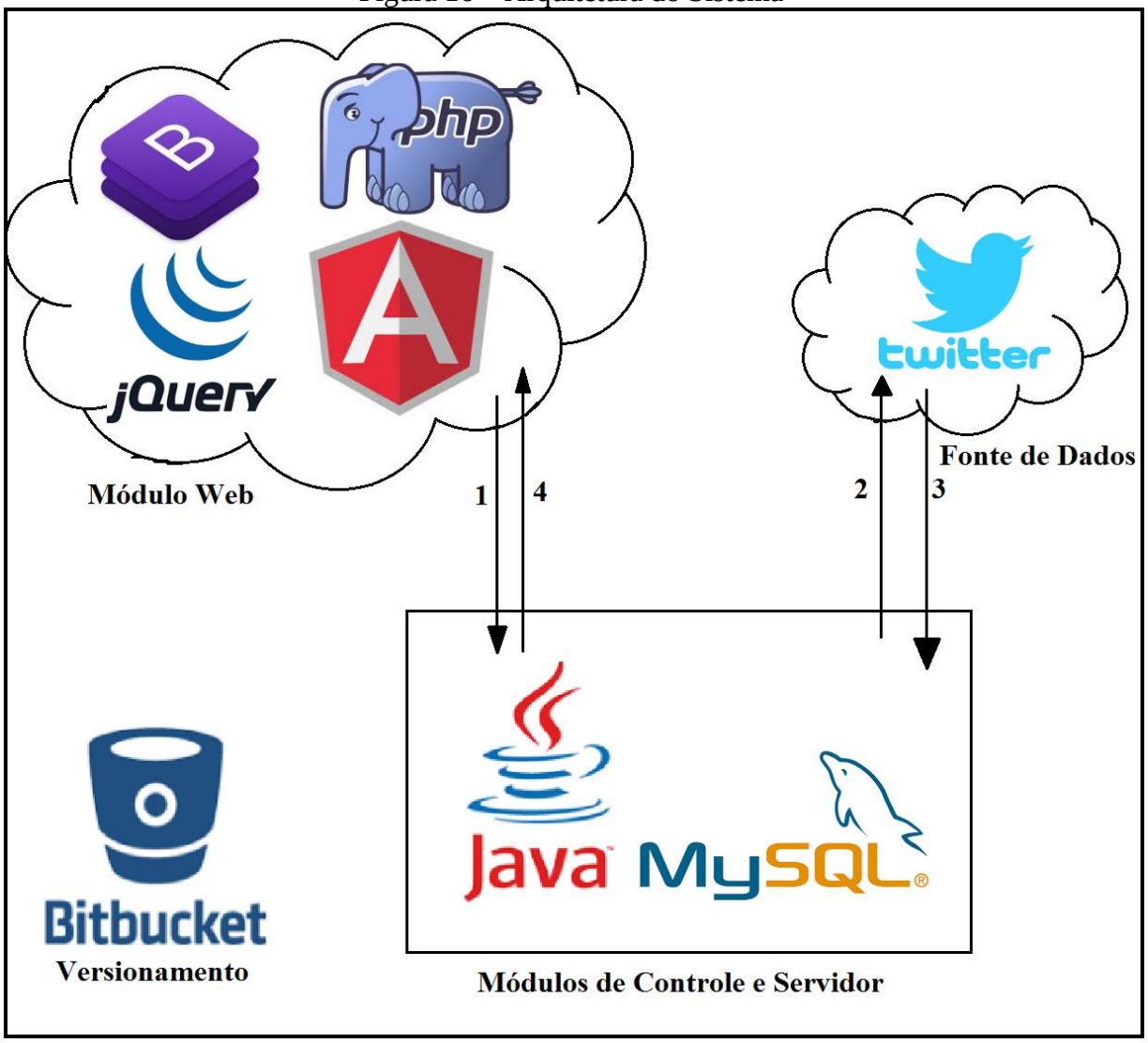
## 3.3 IMPLEMENTAÇÃO

A seguir são apresentadas as técnicas e ferramentas utilizadas, o funcionamento das partes principais do sistema e a operacionalidade da implementação.

### 3.3.1 Técnicas e ferramentas utilizadas

O sistema foi desenvolvido em três módulos: `Controle`, `Servidor` e `Web`. Tanto o `Módulo Controle` e o `Servidor` foram desenvolvidas na linguagem Java, pois esta tecnologia é mais eficiente no trabalho com programação concorrente (*threads*). Além disso, esta linguagem possibilita a utilização da técnica de votação, criando múltiplas instâncias de `Servidor` que se comunicam com o `Controle` através do RMI. Na Figura 16 é mostrada a arquitetura de todo o sistema, onde o processo origina-se do `Módulo Web` (1), realiza a requisição para o `Módulo Controle` e o `Módulo Servidor`, que buscam os dados do Twitter (2), e após realizar a análise (3) retornam o resultado para o `Módulo Web` (4).

Figura 16 – Arquitetura do Sistema




Fonte: elaborado pelo autor.

Nos módulos desenvolvidos na linguagem Java foi utilizado o software Netbeans por conta da praticidade que este oferece em relação a arquitetura do sistema. Para a mineração de dados da web foi utilizada a classe do Java `URLConnection`, que permite ler o conteúdo de um site. Para a análise de sentimentos foram utilizadas as tecnologias detalhadas na seção 3.3.2.

Para armazenar os dados das pesquisas e os resultados é utilizado o banco de dados MySQL, devido a sua compatibilidade e facilidade ao trabalhar tanto com Java quanto com Hypertext Preprocessor (PHP). Para a conexão entre os módulos do sistema foi utilizado o protocolo de comunicação SOAP, que cria uma interface de comunicação entre as distintas linguagens utilizadas para o desenvolvimento.

O `Módulo Web` foi desenvolvido utilizando a linguagem PHP em sua versão 5.5. Para o desenvolvimento das interfaces do sistema foi utilizado o *framework* Bootstrap 3, que fornece ferramentas para criar interfaces intuitivas e responsivas, isto é, prepara o sistema para uso

por qualquer dispositivo. Simultaneamente foi utilizada a biblioteca Jquery e o *framework* AngularJs, este último sendo o responsável pela realização das pesquisas de forma assíncrona, carregando os resultados da pesquisa em etapas. Por fim, para o desenvolvimento do `Módulo Web` foi utilizado o SublimeText 3. Durante o desenvolvimento do sistema foi utilizada a ferramenta de controle de versão SourceTree, através do serviço on-line BitBucket.

### 3.3.2 Tecnologias utilizadas

Nesta seção são conceituadas as tecnologias de terceiros que foram utilizadas no desenvolvimento deste trabalho, assim como uma explicação de seu propósito e também a conceituação de seu funcionamento. Para exemplificar as bibliotecas foi utilizada a frase “Kinect is getting killed by Microsoft, and people are really sad about it”, que significa “Kinect está sendo morto pela Microsoft, e as pessoas estão realmente tristes sobre isto”.

#### 3.3.2.1 WordNet

O WordNet é um grande banco de dados léxico de palavras da língua inglesa, onde as palavras com mesmo sentido ou sinônimas são agrupadas em um único termo. Cada um destes agrupamentos de palavras pode pertencer a uma categoria sintática, sendo elas: N – substantivo; V – verbo; R – advérbio; A – adjetivo principal e S – adjetivo satélite. A subdivisão dos adjetivos existe para separar os adjetivos básicos (principais) dos demais. Esta biblioteca foi desenvolvida visando disponibilizar uma ferramenta semelhante a um dicionário para o uso por aplicações (MILLER, 1995).

A utilização da biblioteca neste trabalho se faz necessária para o procedimento de lematização das palavras ao seu radical, onde sobre cada termo pesquisado é verificado seu radical. Neste trabalho foi utilizada a versão 3.0, sendo essa a mais atual da biblioteca.

#### 3.3.2.2 SentiWordNet

O SentiWordNet é uma biblioteca léxica que classifica expressões da língua inglesa em relação ao seu sentimento, desenvolvida para permitir a criação de aplicações que utilizem a análise de sentimentos e a mineração de opinião sobre textos. Esta biblioteca foi desenvolvida com base no WordNet, onde para cada grupo de palavras do WordNet foram atribuídos valores positivos e negativos visando numerar o sentimento de cada um (BACCIANELLA, 2010). Neste trabalho foi utilizada a versão 3.0 da biblioteca, que contempla uma versão atualizada dos termos, assim como um maior número de termos.

A escolha pelo SentiWordNet foi devido a sua utilização nos mais renomados trabalhos científicos da área apresentando confiabilidade em seu ranqueamento pelo número de termos contemplados, bem como pela forma de distribuição da biblioteca através de um arquivo de texto, que pode ser importado facilmente por sistemas. No Anexo A é demonstrada uma parte da biblioteca, contendo a explanação presente no arquivo de texto, assim como a Tabela 7, onde são mostrados grupos de palavras distintos de cada uma das categorias gramaticais definidas pelo WordNet. Na Tabela 4 é mostrada a positividade das palavras que pertencem a frase base e possuem alguma polaridade, segundo o SentiWordNet, em conjunto com suas respectivas classes gramaticais (POS).

Tabela 4 – Positividade das palavras segundo o SentiWordNet

| POS | Palavra | Positividade |
|---|---|---|
| r | really | 100% |
| v | kill | 0% |
| a | sad | 0% |
| v | get | 100% |

Fonte: elaborado pelo autor.

#### 3.3.2.3 MIT Java WordNet Interface

A biblioteca MIT Java WordNet Interface (JWI) é uma biblioteca desenvolvida no Instituto de Tecnologia do Massachusetts (MIT) que fornece uma interface para a utilização do WordNet em Java. Para a utilização desta biblioteca é carregado o dicionário do WordNet em memória, para que então sejam disponibilizados métodos para trabalhar com o mesmo, incluindo métodos para efetuar a lematização das palavras (FINLAYSON, 2014).

Neste trabalho a biblioteca JWI foi utilizada para efetuar a lematização de palavras ao seu radical, utilizando-a em conjunto com o WordNet. Assim unificando os termos em qualquer tempo verbal ou número, e facilitando o reconhecimento dos mesmos pela análise de sentimentos. Foi extraído a partir da frase base cada palavra e seu respectivo radical, utilizando esta biblioteca em conjunto com o Wordnet, as palavras que sofreram alterações são destacadas na Tabela 5.

Tabela 5 – Lista de palavras e radicais extraídos

| Palavra | Radical |
|---|---|
| getting | get |
| killed | kill |

Fonte: elaborado pelo autor.

#### 3.3.2.4 Stanford Log-linear Part-Of-Speech Tagger

A biblioteca de POS Tagger desenvolvida na Universidade de Stanford fornece uma ferramenta para a divisão gramatical das palavras de diferentes linguagens em Java

(TOUTANOVA et al., 2003). Na inicialização desta biblioteca deve ser informado qual idioma será utilizado, neste caso o inglês, e, após isso, podem ser utilizados os métodos da biblioteca para a separação gramatical das palavras.

No sistema proposto, a biblioteca de POS Tagger de Stanford tem o propósito de classificar as frases oriundas das postagens do Twitter, definindo para cada termo de cada postagem sua classe gramatical na língua inglesa. Uma vez definida a classe para cada termo a mesma é utilizada na análise de sentimentos, tanto para retirar o radical das palavras, através da utilização no JWI, quanto para a busca do valor do termo no SentiWordNet, pois este diferencia o valor das palavras para cada classe gramatical.

Ao utilizar esta biblioteca para a frase base o retorno é mostrado na Figura 17, a qual foi classificada em relação às classes gramaticais relevantes para a aplicação: substantivo (prefixo NN), palavras Kinect, Microsoft e *people*; verbo (prefixo VB), palavras *is*, *getting*, *killed* e *are*; adjetivo (prefixo JJ), palavra *sad*; advérbio (prefixo RB), palavra *really*.

Figura 17 – Resultado da aplicação da biblioteca de POS Tagger

```
The_DT Kinect_NNP is_VBZ getting_VBG killed_VBN by_IN Microsoft_NNP ,_,
and_CC people_NNS are_VBP really_RB sad_JJ about_IN it_PRP
```

Fonte: elaborado pelo autor.

### 3.3.3 Funcionamento do sistema

Nesta seção são demonstrados os principais trechos de código do sistema. São detalhados o funcionamento da programação concorrente do sistema, assim como de cada área demonstrada no diagrama de atividades, a mineração de dados, a filtragem de dados e por fim a análise de sentimentos.

#### 3.3.3.1 Funcionamento da programação concorrente

No Quadro 3 pode-se visualizar o trecho de código no `Servidor` da classe `Controle`, responsável por inicializar o processo da busca e classificação de uma marca. Para isso, nessa etapa são inicializadas as 3 partes do processo em paralelo: a leitura de dados, onde é realizada a mineração de dados da web; a filtragem de dados, onde é tratada cada postagem encontrada na leitura; e, por fim, a análise dos dados, onde para cada novo termo separado pela filtragem é efetuada a análise de sentimentos para buscar o seu valor.

Cada parte tem seu método de inicialização, sendo que cada um realiza o *start* das respectivas *threads*. No final deste processo de inicialização são chamados os respectivos

métodos de finalização, para que o sistema aguarde a finalização de todas as *threads*. Por fim, é efetuada a finalização de todo o processo, formatando o resultado final.

Outro controle realizado é para que todos os métodos que utilizam *threads* antes de iniciar seu processamento requisitem um lugar no semáforo, e no final de seu processamento liberem seu lugar. Este semáforo é uma estrutura dentro da programação concorrente que visa limitar o número de *threads* simultâneas dentro do sistema, assim impedindo que problemas aconteçam na execução.

Quadro 3 – Inicialização do processo

```
public void disparaPrograma(ArrayList<String> listaMarcas){
   this.leitura = new Leitura();
   this.leitura.setControle(this);
   this.leitura.iniciaLeitura();

   this.filtrar = new Filtrar();
   this.filtrar.setControle(this);
   this.filtrar.iniciaFiltragem();

   this.analise.iniciaAnalise();

   this.leitura.finalizaLeitura();
   this.filtrar.finalizaFiltrar();
   this.analise.finalizaAnalise();

   this.finalizaProcesso();
}
```

Fonte: elaborado pelo autor.

O Quadro 4 apresenta área de concorrência envolvendo a lista de postagens, que está situada no `Servidor`, na classe `Controle`. Esta lista é populada pelas *threads* de leitura através do método `adicionarPost`, e paralelamente é lida pelas *threads* de filtragem de dados, através do método `getProximoPost`. Por conta deste paralelismo no acesso aos dados é necessário realizar um controle deste acesso, permitindo apenas uma *thread* por vez, onde após a execução é necessário avisar as demais para que o processo continue.

Quadro 4 – Área de concorrência das postagens

```
public synchronized void adicionarPost(String marca, String post){
        Post postagem = new Post(post, marca);
        this.getListaPostagem().add(postagem);
        notifyAll();
    }

public synchronized Post getProximoPost(){
        Post retorno = null;

        while(this.estadoLeitura == 0 ||
(this.getListaPostagem().isEmpty() && this.estadoLeitura == 1)){
           wait(100);
        }

        if(this.getListaPostagem().isEmpty()){
             return retorno;
        }

        for (Post postagem : this.getListaPostagem()) {
            retorno = postagem;
            break;
        }
        this.getListaPostagem().remove(retorno);

        notifyAll();
        return retorno;
    }
```

Fonte: elaborado pelo autor.

### 3.3.3.2 Execução da mineração de dados

No Quadro 5 é demonstrada a inicialização dos sites dos quais serão minerados os dados no sistema, este trecho está situado no `Servidor` da classe `Acesso`. Uma pesquisa de uma marca tem como base a utilização de dois sites base que representam a busca avançada do Twitter. O primeiro site representa a aba `Em Destaque`, que traz as postagens com maior relevância, segundo o algoritmo do Twitter, e o segundo site traz os resultados da aba `Últimas`, ou seja, as postagens mais recentes. Cada Site possui como atributos: a URL de acesso; a codificação do site; qual o caracter que simboliza o espaço em branco na busca; o identificador que indica em qual parte do site estão os dados; e, por fim, uma lista de classes a serem ignoradas.

Quadro 5 – Inicialização dos sites

```
public void iniciaSites(){
       this.getSitePesquisa().add(new Site(
        "https://twitter.com/search?f=tweets&q=\"replace\"&l=en&src=typd",
         "UTF-8", "%20", "js-tweet-text-container", listaClassesIgnorar));

       this.getSitePesquisa().add(new Site(
"https://twitter.com/search?vertical=default&q=\"replace\"&l=en&src=typd",
         "UTF-8", "%20", "js-tweet-text-container", listaClassesIgnorar));

       if(this.getCategoria() == 0){
          return ;
       }

       switch(this.getCategoria()){
         case 1:{ nrDias = 7; break; }
         case 2:{ nrDias = 15; break; }
         case 3:{ nrDias = 30; break; }
       }

       for (int i = 0; i < nrDias; i++) {
          String dataFinal = getData(-i);
          String dataInicio = getData(-(i+1));

       this.getSitePesquisa().add(new Site(
        "https://twitter.com/search?f=tweets&q=\"replace\"&l=en
%20since%3A"+dataInicio+"%20until%3A"+dataFinal+"&src=typd",
         "UTF-8", "%20", "js-tweet-text-container", listaClassesIgnorar));

       this.getSitePesquisa().add(new Site(     "
"https://twitter.com/search?vertical=default&q=\"replace\"&l=en
%20since%3A"+dataInicio+"%20until%3A"+dataFinal+"&src=typd",
         "UTF-8", "%20", "js-tweet-text-container", listaClassesIgnorar));

       }
    }
```

Fonte: elaborado pelo autor.

No Quadro 5 é demonstrado que caso o usuário escolha uma categoria diferente da diária, serão adicionados a pesquisa novos sites, onde cada site corresponde a busca avançada do Twitter mas com a adição de um intervalo de datas ao final da URL desta busca. Esta verificação é feita com base na variável da classe `this.getCategoria()`,onde 0 indica a categoria diária, 1 a semanal, 2 a quinzenal e 3 a mensal.

No Quadro 6 é demonstrado o processo de leitura de dados de um site, onde a biblioteca do Java `URLConnection` se encarrega de buscar o conteúdo do mesmo e armazena-lo em um *buffer*. A leitura então é feita para cada linha deste site, onde primeiramente é localizada a tag `body`, que indica o começo do conteúdo da página, seguido do identificador dos dados a serem minerados deste site. Uma vez encontrado este identificador, o conteúdo desta linha do site é então adicionado à lista de postagens.

Quadro 6 – Leitura de dados de um site

```
private void efetuaLeitura(String endereco){
       this.controle.requisitaLugarSemaforo();
       URLConnection connection = new URL(endereco).openConnection();
       connection.connect();

       BufferedReader in = new BufferedReader(
             new InputStreamReader(connection.getInputStream(),
             this.site.getCodificacao()));

       while ((inputLine = in.readLine()) != null){

         if(inputLine.contains("<body")){dentroDoBody = true;}
         if(!dentroDoBody){continue;}

         linhaFormatada = this.tratarLinha(inputLine);
         if(linhaFormatada.contains(this.site.getTagInformacoes()) ){
            dentroConteudo = true;
         }

         if(!dentroConteudo){continue;}
         if(inputLine.contains("</div>")){
            dentroConteudo = false;continue;
         }

         linhaFormatada = this.removeHtml(inputLine);
         linhaFormatada = this.tratarLinha(linhaFormatada);
         if(linhaFormatada.equals("")){continue;}

         this.controle.adicionarPost(marca, inputLine);
       }
       this.controle.liberaLugarSemaforo();
}
```

Fonte: elaborado pelo autor.

### 3.3.3.3 Execução da filtragem dos dados

Para a filtragem de dados são utilizadas duas técnicas: StopWords ou StopList e o tratamento dos caracteres HTML, ambas situadas na classe `FiltrarSite` no `Servidor`. A primeira técnica, StopWords ou StopList, detalhada na seção 2.1.1, é populada na inicialização da filtragem, conforme demonstrado no Quadro 7. Esta listagem foi definida com base no trabalho de Schütze (2008). Porém, foram adicionadas mais palavras durante o desenvolvimento do sistema, visando remover as palavras que não agregam valor para análise, sendo elas: *any*, *her*, *I*, *me*, *my*, *so*, *you*.

Quadro 7 – Lista de StopWords

```
private void populaStopList(){
       String[] stopListArray =
        {"a","an","and","any","are","as",
        "at","be","by","for","from","has",
        "he","her","i","in","is","it",
        "its","me","my","of","on","so",
        "that","the","to","was","were",
        "will","with","you"};

        for (String stopWord : stopListArray) {
            this.adicionaTermoIgnorar(stopWord);
}}
```

Fonte: elaborado pelo autor.

A segunda técnica se baseia em tratar os caracteres provenientes do HTML, através de uma série de métodos conforme demonstrado no Quadro 8. O método `filtraTweets` percorre a lista de postagens disponíveis. Para cada postagem é realizado um tratamento, que começa com o método de remoção de links, seguido da remoção das classes ignoradas de cada site, da remoção de tags HTMLs, do tratamento do texto visando remover caracteres específicos, e da remoção de menções a outros usuários e números. Depois deste tratamento é chamada a biblioteca de POS Tagger, onde é efetuada a divisão gramatical dos termos contidos em cada postagem, e, por fim, é realizada a divisão da postagem em uma lista, onde cada termo da postagem é adicionado à lista de análise.

Quadro 8 – Tratamento de caracteres HTML

```
private void filtraTweets(){
        this.controle.requisitaLugarSemaforo();
        while((Post postagem = this.controle.getProximoPost())
!= null){
          texto = this.removeLink(postagem.getPostagem());
          texto = this.removeIngorados(texto);
          texto = this.removeHtml(texto);
          texto = this.tratarLinha(texto);
          texto = this.removeUsers(texto);
          texto = this.removeNumeros(texto);
          if(texto.equals("")){continue;}

          textoClassificado = tagger.getStringTagged(texto);

          String[] termos = texto.split(" ");
          for (String termo : termos) {
            termo = termo.trim();
            if(termo.equals("") || (termo.length() == 1) ||
this.termoNaLista(termo)){
               continue;
            }

            this.getControle().adicionaTermoTratar(
               postagem.getMarca(), termo, textoClassificado);
          }
          this.controle.liberaLugarSemaforo();
}}
```

Fonte: elaborado pelo autor.

#### 3.3.3.4 Execução da análise de sentimentos

A execução da análise de sentimentos foi dividida em duas seções: análise por tipo, que corresponde a aplicação da técnica de lematização e a análise por valor, que corresponde a análise de sentimentos. Na primeira seção é detalhada a análise por tipo, onde é realizada a lematização dos termos, através do método `analisaTermo`, que está na classe `AnaliseTipo` do `Módulo Servidor`. Este processo é demonstrado no Neste método primeiramente é encontrada a classe gramatical do termo, utilizando a função auxiliar `getTipoPos`, que faz a tradução da classificação gramatical previamente gerada na filtragem para os tipos do SentiWordNet. Na sequência, caso o tipo seja relevante, é feita a busca do radical da palavra utilizando o WordNet e a biblioteca JWI. No caso em que a busca retorna um termo diferente, o mesmo é atualizado. No fim do processo o termo final é então adicionado a listagem de termos.

Quadro 9.

Neste método primeiramente é encontrada a classe gramatical do termo, utilizando a função auxiliar `getTipoPos`, que faz a tradução da classificação gramatical previamente gerada na filtragem para os tipos do SentiWordNet. Na sequência, caso o tipo seja relevante, é feita a busca do radical da palavra utilizando o WordNet e a biblioteca JWI. No caso em que a busca retorna um termo diferente, o mesmo é atualizado. No fim do processo o termo final é então adicionado a listagem de termos.

Quadro 9 – Execução da análise por tipo

```
private void analisaTermos(){
       this.controle.requisitaLugarSemaforo();
       while((TermoAnalise termoAnalise =
             this.controle.getProximoTermoTratar()) != null){
         tipo = this.getTipoPos(termoAnalise);
         if(tipo != ""){
           termoNovo = this.getWordnet().getRadicalPalavra(
            termoAnalise.getTermo(),tipo.charAt(0));

           if(!termoNovo.equals(termoAnalise.getTermo())){
             termoAnalise.setTermo(termoNovo);
           }
       }}
       this.controle.adicionaTermo(
         termoAnalise.getMarca(), termoAnalise.getTermo(), tipo);
       this.controle.liberaLugarSemaforo();
}

private void getTipoPos(){
       Pattern pattern = Pattern.compile(termoAnalise.getTermo() +
"[^\\s]\\w+");
       Matcher matcher = pattern.matcher(
         termoAnalise.getTextoFormatado()).find();
```

```
        String[] termo = matcher.group(0).split("_");
        switch(termo[1]){
          case "VB": case "VBD": case "VBG": case "VBN": case "VBP":
          case "VBZ": return "v";
          case "NN": case "NNS": case "NNP": case "NNPS":
            return "n";
          case "JJ": case "JJR": case "JJS":
            return "a";
          case "RB": case "RBR": case "RBS":
            return "r";
             }
 return "";}
```

Fonte: elaborado pelo autor.

A interação com a biblioteca JWI e o Wordnet é realizada na classe `WordNet`, na qual os principais trechos desta classe estão demonstrados no Quadro 10. Nesta classe é feita a inicialização do dicionário do WordNet no método `iniciaWordNet` com base no caminho físico da biblioteca. Após a inicialização do dicionário o mesmo é passado para a classe `WordnetStemmer`, classe essa interna do JWI.

O método `getRadicalPalavra`, que é o mesmo chamado pela análise de tipo, trata de usar a biblioteca JWI para buscar os *stems*, que no caso do JWI correspondem aos possíveis radicais da palavra passada. Antes de buscar os radicais é efetuada a busca da classe gramatical no formato do JWI através da chamada da função `getPosFromChar`, na qual retorna a constante que representa o tipo no WordNet.

Quadro 10 – Busca do radical da palavra, usando o WordNet e JWI

```
private void iniciaWordNet(){
      this.dicionario = new Dictionary(new File(pathToWn));
      this.dicionario.open();
      this.stemmer = new WordnetStemmer(
                          (IDictionary) this.dicionario);
}

public String getRadicalPalavra(String palavra, char tipo){
      String retorno = palavra;
      List<String> stems = stemmer.findStems(palavra,
                            this.getPosFromChar(tipo));
      if(stems.isEmpty()){ return retorno; }

      for (int i = 0; i < stems.size(); i++) {
         retorno = stems.get(i);
      }
      return retorno;
}

private POS getPosFromChar(char tipo){
      switch(tipo){
        case 'v':{ return POS.VERB; }
        case 'n':{ return POS.NOUN; }
        case 's':
        case 'a':{ return POS.ADJECTIVE; }
        case 'r':{ return POS.ADVERB; }
      }
      return null;
```

```
}
```

Fonte: elaborado pelo autor.

Na segunda seção é executada a análise do valor dos termos na classe `AnaliseValor` do `Servidor`, conforme demonstrado no Quadro 11. Esta análise ocorre com base no dicionário que possui os valores de cada termo definido pelo SentiWordNet, onde a população deste dicionário pode ser visualizada no Apêndice A. O método `avaliaTermo` faz então a busca no dicionário deste termo e seta na própria classe do termo essa lista com os valores.

Quadro 11 – Execução da análise de valores

```
private void avaliaTermo (){
        HashMap<String, Double> hashTemp;
        this.controle.requisitaLugarSemaforo();

        //busca os termos para setar o valor
        while((Termo termo = this.controle.getProximoTermo())
              != null)
        {
         hashTemp = this.dicionario.getDictionary()
                              .get(termo.getTermo());

         this.controle.setNotaTermo(termo.getTermo(),hashTemp);
        }
        this.controle.liberaLugarSemaforo();
}
```

Fonte: elaborado pelo autor.

### 3.3.3.5 Processo de votação

O Quadro 12 apresenta a criação de 3 classes `RMIExecuta` no `Módulo Controle`, na função `efetuaLeitura`, onde cada uma destas classes é uma *thread* e se responsabiliza em criar e chamar sua própria instância do `Servidor`. Após a execução de cada classe, os resultados são comparados no sistema de votação, para verificar que os dados de saída sejam os mesmos.

A comparação acontece no método `efetuaVotacao`, localizado na classe `RMIProcessa` do pacote `Controle`. Nele são comparados os resultados de cada servidor, e, caso seja identificado dois resultados idênticos é feito o retorno destes dados, já que isto prova

que o resultado está correto. Os dados usados para esta comparação são: número de termos, porcentagem positiva e porcentagem negativa do resultado.

Quadro 12 – Processo de Votação

```
private HashMap<String,ResultadoBusca> efetuaLeitura(ArrayList<String>
marcas, Integer categoria){
        RMIExecuta executa1,executa2,executa3;
        HashMap<String,ResultadoBusca> retornoFinal;

        executa1 = new RMIExecuta(1, marcas, categoria);
        executa1.start();

        executa2 = new RMIExecuta(2, marcas, categoria);
        executa2.start();

        executa3 = new RMIExecuta(3, marcas, categoria);
        executa3.start();

        executa1.join();executa2.join();executa3.join();
        return this.efetuaVotacao(
          executa1.getRetorno(), executa2.getRetorno(),
          executa3.getRetorno()
        );
}
```

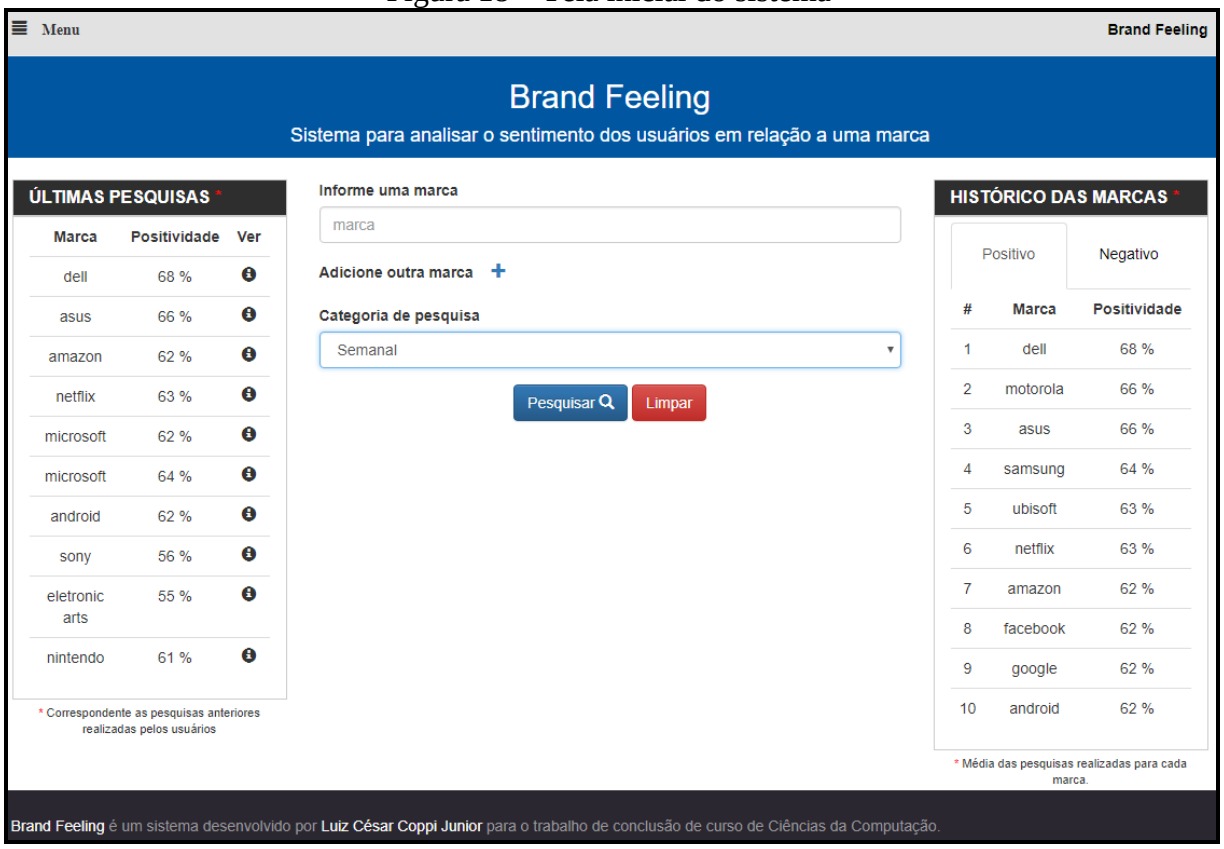
Fonte: elaborado pelo autor.

### 3.3.4 Operacionalidade da implementação

Este sistema foi desenvolvido como uma aplicação web que pode ser acessada pelo usuário através de uma URL. A tela inicial, apresentada na Figura 18, é o ponto de partida do usuário no sistema. No lado esquerdo da tela o usuário pode visualizar o grau de positividade das `Últimas Pesquisas` realizadas no sistema, bem como visualizar os resultados específicos de cada pesquisa através do ícone de informação ao lado delas. Outro dado disponível para o usuário é o `Histórico das Marcas`, que é baseado nas pesquisas realizadas anteriormente sobre cada marca por todos os usuários do sistema, onde é apresentado o grau de positividade médio de cada marca, sendo possível alternar entre as marcas que possuem resultados com sentimento mais `Positivo` e mais `Negativo`.

O usuário pode ainda navegar entre as categorias de pesquisa, utilizando a caixa de seleção no centro da tela. Ao alterar a categoria, tanto o `Histórico das Marcas` quanto as `Últimas Pesquisas` serão recarregadas com os dados para a categoria selecionada. Por fim, o usuário pode realizar uma nova pesquisa, selecionado a categoria, informando uma marca, opcionalmente informando uma segunda marca e clicando no botão `Pesquisar`.

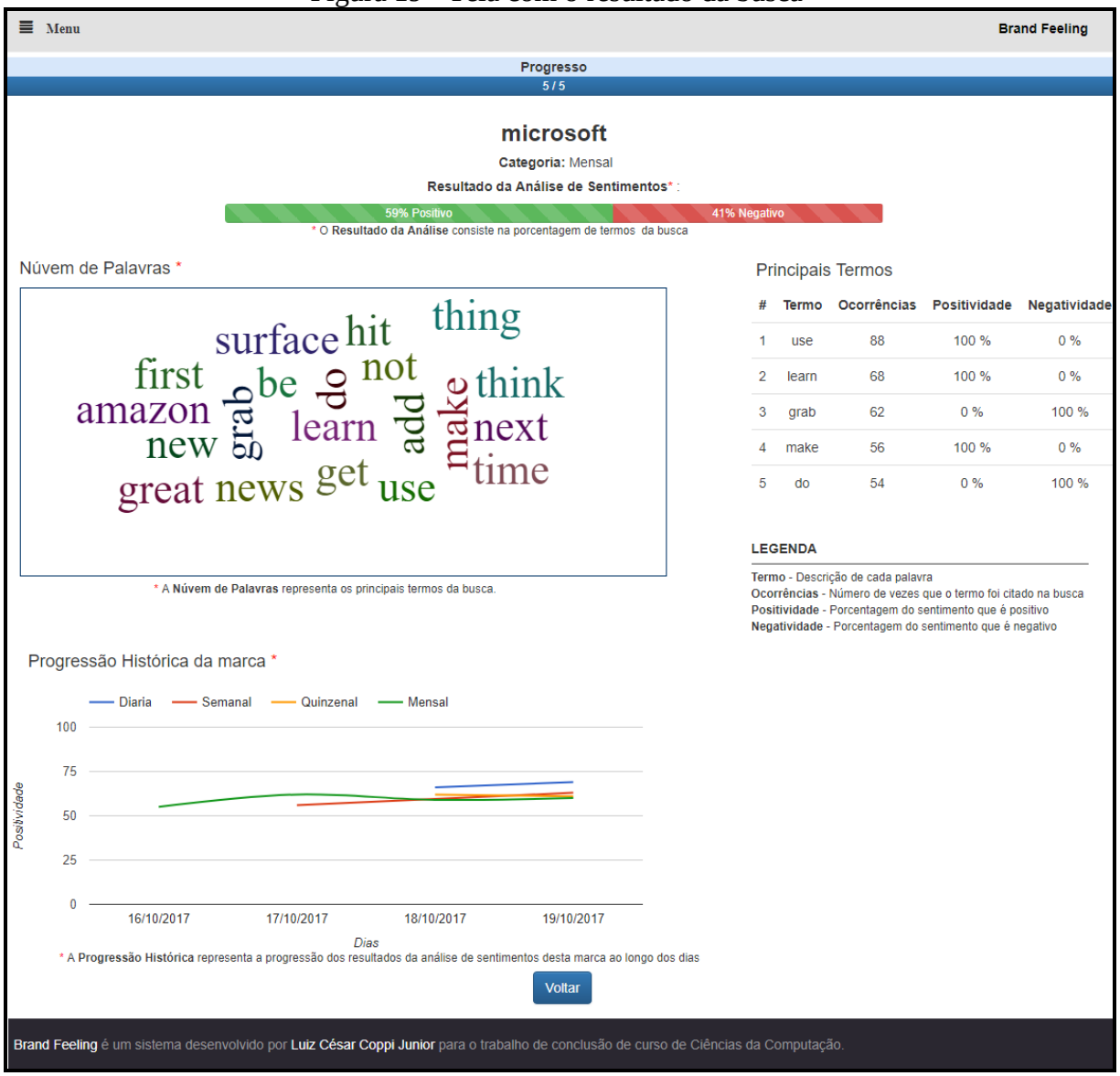
Figura 18 – Tela inicial do sistema




Fonte: elaborado pelo autor.

Ao realizar uma pesquisa, o usuário é redirecionado para a segunda tela do sistema onde são apresentados os resultados, conforme mostrado na Figura 19. Esta tela possui uma barra de progresso com 5 etapas pré-definidas, com o objetivo de informar o usuário sobre o progresso do sistema enquanto carrega as informações. As etapas consistem em: verificar na base de dados se esta marca foi pesquisada na mesma categoria na última hora, com o objetivo de evitar uma nova busca; realizar uma nova busca de dados, caso necessário; retornar os resultados da busca da base de dados; mostrar o resultado da busca, a `Nuvem de Palavras` e os `Principais Termos`; mostrar a `Progressão Histórica da Marca`.

Figura 19 - Tela com o resultado da busca




Fonte: elaborado pelo autor.

A próxima informação apresentada na tela é o resultado da busca com a marca, a categoria e o resultado da análise de sentimentos, que mostra para o usuário o percentual total dos termos positivos e negativos que possuem maior ocorrência na busca. Também são apresentadas informações relacionadas ao termo em uma `Nuvem de Palavras`, que é uma montagem visual com os termos mais utilizados. No lado direito da tela são apresentados os `Principais Termos` da busca, contendo a quantidade de ocorrências e o percentual de positividade e negatividade de cada um. Por fim, na parte inferior da tela é exibida a `Progressão Histórica da Marca`, através de um gráfico contendo a progressão da positividade da marca ao longo do tempo, para cada categoria.

## 3.4 ANÁLISE DOS RESULTADOS

Nesta seção são apresentados os testes realizados e os resultados do desenvolvimento do sistema. Na seção 3.4.1 são apresentados os testes realizados durante o desenvolvimento do sistema, enquanto que a seção 3.4.2 mostra os resultados gerados pela execução do sistema. Por fim, na seção 3.4.3 é apresentada a comparação deste trabalho com os trabalhos correlatos.

### 3.4.1 Testes realizados

Conforme demonstrado na seção 3.2.2, o desenvolvimento deste sistema foi definido em três partes chaves: a análise de dados, a filtragem de dados e a mineração de dados. Com isso, foram estudadas as abordagens disponíveis mais adequadas para o desenvolvimento do sistema proposto.

Para a análise de sentimentos foram identificadas duas bibliotecas, o SentiWordNet e o WordNet-Affect (STRAPPARAVA; VALITUTTI, 2004). Nos testes foi observado que o SentiWordNet engloba um número maior de termos, bem como possui uma distribuição que facilita o uso no sistema e, por este motivo, foi escolhido para ser aplicado neste sistema. O SentiWordNet é detalhado na seção 3.3.2.2.

Para a filtragem de dados foi visto que os métodos de lematização, StopWords e de remoção dos caracteres HTML seriam suficientes para o desenvolvimento deste trabalho. A mineração de dados, entretanto, foi modificada durante o processo. Num estudo inicial havia sido definido que seria utilizada a busca pelos dados mais recentes através do próprio site de pesquisa do Twitter. Entretanto, foi verificado que a quantidade de postagens retornadas é limitada pelo próprio Twitter, sendo necessário alterar esta abordagem.

Para resolver este problema foram encontradas três maneiras distintas: através de um *widget*, utilizando a API do Twitter ou utilizando a busca avançada com alguma alteração. Um *widget* é um componente disponibilizado por sites para que seja incorporado seu conteúdo em outros sites. Neste caso, o objetivo foi de utilizá-lo para retirar mais dados de uma busca, pois o limite do Twitter não se aplica a esta página. Entretanto, não foi possível tornar a busca dinâmica pelo *widget*, e por este motivo esta opção foi descartada.

A segunda alternativa foi de utilizar a API do Twitter, que é o método oficial disponibilizado pelo Twitter para a obtenção de dados a partir da plataforma. Para isso, foram realizados testes chamando esta API através do PHP, porém não se obteve sucesso, pois a API retornava uma mensagem informando um erro de autenticação. Com base nesta mensagem de erro e na documentação da API, foi verificado que apenas requisições partindo de endereços

utilizando o Protocolo de Transferência de Hipertexto Seguro (HTTPS) seriam aceitas. O HTPPS garante a identidade do site que faz a requisição, sendo disponibilizado para sites com domínio registrado e verificado, e, como o sistema não pode ter este protocolo, esta opção também foi descartada.

Por fim, a alternativa restante para a mineração de dados utilizada neste trabalho, foi a busca avançada do Twitter. Entretanto, para superar o problema de a busca apenas retornar no máximo 25 postagens por vez foram feitas duas alterações. A primeira visou aumentar as buscas de um dia, adicionando a busca tanto na aba `Últimas` quanto na aba `Em Destaque`, duplicando assim o número de postagens. A Figura 20 apresenta as abas disponíveis para a pesquisa, logo abaixo da palavra “microsoft”.

Figura 20 – Abas da pesquisa avançada do Twitter

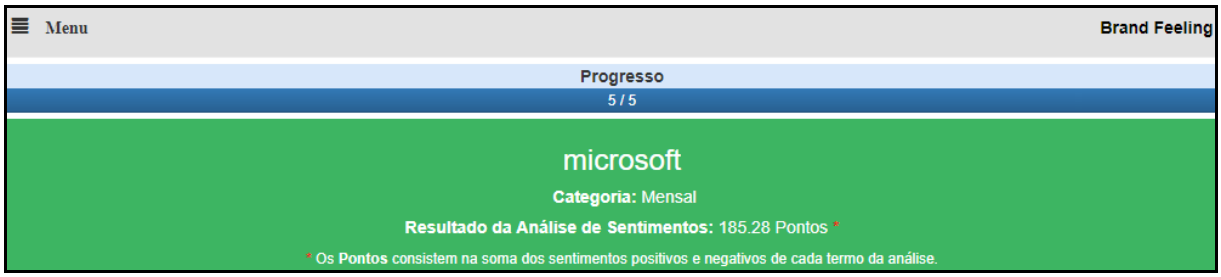

Fonte: Twitter.

A segunda alteração foi a adição da busca por categorias. Embora a busca do Twitter retorne apenas 50 postagens, ela permite um filtro adicional por um intervalo de datas. Então, para permitir uma análise mais precisa de uma marca foram definidas 4 categorias: diária, onde é realizada a busca apenas dos dados do dia; semanal, onde a pesquisa acontece nos últimos 7 dias, retornando em média 350 postagens; quinzenal, pesquisa feita nos últimos 15 dias, duplicando o número de postagens; mensal, sendo esta a que avalia o maior número de postagens, pois efetua a busca de dados dos últimos 30 dias. A adição das categorias possibilitou não só uma visão da progressão histórica da marca, permitindo que o usuário tenha mais opções, mas também deixou os resultados mais consistentes.

No final do desenvolvimento foi alterado o método para apresentar os resultados aos usuários, visando deixar claro para o usuário qual o sentimento do público sobre cada marca.

Originalmente foi utilizada uma pontuação, conforme mostrado na Figura 21 onde a busca obteve 185.28 pontos, esta pontuação consistia numa multiplicação do valor de cada termo no SentiWordNet pelo número de ocorrências, e depois da soma dos valores de todos os termos de uma busca, porém esta pontuação era vaga, não deixando claro o quão positivo ou negativo era a visão do público sobre uma marca. Foi então criado um novo método, onde é apresentado como resultado da busca a porcentagem de positividade de uma marca, assim como dos principais termos desta busca, deixando claro se uma marca possui uma avaliação positiva ou negativa na visão do público.

Figura 21 – Antigo método para apresentar resultados




Fonte: elaborado pelo autor.

### 3.4.2 Resultados do sistema

Durante o desenvolvimento do sistema, um dos objetivos foi o de realizar a análise sobre o maior número de termos no menor tempo possível. Para isso foram empregadas as técnicas de programação concorrente demonstradas na seção 3.3.3. Nesta seção são apresentados então os resultados do sistema.

Na Tabela 6 são detalhados os resultados das pesquisas para cada categoria disponível da marca “Microsoft”. Os critérios a serem analisados na execução são: número de termos, isto é, o número de palavras distintas geradas que possuem polaridade positivo-negativo; número de sites visitados nesta busca, onde cada site corresponde a busca avançada do Twitter, junto de um intervalo de datas; número de postagens do Twitter analisadas; termo não neutro com maior recorrência e o número de vezes em que é citado; e o grau de positividade de cada pesquisa, onde é calculado sobre o número de termos da pesquisa a porcentagem dos termos positivos e o tempo de execução em segundos de cada uma das buscas.

Tabela 6 – Resultados gerados pelo sistema na busca pelo termo Microsoft

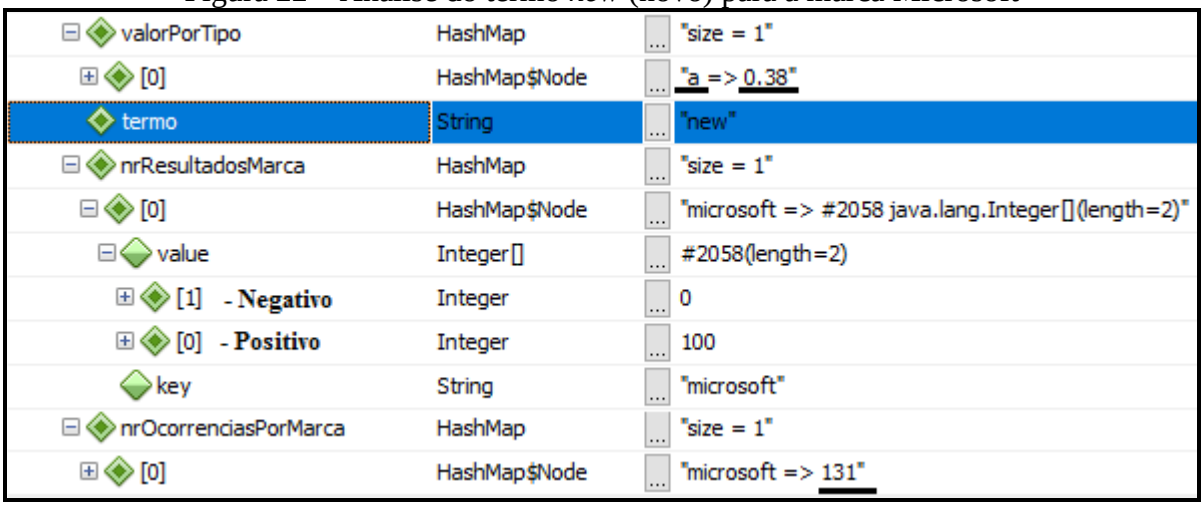
| Categorias | Número de Termos | Número de Sites | Número de Postagens | Termo com maior recorrência | Positividade | Tempo em segundos (S) |
|---|---|---|---|---|---|---|
| **Diária** | 65 | 2 | 48 | add – 8 vezes | 73 % | 5,3 S |
| **Semanal** | 284 | 16 | 402 | new – 50 vezes | 62 % | 7,8 S |
| **Quinzenal** | 456 | 32 | 740 | new – 86 vezes | 61 % | 11,6 S |
| **Mensal** | 669 | 62 | 1316 | new – 163 vezes | 60 % | 18,2 S |

Fonte: elaborado pelo autor.

Com os resultados pode-se concluir que as categorias estão organizadas em uma ordem crescente, isto é, ao avançar nas categorias temos um maior número de sites, postagens e termos avaliados, entretanto o tempo de execução da pesquisa não escala na mesma medida. Por conta da programação concorrente, o tempo de resposta do sistema é um tempo satisfatório para o usuário, independentemente da categoria escolhida.

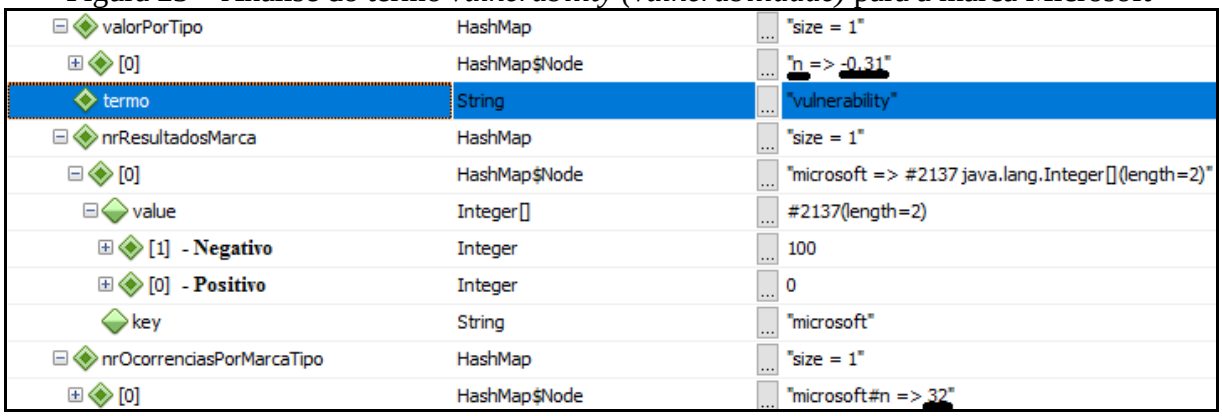
Além disso, verificou-se que, apesar da categoria mais básica retornar apenas 65 termos com polaridade positivo-negativo, a pesquisa na categoria mensal retorna um número superior de termos com polaridade, deixando a análise mais concisa. Como exemplo destes termos analisados pode-se observar a Figura 22, onde é apresentado o resultado da análise do termo *new* (novo), que possui polaridade positiva, e foi citado 131 vezes. Na figura é representado na variável `nrOcorrenciasPorMarca`, na posição 0. A mesma situação é demonstrada na Figura 23, que contém o termo *vulnerability* (vulnerabilidade). Neste caso ele foi citado 32 vezes, mas possui uma polaridade negativa, isto é demonstrado na variável `nrResultadosMarca` na posição 0. Estes exemplos mostram que o sistema foi capaz de agrupar os termos e determinar com base nisto sua respectiva polaridade.

Figura 22 – Análise do termo *new* (novo) para a marca Microsoft




Fonte: elaborado pelo autor.

Figura 23 – Análise do termo *vulnerability (vulnerabilidade)* para a marca Microsoft




Fonte: elaborado pelo autor.

### 3.4.3 Comparação entre o sistema e os trabalhos correlatos

O Quadro 13 apresenta uma comparação entre o sistema desenvolvido e os trabalhos correlatos.

Quadro 13 – Comparação entre o sistema e os trabalhos correlatos

| Trabalhos / Características | Trabalho Desenvolvido | Thelwall, Buckley e Paltoglou (2012) | Santos (2010) | Paltoglou e Thelwall (2012) |
|---|---|---|---|---|
| Objetivo do trabalho | Analisar o sentimento de usuários em relação a uma marca. | Verificar o desempenho do algoritmo SentiStrength | Analisar opinião na web | Validar uma abordagem baseada num dicionário de expressões |
| Geração de dados | Mineração de dados do Twitter | Extraídos de redes sociais | Utiliza um *crawler* | Extraídos do Twitter, MySpace e Digg |
| Utiliza algoritmo de aprendizado de máquina | Não | Não | SVM | Classificador não supervisionado criado pelos autores |
| Algoritmo de análise de sentimentos utilizado | SentiWordNet 3.0 | *SentiStrength* 2.0 | Algoritmo próprio | Classificador baseado em um dicionário de expressões pré-definidas |
| Interface apresenta resultados dos termos pesquisados aos usuários | Sim | Não | Sim | Não |

Fonte: elaborado pelo autor.

É possível visualizar uma finalidade em comum entre os trabalhos correlatos e o sistema desenvolvido, que é efetuar a análise de sentimentos para seus respectivos propósitos. Um segundo fator em comum é quanto a geração de dados, todos utilizam de alguma forma a

extração de redes sociais, porém apenas o sistema desenvolvido e o trabalho de Santos (2010) fazem a mineração em tempo real dos dados.

Santos (2010) e Paltoglou e Thelwall (2012) utilizam abordagens de aprendizado de máquina como base para a construção de seu algoritmo de análise de sentimentos, já Thelwall, Buckley e Paltoglou (2012) utilizam o algoritmo SentiStrength, que não se baseia no aprendizado de máquina. Para o sistema desenvolvido a utilização do aprendizado de máquina não foi necessária no momento, ficando apenas como extensão.

Apesar de todos os trabalhos terem como objetivo principal a análise de sentimentos, cada um utilizou um algoritmo diferente. Santos (2010) e Paltoglou e Thelwall (2012) desenvolveram abordagens próprias, enquanto que no trabalho de Thelwall, Buckley e Paltoglou (2012) foi utilizado o *SentiStrength*, e no sistema desenvolvido para este trabalho foi feita a escolha pelo uso do SentiWordNet. Por fim, apenas o trabalho desenvolvido e o trabalho de Santos (2010) criaram uma interface para a utilização por usuários.

# 4 CONCLUSÕES

Durante o desenvolvimento deste trabalho, criou-se um sistema web que analisa os sentimentos dos usuários em relação a uma determinada marca. Para isso, foi desenvolvida uma interface web onde o usuário pode realizar a entrada dos dados, selecionar uma categoria ou uma segunda marca, e obter como resultado a porcentagem de polaridade da marca na rede social Twitter, bem como visualizar pesquisas antigas e o histórico das marcas. O sistema obteve sucesso em realizar a busca dos dados do Twitter, bem como filtrar os dados e realizar a análise de sentimentos sobre esta pesquisa. Portanto, tanto o objetivo principal do trabalho, quanto seus objetivos específicos foram atingidos, fornecendo ao usuário comum uma ferramenta para verificar a reputação de uma marca perante ao público.

Contudo, foi encontrada uma dificuldade na geração dos dados, onde embora fosse possível recuperá-los, o número disponível pelo Twitter era limitado, não satisfazendo a análise. Para superar esta dificuldade foi alterada a busca de dados, adicionando categorias, que através da busca em um intervalo de dias solucionaram este problema. Também foi necessário incrementar a análise de sentimentos, onde inicialmente estava previsto apenas utilizar os valores de cada termo definidos pelo SentiWordNet. Entretanto, por conta da biblioteca escolhida, foi necessário adicionar etapas prévias a esta análise onde foi realizada a lematização das palavras e a definição da classe gramatical ou POS de cada uma. Uma vez realizadas as etapas anteriores, foi possível extrair com sucesso os valores de cada palavra na busca.

Com a adição das categorias o sistema se mostrou mais confiável na análise de sentimentos, não só realizando a análise com um número maior de termos, mas também fornecendo um mecanismo para medir a progressão histórica de uma marca. Também foi notado um resultado acima do esperado no que diz respeito ao tempo de realização de todo o processo, mantendo um tempo máximo de pesquisa de até 30 segundos.

Como conclusão acredita-se que este trabalho contribuiu para a área de análise de sentimentos, disponibilizando uma aplicação que não só beneficia o usuário comum, mas também evidencia o potencial da análise de sentimentos para um público diferente do acadêmico. Além disso, a integração entre as tecnologias para o desenvolvimento deste trabalho pode ser utilizada como base para outros sistemas de análise de sentimentos, visto que o processo de análise documentado pode ser empregado para outras finalidades, contribuindo assim tanto para o meio acadêmico como tecnológico.

No âmbito social este trabalho se torna relevante, pois, a partir dele, foi criado um sistema para que o usuário comum possa verificar qual o sentimento do público sobre uma ou mais marcas, inclusive se esta marca está em ascensão ou em queda na opinião geral. Esta informação deve permitir que os usuários decidam se devem confiar ou não seu dinheiro, tempo ou até mesmo esforço em uma determinada marca.

## 4.1 EXTENSÕES

O sistema desenvolvido nesta monografia atingiu os objetivos definidos. Entretanto existem pontos que podem ser melhorados e novas funcionalidades que podem ser adicionadas ao mesmo. São eles:

a) adicionar um algoritmo de aprendizado de máquina ao `Módulo Servidor`, visando otimizar a performance do mesmo;
b) criar na filtragem de dados um mecanismo para efetuar o tratamento de palavras mal escritas, visando encontrar a palavra correta para a análise;
c) efetuar a mineração de dados de outras redes sociais, como por exemplo o Facebook;
d) disponibilizar no `Módulo Web` uma paginação na tela inicial, deixando o usuário visualizar um número maior de pesquisas recentes e de marcas no histórico;
e) estudar uma abordagem para tratar a utilização de figuras de linguagem, como sarcasmo e ironia, no algoritmo de análise de sentimentos;
f) implementar as etapas de análise semântica e análise pragmática do PLN, visando extrair o significado de cada frase para otimizar a análise de sentimentos.

## REFERÊNCIAS

AGARWAL, Apoorv et al. Sentiment analysis of twitter data. In: WORKSHOP ON LANGUAGES IN SOCIAL MEDIA, 11., 2011, Portland. **Proceedings…** New York: Association for Computational Linguistics, 2011. p. 30-38.

ASUR, Sitaram; HUBERMAN, Bernardo A. Predicting the future with social media. In: IEEE/WIC/ACM INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE AND INTELLIGENT AGENT, 10., 2010, Toronto. **Proceedings...** Palo Alto: IEEE Computer Society, 2010. p. 492-499.

BACCIANELLA, Stefano; ESULI, Andrea; SEBASTIANI, Fabrizio. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In: LREC, 10, 2010, Pisa. **Proceedings...** Valleta: Istituto di Scienza e Tecnologie dell'Informazione, 2010. p. 2200-2204.

BENEVENUTO, Fabrício; RIBEIRO, Filipe; ARAÚJO, Matheus. **Métodos para Análise de Sentimentos em mídias sociais**. [Minas Gerais], 2015. Disponível em: < www.dcc.ufmg.br/~fabricio/download/webmedia-short-course.pdf >. Acesso em: 22 mai. 2017.

FERSINI, Elisabetta et al. **Sentiment Analysis in Social Networks**. Cambridge: Todd Green, 2016.

FINLAYSON, Mark Alan. Java Libraries for Accessing the Princeton Wordnet: Comparison and Evaluation. In: INTERNATIONAL GLOBAL WORDNET CONFERENCE, 7, 2014, Tartu. **Anais…** Cambridge: Computer Science and Artificial Intelligence Laboratory, 2014. p. 78-85.

FORNACCIARI, Paolo; MORDONINI, Monica; TOMAUIOLO, Michele. Social network and sentiment analysis on Twitter: towards a combined approach. In: INTERNATIONAL WORKSHOP ON KNOWLEDGE DISCOVERY ON THE WEB, 1., 2015, Parma. **Proceedings…** Parma: Università degli Studi di Parma, 2015. p.1.

GO, Alec; BHAYANI, Richa; HUANG, Lei. Twitter sentiment classification using distant supervision. **Processing**, Stanford, v. 1, p. 1-6, 2009.

GOLDSCHMIDT, Ronaldo; PASSOS, Emmanuel. **Data Mining.** 2. ed. Rio de Janeiro: Elsevier, 2015.

HAN, Jiawei; PEI, Jian; KAMBER, Micheline. **Data Mining:** Concepts and Techniques. 3. ed. Waltham: Elsevier, 2011.

HUTCHINSON, Andrew. **Twitter Q1 Numbers: Active Users Up, Increasing Advertiser Focus**. 2017. Disponível em: <http://www.socialmediatoday.com/social-networks/twitter-q1-numbers-active-users-increasing-advertiser-focus>. Acesso em: 12 jul. 2017.

JURŠIC, Matjaz et al. Lemmagen: Multilingual lemmatisation with induced ripple-down rules. **Journal of Universal Computer Science**, [S.l.], v. 16, n. 9, p. 1190-1214, 2010.

LIDDY, Elizabeth D. Natural Language Processing. In: Marcel Decker. **Encyclopedia of Library and Information Science**. 2. ed. Nova York: Marcel Decker, 2001.

MANGUKIYA, Piyush. **Social media by the numbers.** [S.l.], 2016. Disponível em: <http://www.huffingtonpost.com/piyush-mangukiya/social-media-by-the-numbe_b_9757926.html>. Acesso em: 04 mar. 2017.

MEMON, Nasrullah et al (Ed.). **Data Mining for Social Network Data:** Annals of Information Systems. 12. ed. [S.l.]: Springer, 2010.

MILLER, George A. WordNet: a lexical database for English. **Communications of the ACM**, Princeton, v. 38, n. 11, p. 39-41, Nov. 1995.

MOHAMMAD, Saif M. Sentiment analysis: Detecting valence, emotions, and other affectual states from text. In: MEISELMAN, Herbert. **Emotion Measurement.** 1. Ed. [S.l.]: MEISELMAN, Herbert, 2015, p. 201-238.

MÜLLER, Daniel. **Processamento de Linguagem Natural**. 2003. 11 f. Relatório para a disciplina CMP187 – Mentes e Maquinas (Programa de Pós Graduação em Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre.

NICHOLLS, Chris; SONG, Fei. Comparison of Feature Selection Methods for Sentiment Analysis. In: CANADIAN CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1., 2010, Ontario. **Proceedings…** Berlin: Springer, 2010. V.6085. 286-289.

PALTOGLOU, Georgios; THELWALL, Mike. Twitter, MySpace, Digg: Unsupervised sentiment analysis in social media. **ACM Transactions on Intelligent Systems and Technology (TIST)**, Wolverhampton, v. 3, n. 4, p. 66, Setembro. 2012.

PENFIELD JR, Paul. Principle of Maximum Entropy: Simple Form. **Massachusetts: Massachusetts Institute of Technology**, Massachusetts, v. 1, p.80-84, Março. 2003.

PEREIRA, Silvio. **Processamento de Linguagem Natural**. [São Paulo], 2013. Disponível em: <http://jeiks.net/wp-content/uploads/2013/10/Processamento_de_Linguagem_Natural.pdf >. Acesso em: 04 jul. 2017.

SANTOS, Leandro Matioli. **Protótipo para mineração de opinião em redes sociais**: estudo de casos selecionados usando o twitter. 2010. 103 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Universidade Federal de Lavras, Lavras.

SCHAPIRE, Robert E. Explaining adaboost. In: SCHÖLKOPF, Bernhard; LUO, Zhiyuan; VOVK, Vladimir. **Empirical Inference.** Berlin: SCHÖLKOPF, Bernhard; LUO, Zhiyuan; VOVK, Vladimir, 2013. p. 37-52.

SCHÜTZE, Hinrich. Introduction to information retrieval. In: INTERNATIONAL COMMUNICATION OF ASSOCIATION FOR COMPUTING MACHINERY CONFERENCE, 1., 2008, **Proceedings…** Cambridge: Cambridge University Press, 2009. 26.

STRAPPARAVA, Carlo; VALITUTTI, Alessandro. WordNet Affect: an Affective Extension of WordNet. In: International Conference on Language Resources and Evaluation, n. 1, 2004, Lisboa. **Proceedings…** Trento: Istituto per la Ricerca Scientifica e Tecnologica, 2004. 4. 1083-1086.

THELWALL, Mike; BUCKLEY, Kevan; PALTOGLOU, Georgios. Sentiment strength detection for the social web. **Journal of the American Society for Information Science and Technology**, [S.l.], v. 63, n. 1, p. 163-173, 2012.

TOUTANOVA, Kristina et al. Feature-rich part-of-speech tagging with a cyclic dependency network. In: CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS ON HUMAN LANGUAGE TECHNOLOGY, 2003, Edmonton. **Proceedings…** Stanford: Association for Computational Linguistics, 2003.v. 1, p. 252-259.

VOUTILAINEN, Atro. Part-of-speech tagging. In: **The Oxford handbook of computational linguistics**,1. ed. [S.l.]: MIKTOV, Ruslan , 2005. p. 219-232.

WITTEN, Ian H. et al. **Data Mining:** Practical machine learning tools and techniques. 4. ed. [S.l.]: Todd Green, 2016.

WRIGHT, Alex. **Mining the Web for Feelings, Not Facts**. 2009. Disponível em: < http://www.nytimes.com/2009/08/24/technology/internet/24emotion.html>. Acesso em: 13 mar. 2017.

## APÊNDICE A – Carregamento do dicionário do SentiWordNet

Neste apêndice é demonstrado como é efetuado o carregamento no Java do dicionário do SentiWordNet. Esta implementação foi baseada no código disponibilizado no site do SentiWordNet, que está disponível em: http://sentiwordnet.isti.cnr.it/code/SentiWordNetDemoCode.java. O Quadro 14 demonstra a inicialização da classe `Dicionario`, onde foi realizada a implementação. Nesta classe é carregado o arquivo do SentiWordNet, transferindo os dados do arquivo para um *buffer*. No método `popularDicionario` é realizado então a inicialização das variáveis que serão utilizadas no processo bem como a transferência dos dados do SentiWordNet.

Quadro 14 – Inicialização da classe `dicionário`

```
public class Dicionario {
    private HashMap<String, HashMap<String, Double>> dictionary;
    private final String pathToSWN = "C:/Users/junio/Google Drive/Furb/
        9° Semestre/TCC II/ BrandFeeling/Servidor/lib/
        SentiWordNet_3.0.0_20130122.txt";

    public Dicionario() {
        this.dictionary = new HashMap<String, HashMap<String, Double>>();
        this.popularDicionario();
    }

    public HashMap<String, HashMap<String, Double>> getDictionary() {
        return dictionary;
    }

    public void popularDicionario(){
        BufferedReader csv = null;
        int lineNumber = 0,synTermRank = 0;
        Double synsetScore,score,sum;
        String line,wordTypeMarker,synTerm,word;
        String[] data,synTermsSplit,synTermAndRank,palavras;
        HashMap<String, HashMap<Integer, Double>> tempDictionary = new
HashMap<String, HashMap<Integer, Double>>();
        Map<Integer, Double> synSetScoreMap;

        csv = new BufferedReader(new FileReader(pathToSWN));
```

Fonte: elaborado pelo autor.

Na sequência é realizado o carregamento básico dos dados do arquivo para o Java. O Quadro 15 ilustra onde é realizada a leitura de cada uma das linhas do arquivo que está no *buffer*. Primeiramente são efetuadas verificações, para descartar possíveis linhas inválidas para a análise, depois é calculado o valor das palavras nesta linha, subtraindo a pontuação negativa da positiva, e, por fim, são percorridas as palavras contidas nesta linha, adicionando cada uma a um dicionário temporário.

Quadro 15 – Carregamento dos dados do arquivo para o Java

```
while ((line = csv.readLine()) != null) {
        lineNumber++;

        if (line.trim().startsWith("#")) { continue;}
        data = line.split("\t");
        wordTypeMarker = data[0];


        synsetScore = Double.parseDouble(data[2]) -
                Double.parseDouble(data[3]);

        if(synsetScore == 0.00){ continue; }

        synTermsSplit = data[4].split(" ");

        for (String synTermSplit : synTermsSplit) {

          synTermAndRank = synTermSplit.split("#");
          synTerm = synTermAndRank[0] + "#" + wordTypeMarker;

          synTermRank = Integer.parseInt(synTermAndRank[1]);

          if (!tempDictionary.containsKey(synTerm)) {
            tempDictionary.put(synTerm, new HashMap<Integer, Double>());
          }

          tempDictionary.get(synTerm).put(synTermRank,synsetScore);
        }
}
```

Fonte: elaborado pelo autor.

Uma vez passado o conteúdo do arquivo para uma estrutura do Java é então definido o dicionário final que será utilizado no sistema, conforme demonstrado no

Quadro 16. Nesta fase é percorrido o dicionário temporário onde é efetuado o cálculo do valor de cada termo. Este valor é calculado com base no ranking de cada termo definido pelo SentiWordNet de acordo com o uso mais comum da palavra na linguagem.

Após o cálculo do valor final é então adicionado o termo e o valor ao dicionário final. Nesta fase, a implementação se difere da original por dois motivos. O primeiro motivo é que a implementação do sistema utiliza o valor total de cada termo dividido pelo ranking atribuído ao mesmo, enquanto que a do exemplo utilizava um cálculo diferente. A outra diferença é que o dicionário final do sistema utiliza apenas o termo como índice e o armazena juntamente com os valores de cada classe. Esta alteração foi feita visando otimizar a análise dos termos.

Quadro 16 – Definição do dicionário final do SentiWordNet

```
for (Map.Entry<String, HashMap<Integer, Double>> entry :
     tempDictionary.entrySet()) {

        word = entry.getKey();
        synSetScoreMap = entry.getValue();
        score = 0.0;
        sum = 0.0;

        for (Map.Entry<Integer, Double> setScore :
             synSetScoreMap.entrySet()) {
          score += setScore.getValue() / (double) setScore.getKey();
        }

        score = Math.round(score * 100.0) / 100.0;
        palavras = word.split("#");

        if(palavras[0].length() == 1){ continue; }

        if (!this.dictionary.containsKey(palavras[0])) {
           this.dictionary.put(palavras[0], new HashMap<>());
        }

        this.dictionary.get(palavras[0]).put(palavras[1], score);
}
csv.close();
```

Fonte: elaborado pelo autor.

**ANEXO A – Representação da biblioteca do SentiWordNet**

Neste anexo é demonstrado parte do arquivo da biblioteca do SentiWordNet onde as palavras foram adaptadas e colocadas na Tabela 7 para melhor visualização. Esta tabela possui as seguintes colunas: POS, que indica a classe gramatical dos termos; ID, código dos termos; PosScore e NegScore, pontuação positiva e negativa dos termos; SynsetTerms, quais são os termos compatíveis com este significado; Gloss, o significado que o(s) termo(s) representa(m).

SentiWordNet v3.0.0 (1 June 2010)


For any information about SentiWordNet:
Web: http://sentiwordnet.isti.cnr.it

Data format.

SentiWordNet v3.0 is based on WordNet version 3.0.
WordNet website: http://wordnet.princeton.edu/

The pair (POS,ID) uniquely identifies a WordNet (3.0) synset.
The values PosScore and NegScore are the positivity and negativity score assigned by SentiWordNet to the synset.
The objectivity score can be calculated as:
ObjScore = 1 - (PosScore + NegScore)
SynsetTerms column reports the terms, with sense number, belonging to the synset (separated by spaces).

Tabela 7 - Biblioteca do SentiWordNet

| POS | ID | PosScore | NegScore | SynsetTerms | Gloss |
|---|---|---|---|---|---|
| a | 00001740 | 0.125 | 0.125 | dorsal#2 abaxial#1 | bursting open with force, as do some ripe seed vessels |
| a | 00005473 | 0.75 | 0 | direct#10 | lacking compromising or mitigating elements; exact; "the direct opposite" |
| a | 000005599 | 0.5 | 0.5 | unquestioning#2 implicit#2 | being without doubt or reserve; "implicit trust" |
| a | 00004296 | 0 | 0 | last#5 | occurring at the time of death; "his last words"; "the last rites" |
| a | 00257462 | 0 | 0.75 | unbound#1 | not secured within a cover; "an unbound book" |
| n | 00034479 | 0 | 0 | thing#2 | an action; "how could you do such a thing?" |
| n | 00035891 | 0 | 0.375 | alienation#4 | the action of alienating; the action of causing to become unfriendly; "his behavior alienated the other students" |
| n | 00037006 | 0.625 | 0 | masterpiece#2 | an outstanding achievement |
| n | 00042311 | 0 | 0 | causing#1 causation#1 | the act of causing something to happen |
| n | 00070807 | 0 | 0.75 | partial_breach#1 | a breach that does not destroy the value of the contract but can give rise to a claim for damages |
| n | 00034479 | 0 | 0 | thing#2 | an action; "how could you do such a thing?" |
| r | 00416430 | 0.25 | 0 | pacifistically#1 | in a pacifistic manner; "the pacifistically inclined liberals" |
| r | 00417299 | 0.25 | 0 | parentally#1 | in a parental manner |
| r | 00418223 | 0.5 | 0.5 | pitiably#1 pathetically#1 | in a manner arousing sympathy and compassion; "the sick child cried pathetically" |
| r | 00428493 | 0.25 | 0 | powerlessly#1 | in a powerless manner |
| r | 00433120 | 0 | 0.5 | unprofitably#1 profitlessly#1 gainlessly#1 | without gain or profit |
| v | 00487554 | 0.125 | 0 | polarize#1 polarise#1 | cause to vibrate in a definite pattern; "polarize light waves" |
| v | 00521478 | 0 | 0 | green#1 | turn or become green; "The trees are greening" |
| v | 00544549 | 0.375 | 0.5 | raise#12 lift#10 elevate#3 | raise in rank or condition; "The new law lifted many people from poverty" |
| v | 00548153 | 0.5 | 0 | return#9 come_back#1 | be restored; "Her old vigor returned" |
| v | 02665617 | 0.125 | 0.5 | look_like#1 | bear a physical resemblance to; "She looks like her mother" |

Fonte: adaptado de SentiWordNet.