UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

EXPLORATOR: UMA FERRAMENTA PARA MINERAÇÃO DE OPINIÕES VEICULADAS NO TWITTER

DIEGO SANTOS LUIZ

BLUMENAU
2013

2013/2-04

DIEGO SANTOS LUIZ

EXPLORATOR: UMA FERRAMENTA PARA MINERAÇÃO DE OPINIÕES VEICULADAS NO TWITTER

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Aurélio Faustino Hoppe, Mestre - Orientador

BLUMENAU
2013

2013/2-04

EXPLORATOR: UMA FERRAMENTA PARA MINERAÇÃO DE OPINIÕES VEICULADAS NO TWITTER

Por

DIEGO SANTOS LUIZ

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

Presidente:	Prof. Aurélio Faustino Hoppe, Mestre – Orientador, FURB
Membro:	Prof. Alexander Roberto Valdameri, Mestre – FURB
Membro:	Prof. Cláudio Ratke, Mestre – FURB

Blumenau, 9 de dezembro de 2013

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

A minha família pela compreensão nos momentos que não estive presente.

Em especial, a minha namorada Jaqueline A. Michalack, pela atenção, compreensão, carinho e paciência no desenvolvimento deste trabalho.

Ao meu orientador, Aurélio Faustino Hoppe, por ter acreditado na conclusão deste trabalho, fornecendo conhecimento e todo o auxílio que necessitei durante o desenvolvimento do projeto.

Tenha coragem de seguir o que seu coração e a sua intuição dizem. Eles já sabem o que você realmente deseja. Todo resto é secundário.

Steve Jobs

RESUMO

Este trabalho apresenta uma ferramenta capaz de realizar a mineração de textos do Twitter. Utilizando técnicas de mineração de textos é possível pré-processar, indexar e classificar as informações textuais de acordo com o interesse do usuário. Os resultados demonstram que a ferramenta obteve em seus testes uma taxa de acerto acima de 80%.

Palavras-chave: Mineração de textos. Descoberta de conhecimento. Base de dados textuais.

ABSTRACT

This work presents a tool able to perform text mining Twitter. Using text mining techniques can preprocess, classify and index the textual information according to user interest. The results show that the tool got on their tests with an accuracy rate above 80%.

Key-words: Text mining. Knowledge discovery. Textual database.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API – *Application Programming Interface*

DCT – Descoberta de Conhecimento em Texto

HTTP – *Hypertext Transfer Protocol*

IDE – *Integrated Development Environment*

LINQ – *Language-Integrated Query*

MD – Mineração de dados

MT – Mineração de textos

MVC – *Model View Controller*

RF – Requisito Funcional

RNF – Requisito Não Funcional

SVM – *Support Vector Machines*

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, as pessoas estão procurando por informações resumidas e relevantes para auxiliá-las em suas decisões. Todavia, estas informações estão sendo encontradas em *blogs*, rede sociais, fóruns, *website* (SANTOS, 2010, p. 9). Entretanto, nos últimos anos ocorreu um aumento significativo do volume de dados disponibilizados nestes meios de comunicação, fato que ultrapassa a habilidade técnica e a capacidade humana na sua interpretação (ARANHA, 2007, p. 16).

Segundo Goldschmidt e Passos (2005, p. 18), a análise de grandes quantidades de dados é inviável sem o auxílio de ferramentas computacionais apropriadas. Portanto, torna-se imprescindível o desenvolvimento de ferramentas que auxiliem o homem, de forma automática e inteligente, na tarefa de analisar, interpretar e relacionar esses dados para que se possa desenvolver e selecionar estratégias de ação em cada contexto de aplicação.

A partir desta necessidade, sugiram as ferramentas de *Text Mining* (MT - Mineração de Textos). Esta tecnologia permite recuperar informações, extrair dados, resumir documentos, descobrir padrões, associações e regras e realizar análises qualitativas ou quantitativas em documentos de texto (ARANHA e PASSOS, 2006, p. 2).

Segundo Monteiro, Gomes e Oliveira (2006, p. 78), a mineração de textos representa a procura por padrões em um texto em linguagem natural e pode ser definido como o processo de análise do texto para extrair informações dele para um propósito em particular. Por exemplo, os usuários do Twitter propendem a expressar suas opiniões livremente. Contudo, tais informações possibilitam as organizações monitorarem os pontos em que a mesma não está prestando ou fornecendo um produto adequado ao seu consumidor.

Segundo Santos (2010), o Twitter é um serviço de *micro-blogging* que permite postagem de mensagens de até 140 caracteres. O Twitter possui uma base rica de informações de fácil acesso, se tornando um ótimo local para se obter opiniões sobre um determinado assunto.

Diante do exposto, este trabalho apresenta o desenvolvimento de uma ferramenta que implementa algumas técnicas de MT para apoiar a extração e a visualização de informações veiculadas no Twitter.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é o desenvolvimento de uma ferramenta que fará a seleção, a classificação e a apresentação de informações extraídas do Twitter.

Os objetivos específicos do trabalho são:

a) estabelecer conexão com a base de dados do Twitter;
b) classificar os *tweets* utilizando as palavras positivas e negativas informadas pelo usuário.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está disposto em quatro capítulos. O primeiro capítulo contém a introdução, objetivo do trabalho e estrutura do trabalho.

O segundo capítulo descreve a fundamentação teórica nas quais são apresentadas conceitos e técnicas exploradas no desenvolvimento deste trabalho e, ao final, são apresentados os trabalhos correlatos.

O terceiro capítulo trata do desenvolvimento da ferramenta, onde são relacionados seus requisitos, a especificação e diagramas. Também é descrito a implementação apresentando técnicas e ferramentas utilizadas, a operacionalidade e, por fim, são apresentados os resultados obtidos.

O quarto capítulo apresenta as conclusões do presente trabalho, limitações e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritos os conteúdos pesquisados para desenvolvimento do trabalho. A seção 2.1 descreve o que é mineração de dados. A seção 2.2 descreve o que é MT, e fornece alguns conceitos. A seção 2.2.1 descreve a etapa de coleta dos textos. A seção 2.1.2 descreve a etapa de pré-processamento dos textos. A seção 2.2.3 descreve a técnica de classificação. A seção 2.1.4 descreve a etapa de análise da informação. A seção 2.3 são apresentado os trabalhos relacionados.

2.1 MINERAÇÃO DE DADOS

A mineração de dados (MD) é o processo de descoberta de padrões independentemente da quantidade de dados a serem analisados. Este processo utiliza-se de técnicas para reconhecimento de padrões, técnicas de estatística e matemática (LAROSE, 2005, p. 2).

Segundo Castanheira (2008, p. 18), mineração de dados é a extração de informações úteis e previamente desconhecidas de grandes bancos de dados, que conforme o propósito pode descobrir comportamentos que seriam dificilmente identificados por especialistas.

As principais técnicas de mineração de dados são:

a) agrupamento: o objetivo do agrupamento (*clustering*) é a junção de dados semelhantes em um mesmo grupo e dados distintos em outros grupos. O agrupamento normalmente é utilizado quando não se sabe quantas classes possui o arquivo a ser minerado (VIERA, 2011, p. 32);
b) associação: segundo Sidney (2010, p. 19), a associação tem como objetivo caracterizar à presença de um conjunto de itens relacionados com a presença de outro conjunto de itens, assim encontrando padrões de como os dados estão armazenados e os seus comportamentos;
a) classificação: a tarefa da classificação refere a uma análise dos dados de entrada em um número finito de classe. O algoritmo de classificação deve ser capaz de identificar alguma relação de um dado de entrada com alguma classe. Para construção do algoritmo de classificação, deve ser feito um treinamento, informando diversos valores para uma classe em específico, assim o algoritmo terá um parâmetro para definir os dados que pertencem a uma determinada classe (SIDNEY, 2010, p. 20);
b) detecção de anomalias: segundo Sidney (2010, p. 22), detecção de anomalias é uma etapa para buscar dados que não estão encaixando nas classes. Estes dados possuem valores não esperados ou atípicos.

Segundo Santos (2008, p. 19), a MD dispõe de algumas técnicas para realizar a extração dos dados, porém, é necessário avaliar com cuidado qual técnica de mineração de dados será capaz de atender aos seus objetivos, pois caso seja escolhido à técnica incorreta os resultados obtidos poderão ser insatisfatórios.

2.2 MINERAÇÃO DE TEXTOS

Segundo Silva E. (2010, p. 18), as pesquisas na área de MD inicialmente focaram-se apenas nos tipos de dados estruturados, mas nos últimos anos ocorreu uma elevação no número de documentos das organizações, assim iniciou as pesquisas de técnicas de manipulação e extração de informações ou de conhecimentos em dados que não possuem uma estruturada definida, por exemplo: documentos de textos.

> A mineração de textos surgiu a partir da necessidade de se descobrir, de forma automática, informações (padrões e anomalias) em textos. O uso dessa tecnologia permite recuperar informações, extrair dados, resumir documentos, descobrir padrões, associações e regras e realizar análises qualitativas ou quantitativas em documentos de texto. (ARANHA; PASSOS, 2006, p. 2).

Segundo Carrilho Junior (2007, p. 26), o processo de mineração de textos possui as seguintes etapas: coleta, pré-processamento, indexação, mineração e análise da informação.

2.2.1 Coleta

Segundo Carrilho Junior (2007, p. 27), a etapa da coleta tem por finalidade a construção do conhecimento e da base textual. Segundo Aranha (2007, p. 41), existem três fontes de dados: pastas de arquivos encontradas no disco rígido, banco de dados e a internet.

Segundo Carrilho Junior (2007, p. 28), a forma mais comum de se armazenar documentos digitais é através de pasta e/ou arquivos textuais ou binários. Nos bancos de dados, a obtenção das informações são feitas através dos conteúdos das colunas do tipo *String*. Entretando, o *Data Warehouse* é uma tecnologia que permite armazenar as informações dispersas, através da identificação, compreensão, integração e agregação dos dados, de forma a posicioná-los nos locais mais apropriados visando atender à estratégia organizacional das empresas (BRACKETT, 1996, p. 12).

O terceira fonte de dados é a internet. Nesta fonte, a heterogeneidade é o desafio predominante, pois existe uma infinidade de tipos de página, como notícias de revistas, *bloggers*, anúncios, documentos, artigos técnicos e planilhas (CARRILHO JUNIOR, 2007, p. 28). Para realizar a coleta neste ambiente é comum a utilização *Crawler* ou *Webcrawle*, que são robôs especializados em coletar dados na internet, salvando páginas visitadas com o objetivo de atender a necessidade do usuário. Estes robôs analisam a página para identificar

textos e *hiperlink*. Os *hiperlink* identificados são armazenados em um fila que será gerenciada pelo *crawler*. Outra função é a gerenciamento do percurso, que tem como objetivo impedir que o robô visite várias vezes a mesma página ou entre em ciclos eternos (SANTOS, 2010, p. 19).

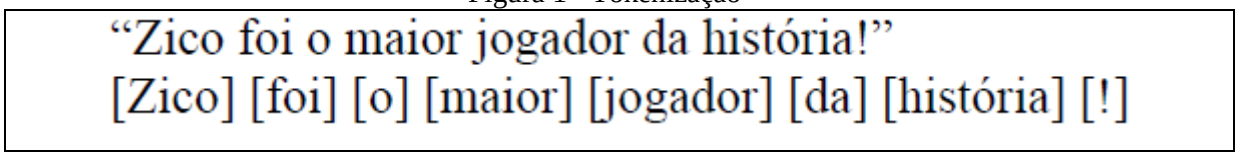
2.2.2 Pré-Processamento

A etapa de pré-processamento tem como objetivo aumentar a qualidade dos dados, realizando a limpeza dos dados e modelando os dados para serem trabalhados nas próximas etapas. Esta etapa possui diversas técnicas que podem ser aplicar sobres os dados, não existindo uma técnica melhor que a outra e sim uma melhor em determinado contexto. (CARRILHO JUNIOR, 2007, p. 30).

Segundo Silva E. (2010, p . 22), as técnicas mais utilizadas na etapa de pré-processamento são: Tokenização, *Stopwords*, Normalização e Sinônimos, serão descritas a seguir:

A técnica de tokenização tem a responsabilidade de extrair unidades mínimas de texto. Tais unidades são conhecidas como *Tokens*. Um *token* não necessariamente representa uma única palavra em alguns casos os *tokens* devem ser agrupados para fazerem sentido (CARRILHO JUNIOR, 2007, p. 31). A frase “Zico foi o maior jogador da história!” possui oito *tokens*, conforme o exemplo abaixo:

Figura 1 - Tokenização



Fonte: Carrilho Junior (2017, p. 31).

A tokenização aparentemente pode parecer simples, mas a sua dificuldade está em identificar quais termos não podem estar em *tokens* separados.

> Entretanto, a tarefa de identificação de tokens, que é relativamente simples para o ser humano, pode ser bastante complexa de ser executada por um computador. Este fato é atribuído ao grande número de papéis que os delimitadores podem assumir. Por exemplo, o “ponto” pode ser usado para marcar o fim de uma sentença, mas também é usado em abreviações e números. Outro exemplo é o travessão, que pode indicar o início de uma citação no texto ou, quando entre dígitos, indicar um número de telefone (ex., (21) 2235-7553) ou uma operação de subtração, também entre números. (CARRILHO JUNIOR, 2007, p. 31).

Para melhorar a tokenização utiliza-se técnicas de representação do texto, as mesmas serão descritas a seguir:

c) palavras abreviadas que não fazem sentido estão em *tokens* separados (SANTOS,

2010, p. 24);

d) palavras combinadas possuem um significado quando juntas, ao separá-las, perde-se o significado. Por exemplo, as palavras Coca Cola separadas não possuem a mesma relevância do que quando lidas em seguida (SANTOS, 2010, p. 24);
e) símbolos da internet tais como emails, urls de *websites,* etc., o ponto pode identificar um novo *token*. Porém, o ponto faz parte da palavra, exemplo: www.furb.br (CARRILHO JUNIOR, 2007, p. 32);
f) números nos textos podem conter vírgula ou ponto e o processo de tokenização pode separar os números em *tokens* diferentes (CARRILHO JUNIOR, 2007, p. 32).

Segundo Uber (2004, p. 15), os textos em geral possuem diversas palavras que são apenas úteis para o entendimento e compreensão geral do texto. Estas palavras são conhecidas como s*topwords*, que correspondem ao chamado de *stoplist* que nada mais é que a lista de *stopwords*.

Uma lista de *stopwords* é constituída pelas palavras de maior aparição em uma massa textual e, normalmente, correspondem aos artigos, preposições, pontuação, conjunções e pronomes de uma língua. A identificação e a remoção desta classe de palavras reduz de forma considerável o tamanho final do léxico, tendo como consequência benéfica o aumento de desempenho do sistema como um todo (CARRILHO JUNIOR, 2007, p. 38).

Segundo Santos (2010, p. 26), a normalização é a etapa que tem o objetivo de identificar as palavras que possuem alguma relação entre elas e com isso às mesmas são agrupadas.

As principais técnicas de normalização são:

a) *stemming*: segundo Carrilho Junior (2007, p. 40), o processo de *stemming* realiza a transformação das palavras buscando seu respectivo radical. Portanto, será eliminado das palavras o plural e os diferentes tempos verbais. Para realizar este processo pode ser optado pelos métodos de *Stemmer* S, Porter e Lovis, apresentados abaixo:
   - método de Porter: o método de Porter analisa as palavras presentes nos textos e as convertem para os seus respectivos radicais, fazendo com que palavras semelhantes se tornem uma palavra só. Por exemplo, ‘correr’, ‘corrida’, ‘corrido’ e ‘corridão’, possuem o mesmo radical ‘corr’ (SANTOS, 2010, p. 27),

- método *Stemmer* S: segundo Carrilho Junior (2007, p. 40), este método é focado na termino das palavras em inglês, procurando remover os sufixos: *ies*, *es* e *s* (com exceções). Este método é bastante utilizado pelo fato de raramente surpreender o usuário com erros,
- método de Lovins: sensível ao contexto. Abrange uma gama maior de sufixo (250 ao todo). Baseia-se numa lista de regras, chamada de regras de Lovins que, num passo único, faz a remoção de, no máximo, um único sufixo por palavra (CARRILHO JUNIOR, 2007, p. 41);

b) *lemmatization*: segundo Santos (2010, p. 27), o processo de *lemmatization* consiste em transformar as diversas derivações das palavras em sua forma primitiva. Por exemplo, as palavras 'livro', 'livros' e 'livraria', compartilham a mesma palavra primitiva, 'livro'.

Segundo Aranha (2007, p. 78), os sinônimos são palavras diferentes, que possuem o mesmo significado. Entretanto, as palavras sinônimas podem possuir significado um pouco divergentes, mas se for analisado o contexto em que se encontram possuem significado idênticos. Os sinônimos podem ser aplicados para correferenciar termos.

> As relações de sinonímia dessas palavras, em geral, são especiais e apresentam na redundância de informação um certo grau de hierarquia de organização do conhecimento como "tipo de" ou "parte de". Apesar de que sinônimos horizontais também são encontrados. (ARANHA, 2007, p. 78).

Em geral os textos apresentam esse tipo de correferência para não repetir a mesma palavra. Exemplo, um texto pode começar falando de uma pesquisa, e depois correferenciar a mesma pesquisa como trabalho, *paper* ou desenvolvimento. No domínio de negócios um exemplo comum é apresentar a empresa e depois chamar de agência, corporação etc. As relações hierárquicas podem ser recuperadas automaticamente utilizando regras, por exemplo, "gasolina e outros combustíveis" indica que gasolina é um tipo de combustível (ARANHA, 2007, p. 78).

2.2.3 Classificação

A classificação pode ser dividida em níveis, ou seja, pode ser classifica em nível de aspecto, que é o nível de maior granularidade dos documentos, quando a preocupação maior está em classificar cada característica do documento (SILVA N., 2010, p. 12).

Para realizar classificação os seguintes passos são realizados, para melhor demonstramos os passos a seguinte frase será utilizada, "Os lanches do Burguer King são

maravilhosos, e a sobremesa não é cara. O McDonald's é muito bom, mas dizer que o McDonald's é saudável é piada":

a) encontrar e classificar as palavra opinativas: este primeiro passo identifica palavras opinativas e classifica-as como positivas, negativas ou neutras. Após a execução deste passo temos o seguinte resultado: "Os lanches do Burguer King são maravilhosos [+1], e a sobremesa não é cara [-1]. O McDonald's é muito bom [+1], mas dizer que o McDonald's é saudável [+1] é piada" (SILVA N., 2010, p.12);
b) cláusulas negativas: o segundo passo procura identificar as palavras negativas, que irão inverter a polaridade das palavras opinativas. Após a execução deste passo temos o seguinte resultado: "Os lanches do Burguer King são maravilhosos [+1], e a sobremesa não é cara [+1]. O McDonald's é muito bom [+1], mas dizer que o McDonald's é saudável [+1] é piada" (SILVA N., 2010, p.12);
c) cláusulas adversativas: palavras que tratam de oposição, como as cláusulas adversativas, mostram opiniões contrárias em relação ao mesmo objeto em estudo (SILVA N., 2010, p. 12). Após a execução deste passo temos o seguinte resultado: "Os lanches do Burguer King são maravilhosos [+1], e a sobremesa não é cara [-1]. O McDonald's é muito bom [+1], mas dizer que o McDonald's é saudável [-1] é piada". Neste cenário, a última palavra opinativa é positiva, por causa da conjunção "mas", as duas palavras opinativas, devem possuir polaridades contrárias.

Uma vantagem desta abordagem é não ser necessário classificar previamente um conjunto de textos, entretando para o sucesso é necessário identificar palavras opinativas de maneira eficiente, pois são elas que ilustram a polaridade dos sentimentos expressos (FERREIRA, 2010).

2.2.4 Análise da informação

Segundo Carrilho Junior (2007, p. 56), a etapa de análise da informação em algumas literaturas é conhecida como o pós-processamento de dados, tendo como objetivo verificar a eficiência da aplicação das técnicas e algoritmos das etapas anteriores, ou seja, avaliar se o objetivo foi cumprido, que é a descoberta de conhecimento desconhecido.

Segundo Santos (2010, p. 33), na análise da informação deve ser verificado se o conhecimento extraído é útil para o assunto em questão. Existem diversas maneiras para

realizar a validação podendo ser utilizado métricas qualitativa ou quantitativa, um exemplo seria a interpretação de um especialista no assunto.

> Existem diversas maneiras de se avaliar a mineração como um todo, seja de forma qualitativa ou quantitativa. A utilização de métricas, conforme já mencionado, é considerada uma forma quantitativa, ao passo que a utilização do conhecimento de especialistas no domínio é considerada uma forma qualitativa. Os especialistas devem sempre ser consultados, em todas as etapas da Mineração, balizando a análise, ajudando a resolver situações de conflito, indicando caminhos e complementando informações. Entretanto, alguns conflitos podem ocorrer como a divergência de opiniões entre dois ou mais especialistas, bem como, a própria mudança de opinião de um mesmo ao longo do tempo. (CARRILHO JUNIOR, 2007, p. 56).

Uma das formas mais intuitivas de se analisar um resultado seria com a utilização de gráficos, tabelas e outras formas visuais. Assim o usuário pode utilizar os resultados para a tomada de decisão, segundo Carrilho Junior (2007, p. 56).

2.3 TRABALHOS CORRELATOS

Esta seção destina-se a apresentar os trabalhos de Santos (2010), Ramos e Bräscher (2009), Torres (2005) e Silva E. (2010).

Santos (2010) construiu um protótipo com o objetivo de realizar a mineração de opinião escritas no Twitter. Para isto é utilizado a modelo conhecido como vetor de características que possui uma abordagem estatística para análise de textos, assim indexando os documentos de texto de maneira estruturada. Na mineração dos textos é utilizado o método de aprendizagem de máquina conhecido com *SVM* para realizar a classificação entre sentimentos positivos e negativos. Segundo Santos (2010), o método *SVM* treinado conseguiu alcançar em média uma taxa de acerto de 80%.

Ramos e Bräscher (2009) produziram uma pesquisa para verificar a eficácia da descoberta de conhecimento em texto (DCT) na descoberta de informações para apoiar à construção de indicadores e definição de políticas públicas, sendo utilizado a base do Serviço Brasileiro de Respostas Técnicas. Para alcançar os seus objetivos, Ramos e Bräscher utilizaram técnicas de agrupamento do software SAS. Segundo Ramos e Bräscher (2009), a aplicação do DCT conseguiu recuperar informações que não podiam ser recuperadas utilizando recursos tradicionais de recuperação de informação, por exemplo, a preocupação dos microempresários com os apectos regulatórios e legais e com as questões ambientais.

Torres (2005) desenvolveu uma ferramenta para montagem de perfis através da identificação das características do usuário, de forma que seja possível fazer o direcionamento automático das informações a partir das preferências de cada usuário. Para isto é utilizado técnicas de mineração de textos para extrair das páginas navegadas características que

representam suas preferências e utilizando o algoritmo de Naive Bayes para realizar a classificação. Segundo Torres (2005), a ferramenta conseguiu encontrar o valor máximo de 79% de acertos.

Silva E. (2010) testou a viabilidade da utilização de técnicas de processamento automático de texto para a anotação das sessões dos debates parlamentares da Assembleia da República de Portugal. Para realizar o pré-processamento dos textos utilizou-se diversas técnicas da ferramenta *open source R/tm*. Para a classificação utilizou-se o algoritmo *KnnFlex*. Segundo Silva E. (2010), o nível de acertos foi de 50 a 60%.

O Quadro 1, apresenta de forma resumida as principais características dos sistemas descritos no decorrer desta seção.

Quadro 1 - Característica dos trabalhos relacionados

características / trabalhos relacionados	SANTOS (2010)	RAMOS E BRÄSCHER (2009)	TORRES (2005)	SILVA E. (2010)
técnica de agrupamento	-	X	-	-
técnica de classificação	*SVM*	-	Naive Bayes	*KnnFlex*
ferramentas de auxilio	-	SAS	-	R/tm
remoção *stopwords*	X	X	X	X
normalização	X	X	X	X
sinônimos	-	-	X	-

A partir do Quadro 1, pode-se verificar que é comum a utilização das técnicas de *stopwords* e normalização na etapa de pré-processamento. Os trabalhos de Ramos e Bräscher (2009), Silva E. (2010), utilizam ferramentas para auxiliar nesta etapa, sendo a SAS e R/tm. Silva E. (2010), utilizou *stopwords* específicas para o contexto parlamentar além das pré-definidas.

3 DESENVOLVIMENTO

Neste capítulo serão apresentados as etapas do desenvolvimento da ferramenta de MT proposta neste trabalho. Na seção 3.1 são enumerados os requisitos principais do projeto. A seção 3.2 apresenta a especificação da ferramenta. A seção 3.3 detalha a implementação das principais técnicas e algoritmos utilizados no desenvolvimento da ferramenta. Por fim, a seção 3.4 apresenta as séries de testes efetuados para a validação do projeto e os resultados obtidos.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) da ferramenta de mineração de textos são:

a) a ferramenta deve utilizar algumas das etapas do processo de MT, sendo elas: coleta, pré-processamento e análise (Requisito Funcional - RF);
b) a ferramenta deve permitir o cadastro de sinônimos (RF);
c) a ferramenta deve permitir coletar os dados do Twitter (RF);
d) a ferramenta deve utilizar as palavras positivas e negativas informadas pelo usuário (RF);
e) a ferramenta deve permitir visualizar os *tweets* classificados;
f) a ferramenta deve utilizar a linguagem C# (RNF);
g) a ferramenta deve utilizar o ambiente de desenvolvimento Microsoft Visual Studio (RNF);
h) a ferramenta deve utilizar a biblioteca LinqToTwitter (RNF).

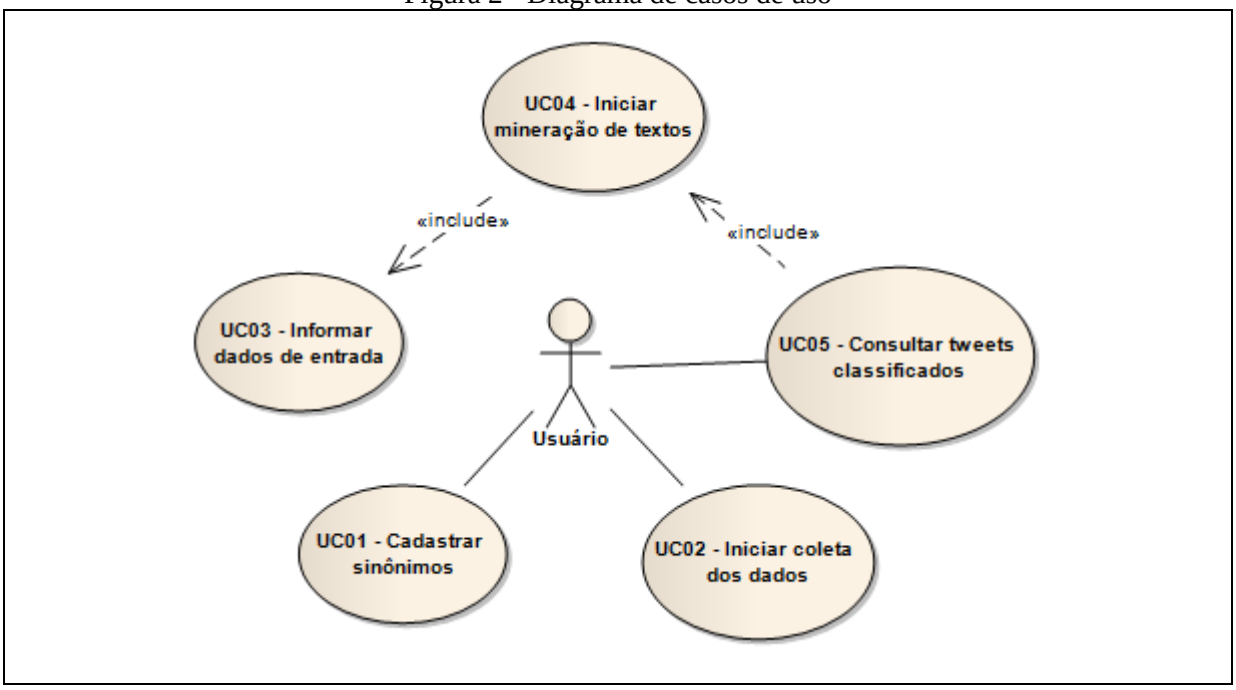
3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação da ferramenta, representada por diagramas da *Unified Modeling Language* (UML) e modelada utilizando a ferramenta Enterprise Architect 7.5. Foram utilizados os diagramas de casos de uso, diagrama de classes para representar a metodologia utilizada no desenvolvimento da ferramenta.

3.2.1 Diagrama de Casos de usos

A Figura 2 exibe o diagrama de casos de uso com as ações disponíveis pela ferramenta. Identificou-se apenas um ator, denominado Usuário, o qual utiliza a funcionalidade do aplicativo.

Figura 2 - Diagrama de casos de uso



Segue detalhamento dos casos de uso exibidos no diagrama da Figura 2:

a) `UC01 - Cadastrar sinônimos`: permite ao usuário cadastrar, alterar e/ou excluir um sinônimo;

b) `UC02 - Iniciar coleta dos dados`: permite ao usuário iniciar a coleta dos dados do Twitter;

c) `UC03 - Informar dados de entradas`: permite que o usuário informe as palavras positivas e palavras negativas ou se irá considerar palavras inteiras antes de iniciar o processo de MT;

d) `UC04 - Iniciar mineração de textos`: permite ao usuário iniciar o processo de MT;

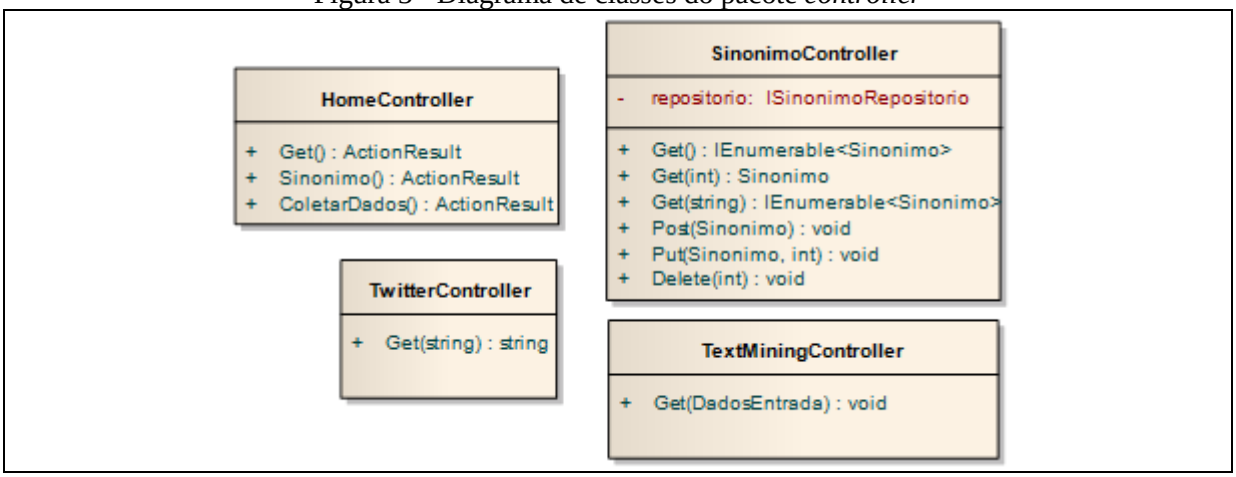
e) `UC05 - Consultar tweets classificados`: permite ao usuário visualizar os *tweets* que foram classificados.

3.2.2 Diagrama de classes

Para facilitar a visualização e o entendimento do relacionamento entre as classes optou-se em separá-las em pacotes: *controller* e *model*.

A Figura 3 representa o diagrama de classes do pacote *controller*.

Figura 3 - Diagrama de classes do pacote *controller*

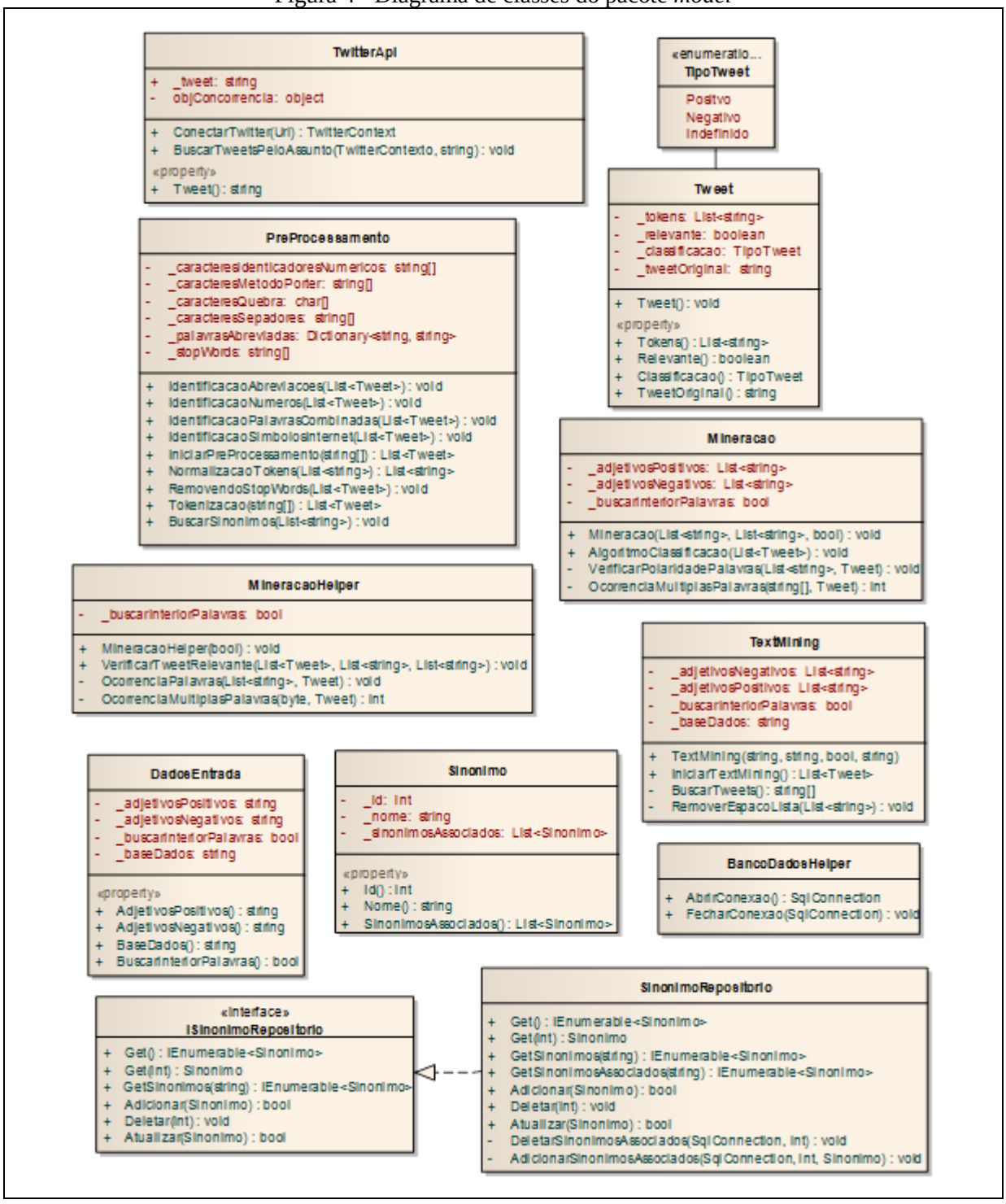


O pacote *controller* tem a responsabilidade de controlar o fluxo da ferramenta. Ele decide quais as classes do pacote *models* que serão chamadas e quais telas serão exibidas. O pacote controller contém as classes `HomeController`, `TwitterController`, `SinonimoController` e `TextMiningController`. Abaixo é descrito a utilização das mesmas:

a) `HomeController`: exibe as telas conforme o usuário interage com a ferramenta. Realiza a chamada dos serviços da ferramenta;
b) `TwitterController`: inicia o processo de coleta dos *tweets*;
c) `SinonimoController`: realiza as tarefas de incluir, atualizar, excluir e associar sinônimos;
d) `TextMiningController`: inicia o processo de TM.

A Figura 4 representa o diagrama de classes do pacote *model*.

Figura 4 - Diagrama de classes do pacote *model*



No pacote *model* estão as classes responsáveis pelas regras de negócio, sendo elas: `BancoDadosHelper`, `DadosEntrada`, `Mineracao`, `Indexacao`, `PreProcessamento`, `Sinonimo`, `SinonimoRepositorio`, `TextMining`, `TipoTweet`, `Tweet`, `TwitterApi`.

a) `BancoDadosHelper`: disponibiliza métodos para criar a conexão com o banco de dados;

b) `DadosEntrada`: possui o objetivo de facilitar a troca de informações entre a

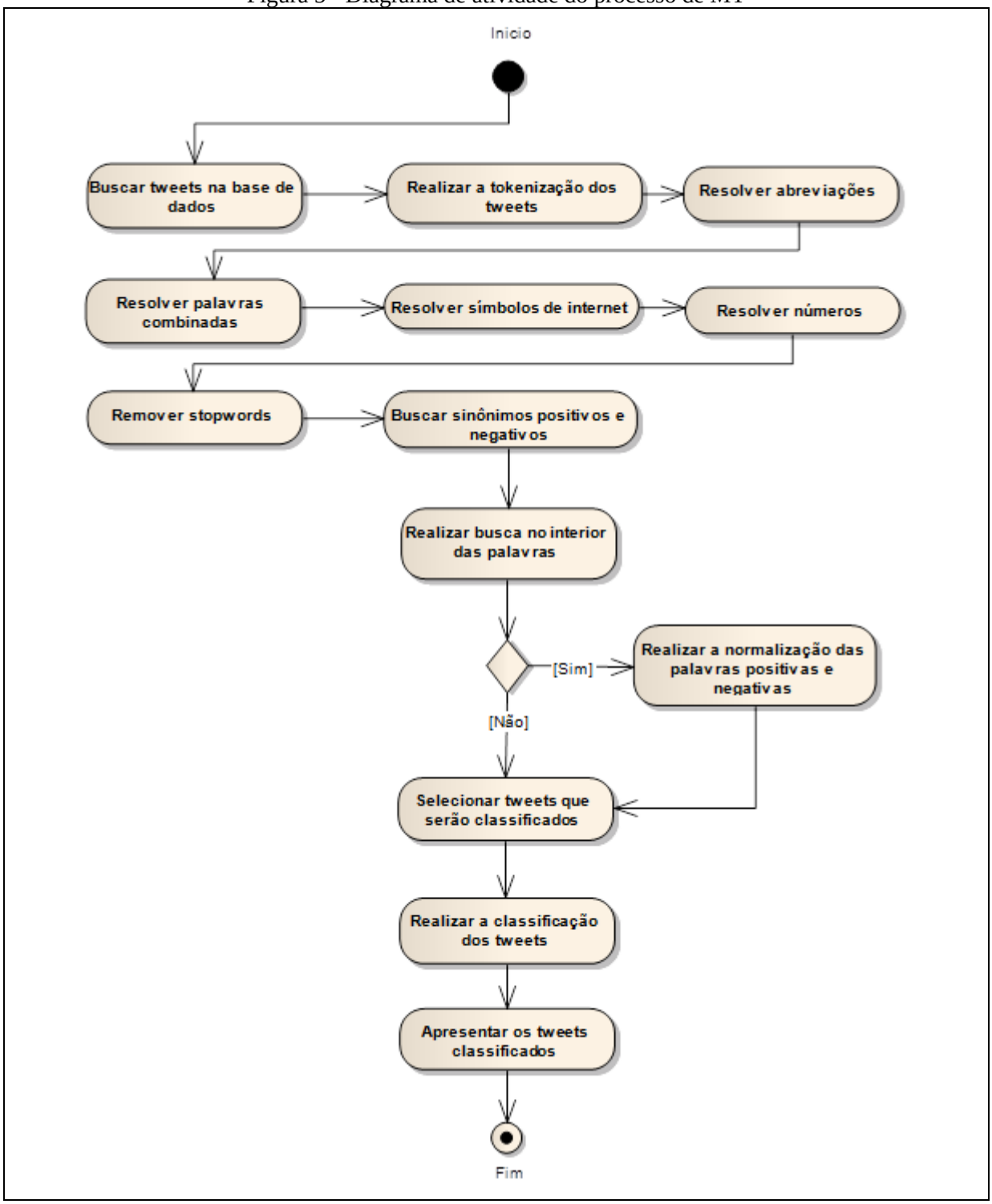
interface e o *controller*;

c) `Mineracao`: realiza a classificação dos *tweets*;
d) `MineracaoHelper`: identifica os *tweets* que devem ser classificados;
e) `PreProcessamento`: se encarrega da etapa de pré-processamento, contendo as técnicas necessárias, tais como: tokenização, *stopwords*, normalização e sinônimos;
f) `Sinonimo`: possui o objetivo de facilitar a troca de informações entre *controllers* e *models*;
g) `SinonimoRepositorio`: possui a responsabilidade de incluir, atualizar, deletar e associar sinônimos;
h) `TextMining`: esta classe orquestra as etapas da MT;
i) `TipoTweet`: uma *enumeration* que contém os estados que um *tweet* pode possuir;
j) `Tweet`: possui o objetivo de facilitar a troca de informações entres as classes tanto de models como de controllers;
k) `TwitterApi`: é responsável pela etapa de coleta dos *tweets*, onde a mesma se conecta ao Twitter e recupera os *tweets*.

3.2.3 Diagrama de atividades

A Figura 5, apresenta um diagrama de atividade que representa as etapas de MT executadas pela ferramenta.

Figura 5 - Diagrama de atividade do processo de MT



A ferramenta inicia o processo de mineração realizando uma busca da base de dados, sendo representada pela ação `Buscar tweets na base de dados`.

Depois de estabelecer a conexão com a base de dados informada pelo usuário, inicia-se a etapa de pré-processamento, representada pelas ações: `Realizar a tokenização dos tweets`, `Resolver abreviações`, `Resolver palavras combinadas`, `Resolver símbolos`

de internet, Resolver números, Remover stopwords, Buscar sinônimos positivos e negativos. Estas etapas possuem o objetivo de limpar os dados e de adaptar o texto para futuras representações mais estruturadas. A ação, Realizar busca no interior das palavras, verifica se o usuário marcou o campo Buscar interior das palavras. Se estiver marcado, executa-se a ação Realizar a normalização das palavras positivas e negativas para encontrar o radical das palavras positivas e negativas.

Após realizar o pré-processamento, é verificado quais *tweets* serão classificados, representada pela ação Selecionar tweets que serão classificados. Esta etapa reduz a quantidade de *tweets* que serão classificados. Com os *tweets* indexados, realiza-se o processo de classificação, representado pela ação Realizar a classificação dos tweets para identificar se os *tweets* são positivos, negativos ou indefinidos. Por fim, a ação Apresentar os tweets classificados mostra os dados para o usuário.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

A ferramanta foi desenvolvida utilizando como plataforma a linguagem C#. Como ambiente de desenvolvimento foi selecionado a IDE Microsoft Visual Studio 2012.

As tecnologias utilizadas na implementação do projeto são enumeradas a seguir:

a) LinqToTwitter: biblioteca que aproveita a sintaxe do LINQ para realizar acessos a API do Twitter que, em sua versão 2.1.9, foi utilizada para realizar o *login* no Twitter e buscar *tweets* para serem minerados. Sua interface de acesso é disponibilizada em C#;
b) JQuery: utilizada na versão 1.8.2, é uma biblioteca *javascript* suportada em múltiplos navegadores desenvolvida para simplificar a interação de *javascripts* para páginas *html*. O JQuery foi utilizado para manipulação dos campos e chamadas da ferramenta. Sua interface de acesso é disponibilizada em javascript;
c) ASP.NET MVC: utilizado na versão 4, é um padrão de arquitetura para a criar aplicações Web baseadas no padrão de projeto *Model View Controller* (MVC). O ASP.NET MVC foi utilizada para criação das interfaces de acesso a ferramenta;
d) ASP.NET Web API: f*ramework* para facilitar a construção de serviços HTTP puros, onde que o pedido e resposta acontecem utilizando o protocolo HTTP. Estes

serviços criados podem disponibilizados para navegadores e dispositivos móveis acessá-los. A ferramenta foi toda desenvolvida utilizando Web API e a plataforma .NET Framework 4.5.

3.3.2 Implementação da ferramenta

Nesta seção são apresentadas as etapas da implementação das funcionalidades da ferramenta desenvolvida neste trabalho. Na seção 3.3.2.1 descreve-se a maneira como é feito a coleta dos dados. Na seção 3.3.2.2 apresenta-se as etapas envolvidas no pré-processamento detalhando as técnicas implementas. Na seção 3.3.2.3 descreve-se a etapa de identificação de tweets relavantes. Na seção 3.3.2.4 descreve-se a etapa de Classificação dos *tweets*.

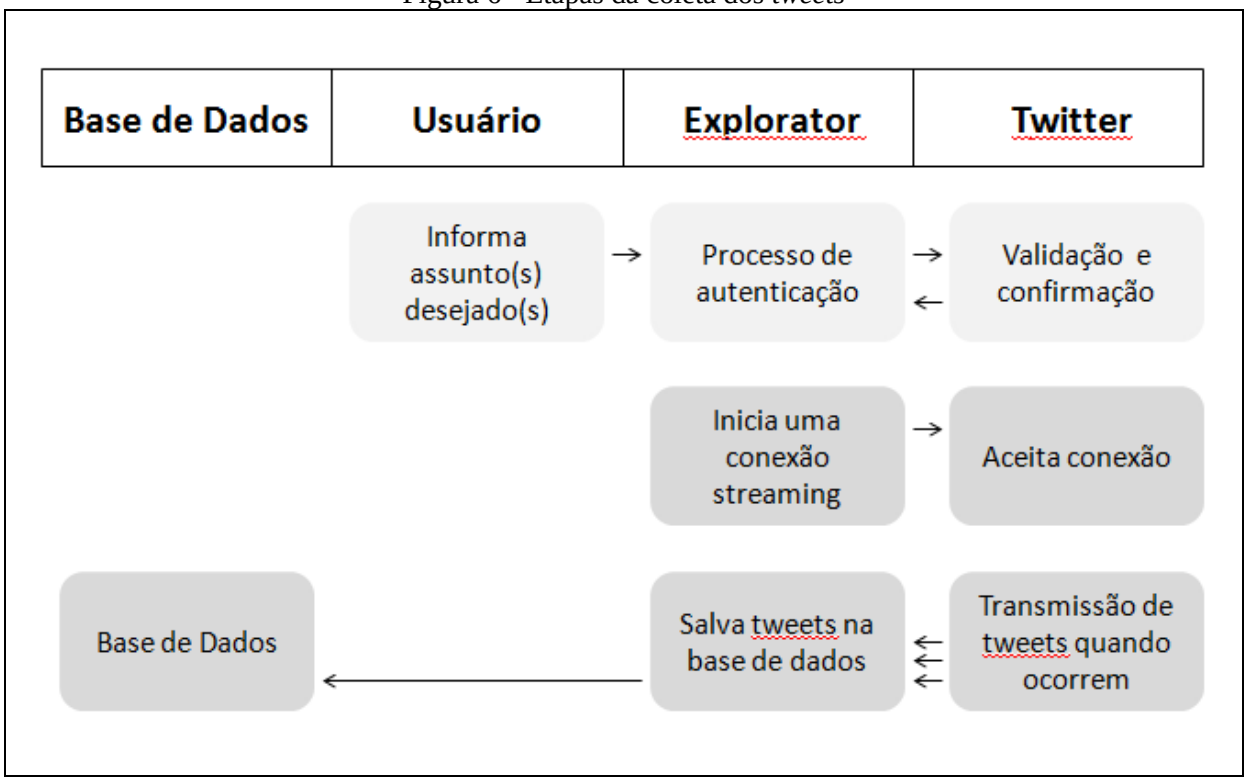
3.3.2.1 Coleta dos dados

A primeira etapa da MT é a coleta dos *tweets*. Para coletar os dados do Twitter, o mesmo fornece duas APIs , sendo elas: REST API e Streaming API.

A REST API prove um acesso simples para a maioria das funcionalidades do Twitter, sendo elas: gerenciar e criar suas aplicações, realizar buscas de *tweets* relevantes, atualizar a *timeline*. A *Streaming* API permite que seja capturado *tweets* em tempo real, neste trabalho está sendo utilizado a *Streaming* API com o auxilo da biblioteca LinqToTwitter.

Para realizar a coleta dos *tweets*, é necessário possuir uma aplicação para cadastrá-la no *site* de desenvolvedores do Twitter (TWITTER, 2012). Tendo, a aplicação cadastrada pode-se iniciar a coleta dos *tweets*. Para realizar está tarefa, segue-se os passos demonstrados na Figura 6. As acões em cinza claro são referentes a conexão com o Twitter e as ações em cinza escuro a coleta.

Figura 6 - Etapas da coleta dos *tweets*



Para realizar a tarefa de autenticar-se no Twitter, o Quadro 2 exibe a implementação do método `ConectarTwitter` (membro da classe `TwitterApi`), responsável por efetuar a autenticação com o Twitter.

Quadro 2 - Autenticação no Twitter

```
01. public TwitterContext ConectarTwitter(Uri retorno)
02. {
03.     SignInAuthorizer auth = new SignInAuthorizer()
04.     {
05.         //Adiciona as credenciais
06.         Credentials = new LinqToTwitter.InMemoryCredentials()
07.         {
08.             ConsumerKey = ConfigurationManager.AppSettings["consumerKey"],
09.             ConsumerSecret =
10. ConfigurationManager.AppSettings["consumerSecret"],
11.             OAuthToken = ConfigurationManager.AppSettings["accessToken"],
12.             AccessToken =
13. ConfigurationManager.AppSettings["accessTokenSecret"]
14.         },
15.         //Valores default
16.         OAuthRequestTokenUrl = "https://api.twitter.com/oauth/request_token",
17.         OAuthAccessTokenUrl = "https://api.twitter.com/oauth/access_token",
18.         OAuthAuthorizeUrl = "https://api.twitter.com/oauth/authorize",
19.     };
20.     //Inicia a autenticacao do aplicativo
21.     auth.BeginAuthorization(retorno);
22.     return new TwitterContext(auth);
23. }
```

Como exposto no Quadro 2, a classe `SignInAuthorizer` (contida na biblioteca LinqToTwitter) é utilizada para realizar a autenticação, ao qual cria uma instância da classe `TwitterContext`. As propriedades `ConsumerKey`, `ConsumerSecret`, `OAuthToken` e

`AcessToken` devem receber as chaves que o Twitter disponibiliza para autenticação, estas chaves estão disponíveis na tela de detalhes de sua aplicação.

Após realizar a autenticação inicia-se a coleta dos *tweets*. No Quadro 3, é exibido a implementação do método `BuscarTweetsPeloAssunto` (membro da classe `TwitterApi`), responsável por buscar os *tweets*.

Quadro 3 - Coleta dos *tweets*

```
01. public void BuscarTweetsPeloAssunto(TwitterContext twitterContexto, string assunto)
02. {
03.     string diret = HostingEnvironment.ApplicationPhysicalPath;
04.     (from strm
05.      in twitterContexto.Streaming
06.      where strm.Type == StreamingType.Filter &&
07.            strm.Track == assunto &&
08.            strm.Language == "pt"
09.      select strm)
10.     .StreamingCallback(streaming =>
11.     {
12.         if (streaming.Status != TwitterErrorStatus.Success)
13.         {
15.             streaming.CloseStream();
15.             return;
16.         }
17.         else if (!string.IsNullOrEmpty(streaming.Content))
18.         {
19.             TwitterStreamApi tweet =
20.                   JsonConvert.DeserializeObject<TwitterStreamApi>(streaming.Content);
21.             lock (objConcorrencia)
22.             {
23.                 string pathFile = Path.Combine(diret, string.Concat(assunto,".txt"));
24.                 StreamWriter arquivo = new StreamWriter(pathFile, true);
25.                 arquivo.WriteLine(tweet.Texto);
26.                 arquivo.Close();
27.             }
28.         }
29.     }).FirstOrDefault();
30. }
```

Como apresentado no Quadro 3, a classe `TwitterContext` (contida na biblioteca LinqToTwitter), é utilizada para realizar a pesquisa dos *tweets*, sendo realizado o seguinte filtro:

a) Os *tweets* devem conter os assuntos informados pelo usuário;

b) Os *tweets* devem está em português brasileiro.

Os *tweets* coletados são gravados em um arquivo `.txt`. O Twitter envia os *tweets* no momento que eles ocorrem, onde é necessário fazer um controle de concorrência antes salvá-los no arquivo.

3.3.2.2 Pré-Processamento

Após a coleta dos dados, inicia-se a etapa de pré-processamento que tem como objetivo remover dados desnecessários para o entendimento do texto. Nesta etapa está sendo

realizado a *tokenization*, remoção de *stopwords*, normalização e busca de sinônimos. Para demonstrar o fluxo das etapas serão utilizadas as seguintes frases: “O windows 8 está custando R$ 169,90 reais que absurdo”, “A Decoracao&Cia possui uma loja com produtos ótimos e preço acessivel”, “Vídeo do novo windows http://www.teste.com”, “O filme novo do Thor ñ tem para ninguem, é mto bom, obg a vc Jaque que foi junto cmg assistir”.

3.3.2.2.1 Tokenização

Para realizar o processo de tokenização, optou-se por quebrar o *tweet* por espaço, tendo assim, uma coleção de *tokens*. Para cada *token* desta coleção, percorre-se os caracteres, procurando identificar a existência de outros caracteres de quebras " `().,<>!-?;':"/.` ". Se for encontrado algum desses caracteres, subdividi-se o *token* atual em novos *tokens*.

No Quadro 4, é apresentado o método `Tokenizacao` (membro da classe `PreProcessamento`), responsável pela transformação do texto em *tokens*.

Quadro 4 - Transforma os *tweets* em *tokens*

```
01. public List<Tweet> Tokenizacao(string[] tweets)
02. {
03.     List<Tweet> tweetsTokens = new List<Tweet>();
04.     foreach (string tweet in tweets)
05.     {
06.         Tweet objTweet = new Tweet();
07.         tweetsTokens.Add(objTweet);
08.         StringBuilder tokens = new StringBuilder();
09.         string[] palavras = tweet.Split(new char[] { ' ' },
10.                                         StringSplitOptions.RemoveEmptyEntries);
11.         foreach (string palavra in palavras)
12.         {
13.             foreach (char caractere in palavra)
14.             {
15.                 if (_caracteresQuebra.Contains(caractere))
16.                 {
17.                     if (tokens.Length != 0)
18.                     {
19.                         objTweet.Tokens.Add(tokens.ToString());
20.                         tokens.Clear();
21.                     }
22.                     objTweet.Tokens.Add(caractere.ToString().Trim());
23.                     continue;
24.                 }
25.                 tokens.Append(caractere.ToString().Trim());
26.             }
27.             if (tokens.Length != 0)
28.             {
29.                 objTweet.Tokens.Add(tokens.ToString());
30.                 tokens.Clear();
31.             }
32.         }
33.     }
34.     return tweetsTokens;
35. }
```

Após a tokenização as frases de exemplos ficaram da seguinte forma:

a) [O][windows][8][está][custando][R$][169][,][00][reais][que][absurdo];

b) [A][Decor][&][Cia][possui][uma][loja][com][produtos][ótimos][e][preço][acessivel];

c) [Vídeo][do][novo][windows][http][:][/][/][www][.][teste][.][com];

d) [O][filme][novo][do][Thor][ñ][tem][para][ninguem][,][é][mto][bom][,][obg][a][vc][Jaque][que][foi][junto][cmg][assistir].

Nos exemplos acima, pode-se notar que alguns *tokens* perdem o sentido serapados, como por exemplo, os símbolos de intenert *url* e o valor. Para melhorar o processo de tokenização foi realizado a identificação de palavras abreviadas, combinadas, símbolos de internet e números.

Para realizar a identificação de abreviações foi utilizado um dicionário pré-estabelecido, onde é verificado se cada *token* pertence a este dicionário, se sim, ele é atualizado pelo valor contido no dicionário. No Quadro 5, é exibido o método `IdentificacaoAbreviacoes` (membro da classe `PreProcessamento`), que realiza o processo descrito anteriormente.

Quadro 5 - Identificação de abreviações

```
01. public void IdentificacaoAbreviacoes(List<Tweet> listaTokens)
02. {
03.     foreach (Tweet tweet in listaTokens)
04.     {
05.         List<string> tokens = tweet.Tokens;
06.         for (int i = 0; i < tokens.Count; i++)
07.         {
08.             string token = tokens[i].ToLower();
09.             //Verifica se o token contem no dicionario de palavras abreaviadas
10.             if (_palavrasAbreviadas.ContainsKey(token))
11.                 tokens[i] = _palavrasAbreviadas[token];
12.         }
13.     }
14. }
```

Após executar do método `IdentificacaoAbreviacoes`, tem-se os seguintes resultados:

a) [O][windows][8][está][custando][R$][169][,][00][reais][que][absurdo];

b) [A][Decor][&][Cia][possui][uma][loja][com][produtos][ótimos][e][preço][acessivel];

c) [Vídeo][do][novo][windows][http][:][/][/][w][w][w][.][teste][.][com];

d) [O][filme][novo][do][Thor][não][tem][para][ninguem][,][é][muito][bom][,][obrigado][a][voce][Jaque][que][foi][junto][comigo][assistir].

Para identificar palavras combinadas, verifica-se a existência de um *token* que seja igual a um desses caracteres: &, -. Caso seja encontrado, o *token* atual recebe a concatenação dos *tokens* anterior e próximo. No Quadro 6, é exibido o método

`IdentificacaoPalavrasCombinadas` (membro da classe `PreProcessamento`), que realiza o processo descrito acima.

Quadro 6 - Identificação de palavras combinadas

```
01. public void IdentificacaoPalavrasCombinadas(List<Tweet> listaTweets)
02. {
03.     foreach (Tweet tweet in listaTweets)
04.     {
05.         List<string> tokens = tweet.Tokens;
06.         foreach (string caractereSeparador in _caracteresSeparadores)
07.         {
08.             int indiceCaracteresSeparadores =
09.                                             tokens.IndexOf(caractereSeparador);
10.             if (indiceCaracteresSeparadores > -1)
11.             {
12.                 for (int i = indiceCaracteresSeparadores; i < tokens.Count; )
13.                 {
14.                     string token = tokens[i];
15.                     if (token == caractereSeparador)
16.                     {
17.                         string tokenAnterior = tokens[i - 1];
18.                         string tokenProximo = tokens[i + 1];
19.                         string novoToken = tokenAnterior + token +
20.                                                                 tokenProximo;
21.                         tokens.RemoveRange(i - 1, 3);
22.                         tokens.Insert((i - 1), novoToken);
23.                         i++;
24.                     }
25.                     i++;
26.                 }
27.             }
28.         }
29.     }
30.
31. }
```

Após executar o método `IdentificacaoPalavrasCombinadas`, a ferramenta apresenta os seguintes resultados:

a) [O][windows][8][está][custando][R$][169][,][00][reais][que][absurdo];

b) [A][Decor&Cia][possui][uma][loja][com][produtos][ótimos][e][preço][acessivel];

c) [Vídeo][do][novo][windows][http][:][/][/][www][.][teste][.][com];

d) [O][filme][novo][do][Thor][não][tem][para][ninguem][,][é][muito][bom][,][obrigado][a][voce][Jaque][que][foi][junto][comigo][assistir].

Para identificar os símbolos de internet (e-mail e *url*) é verificado se algum *token* é igual a seguinte palavra: http. Caso seja, realiza-se uma busca para encontrar as palavras: ‘com’ ou ‘br’. Se for encontrado a palavra ‘com’ é verificado se os próximos *tokens* são: ‘.’ e ‘br’. Caso sejam, os *tokens* são concatenados desde a identificação da palavra ‘http’, formando um único *token*. No Quadro 7, é exibido o método `IdentificaçãoSimbolosInternet` (membro da classe `PreProcessamento`), que tem a responsabilidade anteriormente citada:

Quadro 7 - Identificação de símbolos de internet

```
01. public void IdentificacaoSimbolosInternet(List<Tweet> listaTweets)
02. {
03.     ...
04.     //O código acima possuía um foreach pelos tweets e nos tokens do tweet
05.     //Criado as variáveis indiceInicial e simboloInternet
06.     string token = tokens[i];
07.     if (simboloInternet.Length == 0 && token.IndexOf("http") != -1)
08.     {
09.         indiceInicial = i;
10.         simboloInternet.Append(token);
11.         i++;
12.         for (; i < tokens.Count; i++)
13.         {
14.             token = tokens[i];
15.             if (token == "com")
16.             {
17.                 if (!(tokens[i + 1] == "." && tokens[i + 2] == "br"))
18.                 {
19.                     simboloInternet.Append(token);
20.                     tokens.RemoveRange(indiceInicial,(i - indiceInicial + 1));
21.                     tokens.Insert(indiceInicial, simboloInternet.ToString());
22.                     break;
23.                 }
24.             }
25.             else if (token == "br")
26.             {
27.                 simboloInternet.Append(token);
28.                 tokens.RemoveRange(indiceInicial,( i - (indiceInicial + 1)));
29.                 tokens.Insert(indiceInicial, simboloInternet.ToString());
30.                 break;
31.             }
32.             simboloInternet.Append(token);
33.     //O resto do código é apenas o fechando das chaves dos if, for.
34.
```

Após executar o método `IdentificaoSimbolosInternet`, tem-se os seguintes resultados:

a) [O][windows][8][está][custando][R$][169][,][00][reais][que][absurdo];

b) [A][Decor&Cia][possui][uma][loja][com][produtos][ótimos][e][preço][acessivel];

c) [Vídeo][do][novo][windows][http://www.teste.com];

d) [O][filme][novo][do][Thor][não][tem][para][ninguem][,][é][muito][bom][,][obrigado][a][voce][Jaque][que][foi][junto][comigo][assistir].

Para realizar a identificação de números é verificado a existência dos seguintes caracteres: ‘nº’, ‘nª’, ‘R$’, ‘$’. A partir disso, guarda-se o próximo *token* como número. Posteriormente verifica-se nos próximos *tokens* a existência de pontos, vírgulas ou números, concatenando-os ao *token* que contém o número. No Quadro 8 é apresentado o método `IdentificacaoNumeros` (membro da classe `PreProcessamento`), que possui o objetivo citado anteriormente.

Quadro 8 - Identificação de números

```
01.  public void IdentificacaoNumeros(List<Tweet> listaTweets)
02.  {
03.      //O código acima possuía um loop pelos tweets e nos caracteres numericos
04.      //Criado as variáveis indexCaracterNumerico e simboloInternet
05.      indexCaracterNumerico = tokens.IndexOf(identificadorNumerico);
06.      if (indexCaracterNumerico > -1)
07.      {
08.          for (int i = indexCaracterNumerico; i < tokens.Count; i++)
09.          {
10.              string token = tokens[i];
11.              int inteiro = int.MinValue;
12.              int inicialNumero = i;
13.              novoToken.Clear();
14.              if (token.IndexOf(identificadorNumerico) > -1)
15.              {
16.                  novoToken.Append(token);
17.                  if (int.TryParse(tokens[i + 1], out inteiro))
18.                  {
19.                      novoToken.Append(tokens[i + 1]);
20.                      i += 2;
21.                      while (i < tokens.Count && (tokens[i] == "." ||
22.                      tokens[i] == "," || int.TryParse(tokens[i], out inteiro)))
23.                      {
24.                          novoToken.Append(tokens[i]);
25.                          i++;
26.                      }
27.                      tokens.RemoveRange(inicialNumero, (tokens.Count -
28.                                                          inicialNumero));
29.                      tokens.Insert(inicialNumero, novoToken.ToString().
30.                                                  TrimEnd(new char[] { '.', ',' }));
31.      //O resto do código é apenas o fechando das chaves dos if, for.
```

Após executar o método `IdentificacaoNumeros`, tem-se os seguintes resultados:

a) [O][windows][8][está][custando][R$169,00][reais][que][absurdo];

b) [A][Decor&Cia][possui][uma][loja][com][produtos][ótimos][e][preço][acessivel];

c) [Vídeo][do][novo][windows][http://www.teste.com];

d) [O][filme][novo][do][Thor][não][tem][para][ninguem][,][é][muito][bom][,][obrigado][a][voce][Jaque][que][foi][junto][comigo][assistir].

3.3.2.2.2 *Stopwords*

Para remover as *stopwords* comparou-se o *token* atual com os cadastrados em um dicionário, excluindo os existentes. No Quadro 9, é exibido o método `RemoverStopWords` (membro da classe `PreProcessamento`).

Quadro 9 - Remoção de stopwords

```
01.  public void RemoverStopWords(List<Tweet> tweets)
02.  {
03.      foreach (Tweet tweet in tweets)
04.      {
05.          for (int i=0; i < tweet.Tokens.Count; i++)
06.          {
07.              string token = tweet.Tokens[i];
08.              if (_stopWords.Contains(token.ToLower()))
09.                  tweet.Tokens.RemoveAt(i);
10.          }
11.      }
12.  }
```

Após executar o método de `RemoverStopWords`, tem-se os seguintes resultados:

a) [O][windows][8][está][custando][R$169,00][reais][absurdo];

b) [Decor&Cia][possui][uma][loja][com][produtos][ótimos][e][preço][acessivel];

c) [Vídeo][do][novo][windows][http://www.teste.com];

d) [filme][novo][do][Thor][não][tem][para][ninguem][,][é][muito][bom][,][obrigado] [voce][Jaque][foi][junto][comigo][assistir].

3.3.2.2.3 Sinônimos

A ferramenta possui um cadastro de sinônimos, utilizado para verificar sinônimos de palavras informadas pelo usuário na tela de iniciar processo de MT. Após o usuário informar sua coleção de palavras positivas e negativas, a ferramenta verifica se as mesmas existem, caso existam é concatenado a coleção informado pelo usuário os sinônimos das palavras.

O Quadro 10, apresenta o método `BuscarSinonimos` (membro da classe `PreProcessamento`), que realiza o processo descrito anteriormente.

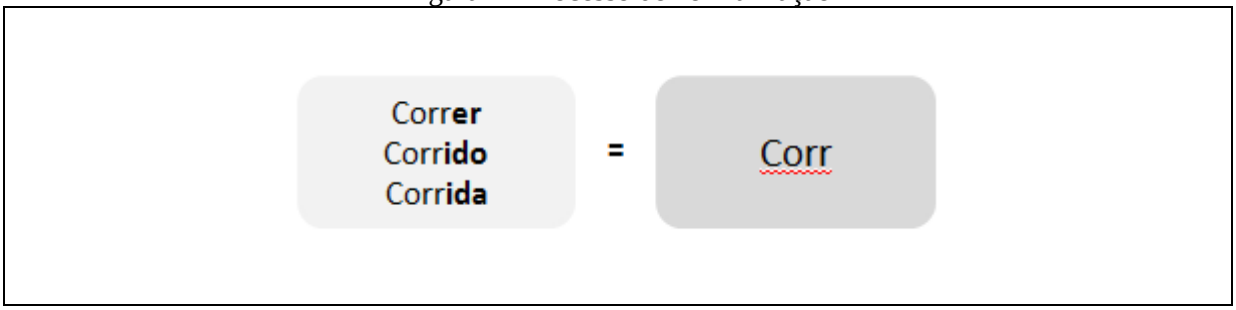
Quadro 10 - Busca de sinônimos

```
01. public void BuscarSinonimos(List<string> adjetivos)
02. {
03.     List<Sinonimos> listaSinonimos = new List<Sinonimos>();
04.     SinonimoRepositorio sinonimoRepositorio = new SinonimoRepositorio();
05.     foreach (string adjetivo in adjetivos)
06.     {
07.         listaSinonimos.AddRange(sinonimoRepositorio.
08.                                                 GetSinonimosAssociados(adjetivo));
09.     }
10.     foreach (Sinonimos sinonimo in listaSinonimos)
11.     {
12.         adjetivos.Add(sinonimo.Nome);
13.     }
14.
15. }
```

3.3.2.2.4 Normalização

Para realizar a normalização, utilizou-se o método de Porter que consiste em identificar a raiz da palavra. Assim, compara-se a parte final do *token* atual com um dicionário pré-estabelecido, eliminando o final dos *tokens* existentes. A ferramenta utiliza o método de Porter por ser de simples implementação. A Figura 7, exibe o processo descrito.

Figura 7 - Processo de normalização



Após o processo de normalização o *token* é validado conforme a quantidade de caracteres, se a mesma for menor que quatro o processo é desfeito e o *token* volta ao seu formato original. O processo de normalização é aplicado sobre as palavras positivas e negativas informadas pelo usuário.

O Quadro 11, apresenta o método `NormalizacaoTokens` (membro da classe `PreProcessamento`), que realiza o processo descrito anteriormente.

Quadro 11 - Normalização das palavras

```
01.  public void NormalizacaoTokens(List<string> palavras)
02.  {
03.      for (int i=0; i < palavras.Count; i++)
04.      {
05.          string palavra = palavras[i];
06.          foreach (string caracteres in _caracteresMetodoPorter)
07.          {
08.              if (palavra.EndsWith(caracteres))
09.              {
10.                  int index = palavra.LastIndexOf(caracteres);
11.                  string radical = palavra.Remove(index, palavra.Length - index);
12.                  if (radical.Length > 3)
13.                  {
14.                      palavras[i] = radical;
15.                      break;
16.                  }
17.              }
18.          }
19.      }
20.
21.  }
```

3.3.2.3 Identificação *tweets* relevantes

Nesta etapa tem-se como objetivo identificar os *tweets* que precisam ser classificados. Para identificar se o *tweet* é relevante para o usuário, verifica-se se o mesmo contém uma palavra positiva ou negativa. Neste processo, os *tweets* são agrupados em dois grupos: os relevantes e os não relevantes, por padrão os *tweets* são considerados como não relevantes. Cada *tweet* é analisado buscando identificar se o mesmo contém uma palavra positiva ou negativa. Se for encontrado uma ou mais ocorrências, ele é caracterizado como relevante. Senão, ele é considerado como não relevante.

No Quadro 12, é apresentado o método `VerificarTweetRelevante` (membros da classe `Indexacao`), que realiza o processo descrito anteriormente.

Quadro 12 - Identificar *tweets* relevantes

```
01. public void VerificarTweetRelevante(List<Tweet> tweets, List<string>
02.                                 palavrasPositivos, List<string> palavrasNegativos)
03. {
04.     foreach (var tweet in tweets)
05.     {
06.         tweet.Relevante = false;
07.         int qtdOcorrencia = OcorrenciaPalavras(adjetivosPositivos, tweet);
08.         if (qtdOcorrencia > 0)
09.             tweet.Relevante = true;
10.         else
11.         {
12.             qtdOcorrencia = OcorrenciaPalavras(adjetivosNegativos, tweet);
13.             if (qtdOcorrencia > 0)
14.                 tweet.Relevante = true;
15.         }
16.     }
17. }
```

No Quadro 13, é apresentado o método `OcorrenciaPalavras` (membro da classe `Indexacao`), que realiza o processo descrito anteriormente.

Quadro 13 - Verificar ocorrência das palavras

```
01. private int OcorrenciaPalavras(List<string> adjetivos, Tweet tweet)
02. {
03.     int qtdOcorrencia = 0;
04.     foreach (string adjetivo in adjetivos)
05.     {
06.         string[] multiplaPalavras = adjetivo.Split(new char[] { ' ' });
07.         if (multiplaPalavras.Length > 1)
08.             qtdOcorrencia += OcorrenciaMultiplasPalavras(tweet,
09.                                                          multiplaPalavras);
10.         else
11.         {
12.             if (_buscarInteriorPalavras)
13.             {
14.                 var listaTokens = tweet.Tokens.Where(t => t == adjetivo);
15.                 qtdOcorrencia += listaTokens.Count();
16.             }
17.             else
18.             {
19.                 var listaTokens = tweet.Tokens.Where(t =>
20.                                                 t.IndexOf(adjetivo) > -1);
21.                 qtdOcorrencia += listaTokens.Count();
22.             }
23.         }
24.         if (qtdOcorrencia > 0)
25.             break;
26.     }
27.     return qtdOcorrencia;
28. }
```

3.3.2.4 Classificação

Nesta etapa será realizado a classificação dos *tweets* em positivos, negativos e indefinidos. Na etapa anterior os *tweets* não relevantes ao usuário foram descartados, sobrando somente os *tweets* relevantes. Inicialmente o processo de classificação soma as

ocorrências das palavras positivas e negativas de cada *tweet*. Se a quantidade for igual, o *tweet* é classificado como indefinido. Senão, o *tweet* é classificado levando em consideração a classe com o maior número de ocorrências.

Para demonstrar como é realizado a classificação dos *tweets* serão utilizadas os seguintes *tweets*: “Primeiros PlayStation 4 vendidos apresentam problemas no HDMI #gamegen”, “PlayStation 4 apresenta gráficos lindos e perfeitos”, “XBOX ONE teve seu lançamento e diversos problemas como PlayStation 4”, “Veja o comando de voz do Playstation 4 em ação, ele é muito rápido”, “PlayStation 4 apresenta o problema da tela azul” e “Encontrada solução para problema no HDMI do PlayStation 4 - http://tinyurl.com/l4m9cjd #gamegen”. As palavras positivas utiliza neste exemplo são: lindo, rápido, perfeito, solução e a palavra negativa é: problema.

No Quadro 14, é apresentado o método `PolaridadePalavras` (membro da classe `Mineração`), que realiza o processo descrito anteriormente.

Quadro 14 - Quantidade de ocorrências das palavras

```
01. private int PolaridadePalavras(List<string> adjetivos, Tweet tweet)
02. {
03.     int qtdOcorrencias = 0;
04.     foreach (string adjetivo in adjetivos)
05.     {
06.         string[] multiplaPalavras = adjetivo.Split(new char[] { ' ' });
07.         if (multiplaPalavras.Length > 1)
08.             qtdOcorrencias += OcorrenciaMultiplasPalavras(tweet,
09.                                                           multiplaPalavras);
10.         else
11.         {
12.             if (_buscarInteriorPalavras)
13.             {
14.                 var listaTokens = tweet.Tokens.Where(t =>
15.                                                      t.IndexOf(adjetivo) > -1);
16.                 qtdOcorrencias += listaTokens.Count();
17.             }
18.             else
19.             {
20.                 var listaTokens = tweet.Tokens.Where(t => t == adjetivo);
21.                 qtdOcorrencias += listaTokens.Count();
22.             }
23.         }
24.     }
25.     return qtdOcorrencias;
26. }
```

No Quadro 15, é apresentado o método `AlgoritmoClassificacao` (membros da classe `Mineração`), que realiza o processo descrito anteriormente.

Quadro 15 - Classificação dos *tweets*

```
01. public void AlgoritmoClassificacao(List<Tweet> tweets)
02. {
03.     foreach (var tweet in tweets)
04.     {
05.         int qtdOcorrPositivo = OcorrenciaPalavras(_adjetivosPositivos, tweet);
06.         int qtdOcorrNegativo = OcorrenciaPalavras(_adjetivosNegativos, tweet);
07.         if (qtdOcorrPositivo > qtdOcorrNegativo)
08.             tweet.Classificacao = TipoTweet.Positivo;
09.         else if (qtdOcorrPositivo < qtdOcorrNegativo)
10.             tweet.Classificacao = TipoTweet.Negativo;
11.         else
12.             tweet.Classificacao = TipoTweet.Indefinido;
13.     }
14. }
```

Após executar o método `AlgoritmoClassificacao` os seguintes resultados foram obtidos, sendo apresentandos na Quadro 16.

Quadro 16 - Classificação dos *tweets* de exemplos

DEMONSTRATIVO DA CLASSICAÇÃO DOS *TWEETS*	
Tweets	**Classificação**
Primeiros PlayStation 4 vendidos apresentam problemas no HDMI #gamegen	Negativo
PlayStation 4 apresenta gráficos lindos e perfeitos	Positivo
XBOX ONE teve seu lançamento e diversos problemas como PlayStation 4	Negativo
Veja o comando de voz do Playstation 4 em ação, ele é muito rápido	Positivo
PlayStation 4 apresenta o problema da tela azul	Negativo
Encontrada solução para problema no HDMI do PlayStation 4 - http://tinyurl.com/l4m9cjd #gamegen	Indefinido

O Quadro 16, demonstra a classificação que ocorre nos *tweets*, conforme palavras definidas como positivas/negativas.

3.3.3 Operacionalidade da implementação

Nesta seção será apresentada a operacionalidade da ferramenta, demonstrando suas principais características. O usuário tem a disponibilidade de realizar as seguintes tarefas, conforme diagrama de caso de uso demonstrado na seção Especificação (3.2):

a) coletar dados do Twitter: inicia o processo de coleta dos dados a partir dos assuntos informados pelo usuário;

b) cadastrar sinônimos: permite ao usuário incluir, alterar e deletar sinônimos;

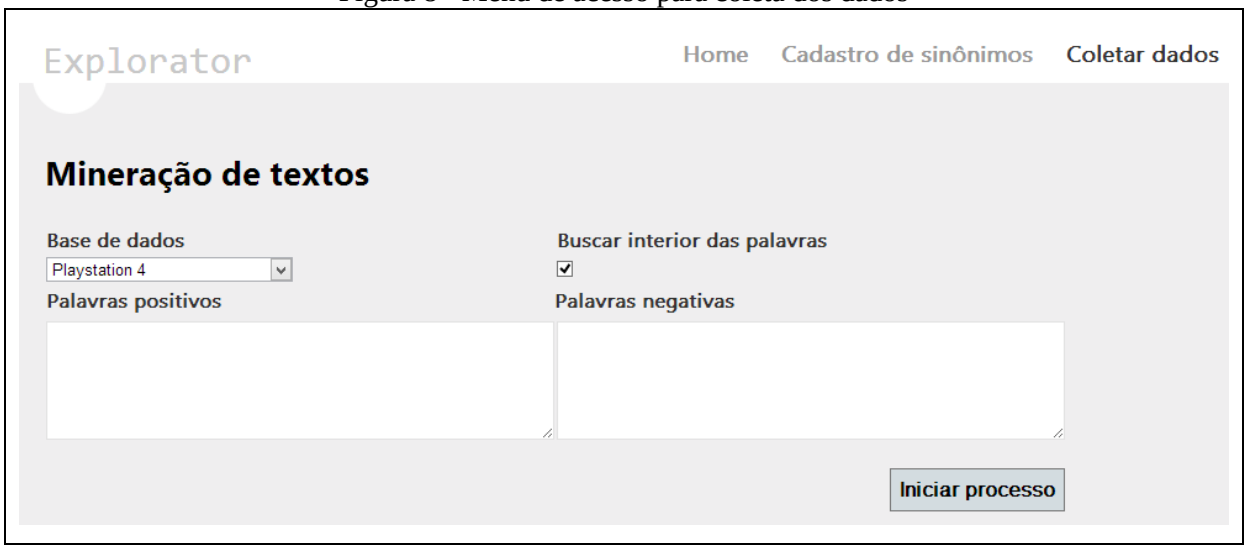
c) iniciar processo de MT: inicia o processo de MT passando pelas etapas de: pré-processamento, indexação e classificação, utilizando as informações preenchidas na tela pelo usuário;

d) visualizar resultados: permite ao usuário visualizar os *tweets* classificados.

3.3.3.1 Coletar dados do Twitter

Inicialmente o usuário precisa selecionar a base de dados do Twitter. Para isto, ele precisa acessar o menu `Coletar dados`, conforme mostra a Figura 8.

Figura 8 - Menu de acesso para coleta dos dados



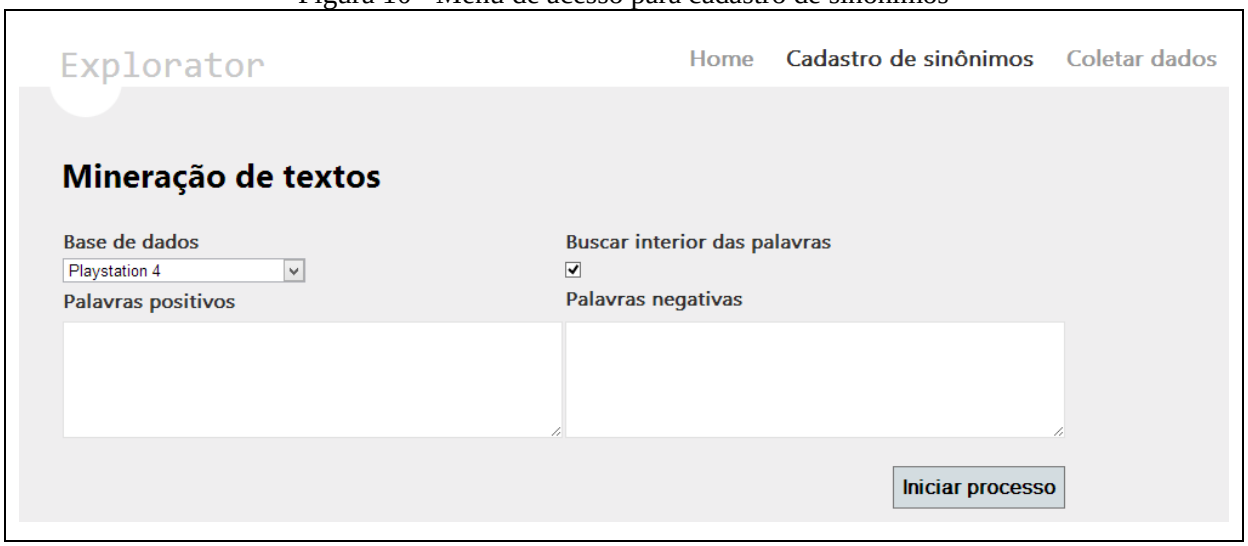
No formulário de coleta de dados, o usuário deve informar o assunto que deseja coletar do Twitter (Figura 9). Após informar o assunto o usuário deve clicar no botão `Iniciar processo`, dando início ao processo de coleta.

Figura 9 - Tela de coleta dos dados

3.3.3.2 Cadastro de sinônimos

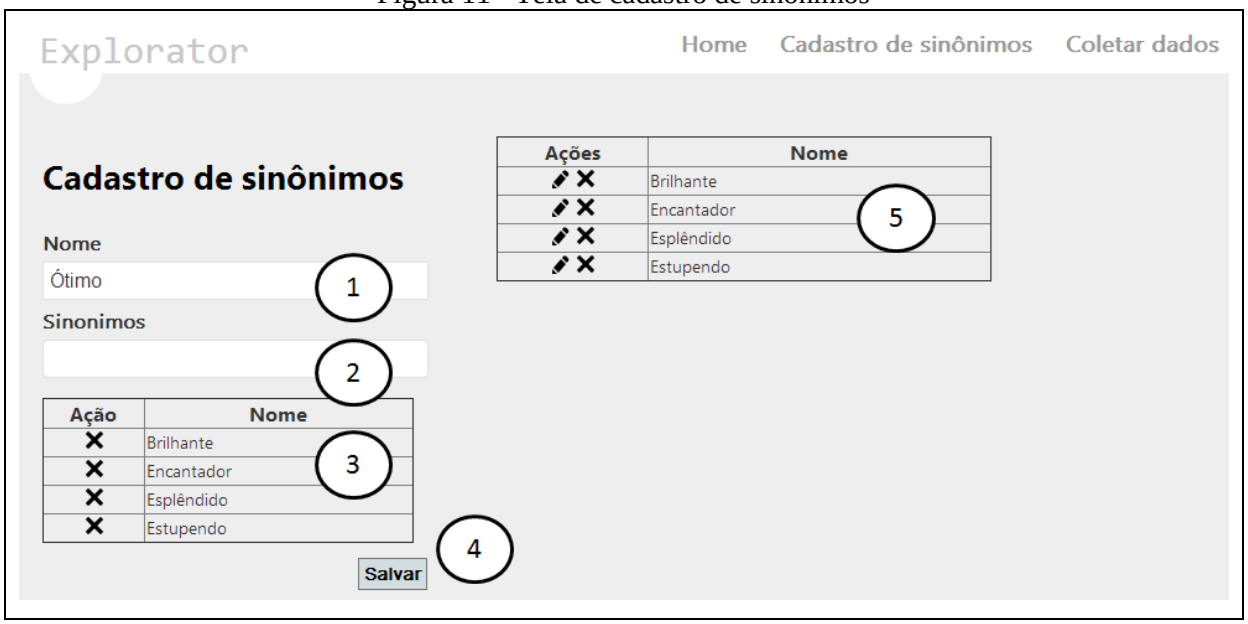
Para efetuar a inclusão, alteração ou exclusão de algum sinônimo, o usuário deve selecionar, no menu superior da aplicação, a opção `Cadastro de sinônimos`, conforme exibido na Figura 10.

Figura 10 - Menu de acesso para cadastro de sinônimos



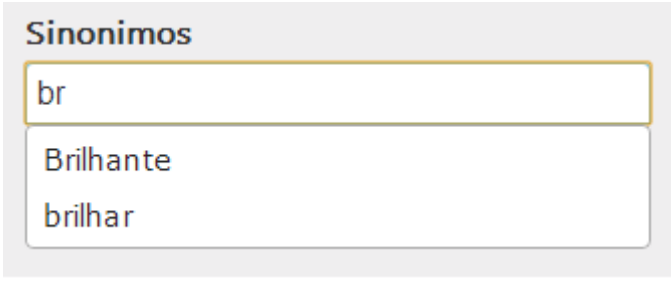
Após selecionar o menu em questão, a tela de cadastro de sinônimos é exibida, como ilustra a Figura 11.

Figura 11 - Tela de cadastro de sinônimos



Para cadastrar um novo sinônimo, o usuário deve informar os campos nome e sinônimos (itens 1 e 2 da Figura 11), conforme é apresentado na Figura 12.

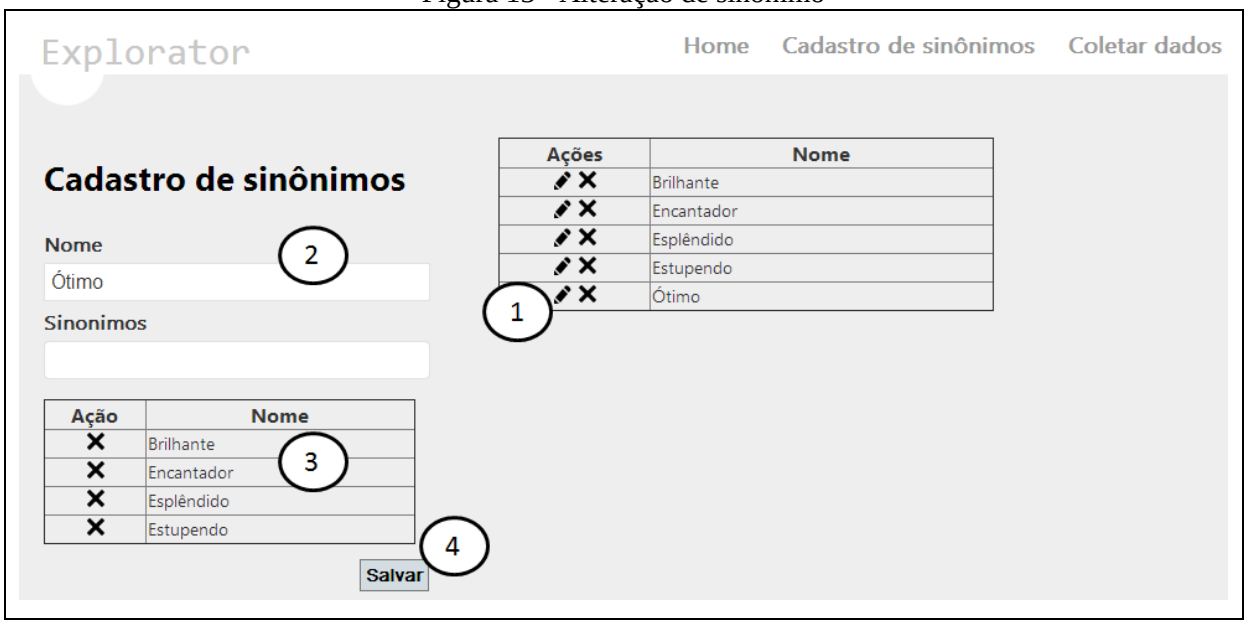
Figura 12 - Lista de sinônimos



Os itens selecionados no campo sinônimos (item 2 da Figura 11) são listados em um *grid* identificado no item 3 da Figura 11. Após realizar o preenchimento dos campos, o usuário deve clicar no botão `Salvar` (item 4 da Figura 11). Ao pressionar o botão `Salvar`, uma mensagem informativa de sucesso ou erro será apresentada.

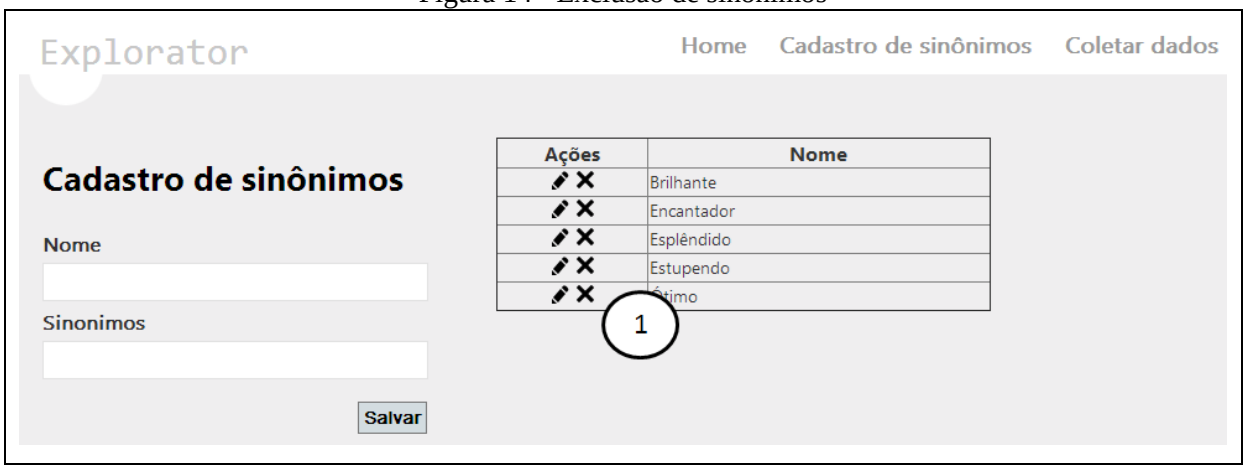
Para alterar os sinônimos basta pressionar o botão em formato de lápis, conforme pode-se observar no item 1 da Figura 13. O sinônimo selecionado será carregado no formulário, onde poderá ser alterado (itens 2, 3 na Figura 13). Após efetuar as alterações, o usuário deverá pressionar o botão `Salvar`, representado pelo item 4 na Figura 13. Ao pressionar o botão `Salvar`, uma mensagem de sucesso ou erro pode ser apresentada.

Figura 13 - Alteração de sinônimo



Para excluir um sinônimo basta pressionar o botão em formato de X, conforme pode-se observar no item 1 da Figura 14.

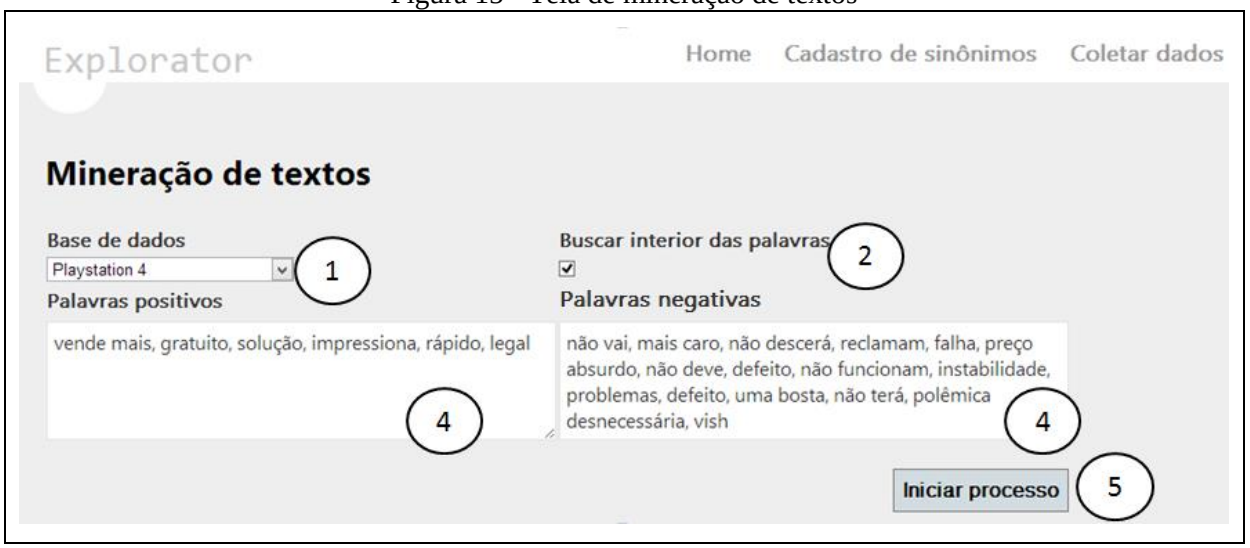
Figura 14 - Exclusão de sinônimos



3.3.3.3 Iniciar processo de MT

Para inicar o processo de MT, o usuário deverá preencher os campos base de dados, adjetivos positivos, adjetivos negativos e buscar interior das palavras (itens 1, 2, 3 e 4 da Figura 15). Para iniciar a mineração o usuário deverá pressionar o botão `Iniciar processo` conforme é exibido na Figura 15.

Figura 15 - Tela de mineração de textos

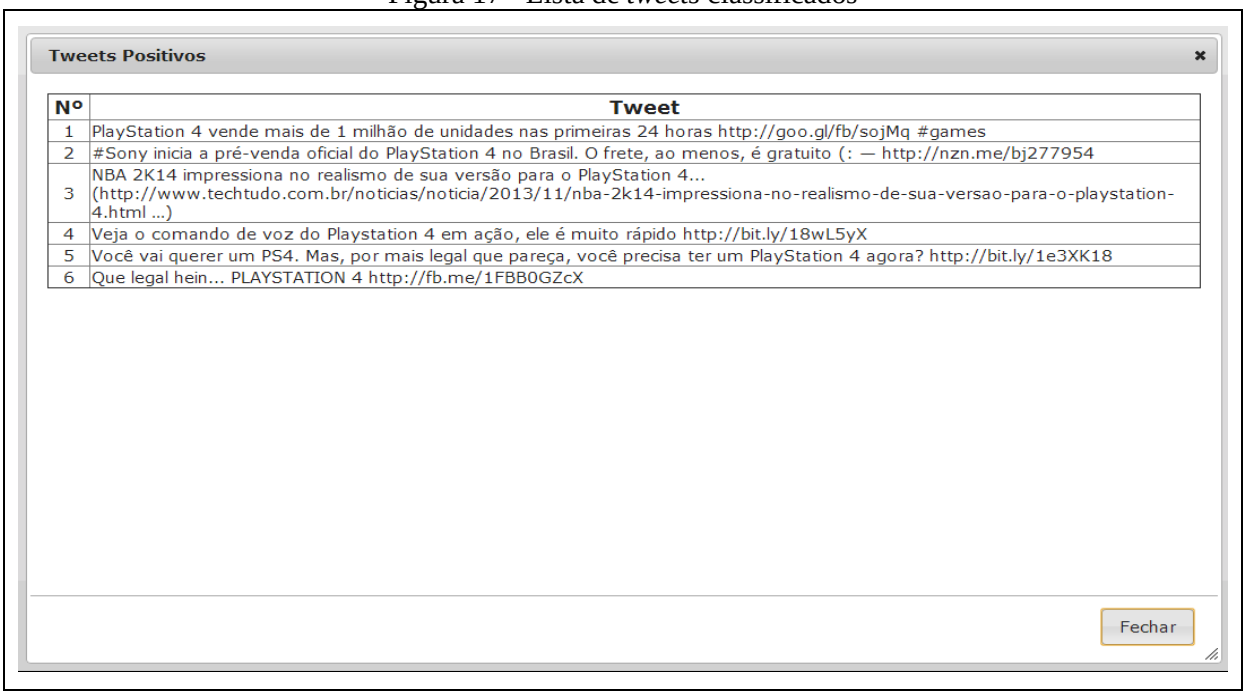


Depois de pressionar o botão `Iniciar Processo` (item 5 da Figura 15), aparecerá algumas caixas que indicam a quantidade de *tweets* positivos, negativos e indefinidos, conforme apresentado no item 1 da Figura 16.

Figura 16 - Resultados da MT

Para visualizar os *tweets* classificados, o usuário deverá pressionar as caixas correspondentes ao resultados, conforme pode-se observar no item 1 da Figura 16. Os resultados são apresentados na forma de relatório, conforme mostra a Figura 17.

Figura 17 - Lista de *tweets* classificados



3.4 RESULTADOS E DISCUSSÃO

Nesta seção serão apresentados os experimentos realizados, bem como os resultados obtidos. Foram realizados três experimentos onde cada um possui uma base de dados diferente. A seção 3.4.1, apresenta os resultados obtidos no experimento 1, a seção 3.4.2, exibe os resultados obtidos com o experimento 2 e a seção 3.4.3, demonstra o experimento 3 e os resultados obtidos.

Os resultados obtidos pela ferramenta foram comparados com os resultados alcançados/realizados de forma manual. Para realizar a classificação manual, analisou-se o texto do *tweet* para identificar se o autor expressava uma opnião positiva ou negativa em relação ao assunto definido pelo usuário.

3.4.1 Experimento 1

O experimento 1, utiliza a base de dados nomeada como “Thor.txt”, tendo como assunto principal o filme “Thor: O Mundo Sombrio”, lançado nos cinemas dia 01/11/2013. Foram capturados 101 *tweets*.

No primeiro teste o campo “Buscar interior das palavras” ficou marcado e os campos de palavras positivas e negativas receberam as seguintes informações:

a) positivo: ver, lindo, primeiro, assistir, obrigado, gostei, melhor, lidera, delicia, fofinho, massa,1º lugar, largar, perfeito, boa, topo, preferidos, força, legal, foda e favoritos;

b) negativo: causa, sofrendo, dramalhão, bicha, odeio, idiota e não iamos.

Os resultados obtidos neste experimento são apresentados na Tabela 1.

Tabela 1 - Resultados do experimento 1 teste um

RESULTADOS DO EXPERIMENTO 1 (TESTE 1)		
Classificação	**Classificados manualmente**	**Explorator**
Positivo	45	53
Negativo	5	3
Indefinido	-	2

Como é possível observar na Tabela 1, os *tweets* classificados pela ferramenta apresentam uma divergência com relação a manual. O percentual de acerto neste teste foi de 86,87%. O Quadro 17, apresentada os *tweets* que não deveriam ser classificados como positivos.

Quadro 17 - *Tweets* classsificados errados do experimento 1

a) "Thor teve uma melhora: nao teve convulsões essa noite. :)";

b) "Gostei de um vídeo @YouTube de @depphiddles http://youtu.be/XJhL1cauUJA?a Who's Better: Thor or Loki? [LEGENDADO]";

c) "Alguém sabe se eu assistir thor o mundo sombrio sem ter assistido ao primeiro filme eu vou ficar boiando?";

d) "Agents of SHIELD: veja uma cena completa do crossover com Thor • http://ift.tt/I1fM9y";

e) "BOMDIAAAA <3 Hoje é dia de resolver MUITAS COISAS e do Thor ficar cheiroso *-* já deixei ele no banho \o/".

No Quadro 17, os *tweets* "a", "b" e "c" foram classificados como positivos, pois a identificação das palavras se faz pelo campo palavra positiva. Os *tweets* "d" e "e", tiveram a palavra "ver" encontrada na mineração, sendo que estas palavras inteiras correspondem a "crossover" e "resolver". A ferramenta verifica se a *string*, por exemplo "ver", está dentro de outra *string*, por exemplo "resolver". Se estiver o *tweet* é classificado como positivo.

Este teste foi refeito desmarcando o campo `Buscar interior das palavras`. Os resultados obtidos são apresentandos na Tabela 2.

Tabela 2 - Resultados do experimento 1 teste dois

RESULTADOS DO EXPERIMENTO 1 (TESTE 2)		
Classificação	**Classificados manualmente**	**Explorator**
Positivo	45	40
Negativo	5	4
Indefinido	-	1

Como é possível verificar na Tabela 2, o percentual de acerto diminuiu para 81,82%. Neste teste considerou-se a ocorrência exata da palavra e sem a normalização da palavra.

Entretanto, alguns *tweets* não foram recuperados. Porém, o problema apresentando anteriormente do *tweets* “d” e “e”, não aconteceu.

Realizou-se um terceiro teste, alterando apenas as palavras positivas. Foram adicionadas as seguintes palavras: assistimos, assistindo, assisti, liderança, lindinho, primeira, gosto e obrigada. Os resultados obtidos nesta execução são apresentados na Tabela 3.

Tabela 3 - Resultados do experimento 1 teste três

RESULTADOS DO EXPERIMENTO 1 (TESTE 3)		
Classificação	**Classificados manualmente**	**Explorator**
Positivo	45	50
Negativo	5	3
Indefinido	-	2

Como é possível verificar na Tabela 3, chegou-se a um percentual de acerto de 89,9%. Com este teste, conclui-se que a escolha das palavras influencia nos resultados.

3.4.2 Experimento 2

No experimento 2, utiliza-se a base de dados nomeada como “Submarino.txt”, tendo como assunto principal o loja: “Submarino”. Foram capturados 100 *tweets*.

No primeiro teste o campo `Buscar interior das palavras` ficou marcado e os campos de palavras positivas e negativas receberam as seguintes informações:

a) positivo: uhul, por apenas, felicidade, fiel, demais, baratinho, quero, promoção, fantástica, desconto, menor preço, interessante e oferta;
b) negativo: afundar, chorando, aguentar, não vão, somente, não vende, sofre, palhaçada, não mesmo, puta, não tem, amassada, quebrada, problema e desonestas.

Os resultados obtidos nesta execução do teste 1 são apresentados na Tabela 5.

Tabela 4 - Resultados do experimento 2 teste um

RESULTADOS DO EXPERIMENTO 2 (TESTE 1)		
Classificação	**Classificados manualmente**	**Explorator**
Positivo	38	38
Negativo	15	12
Indefinido	-	4

Como é possível observar na Tabela 4, os valores não possuem grande divergências, tendo um percentual de acerto de 94%. No Quadro 18, são apresentados os *tweets* classificados indevidamente como indefinidos.

Quadro 18 - *Tweets* classsificados errados do experimento 2

a) “Que incrível. Eu to torcendo pra ter uma super oferta na Black Friday. SE É QUE o site do Submarino vai aguentar desta vez.”;
b) “Ainda não tem os meus livros? O Submarino fez uma promoção e o

kit com os dois está custando R$39,90.… http://instagram.com/p/g3I_JOrtqG/";

c) "Submarino.com.br Quem não sofreu com Caverna do Dragão, não sabe o que é angustia. hehehe. Aproveite a oferta WOW de hoje! http://bit.ly/1cIkB4X";

d) "OI @SUBMARINO RECEBI UM EMAIL DE VOCES MAS ACHO QUE TA COM PROBLEMA, DIZ ALI SUPER OFERTA HD 1TERA POR 279, ESSE É O PREÇO NORMAL DELE".

No Quadro 18, os *tweets* foram classificados como indefinidos porque apresentam uma palavra positiva e outra negativa.

Em um segundo teste, utilizou-se os mesmos valores para os campos positivos e negativos, somente alterando o campo `Buscar interior das palavras` para desmarcado, os resultados obtidos são apresentandos na Tabela 5.

Tabela 5 - Resultados do experimento 2 teste dois

RESULTADOS DO EXPERIMENTO 2 (TESTE 2)		
Classificação	**Classificados manualmente**	**Explorator**
Positivo	38	31
Negativo	15	11
Indefinido	-	3

Como é possível verificar na Tabela 5, os resultados neste teste foram satisfatórios tendo um percentual de acerto de 87%.

Foi realizado um terceiro teste, alterando somente o campo de palavras positivas, adicionando as seguintes palavras: promoções, descontos, ofertas e queria. Os resultados obtidos nesta execução estão apresentados na Tabela 6.

Tabela 6 - Resultados do experimento 2 teste três

RESULTADOS DO EXPERIMENTO 2 (TESTE 3)		
Classificação	**Classificados manualmente**	**Explorator**
Positivo	38	35
Negativo	15	11
Indefinido	-	3

Como é possível observar na Tabela 6, chegou-se a um percentual de 91% de acerto.

3.4.3 Experimento 3

O experimento 3, utiliza a base de dados nomeada como "PS4.txt" e a mesma está relacionado ao assunto: "Playstation 4" . Foram capturados 100 *tweets*.

No primeiro teste o campo `Buscar interior das palavras` ficou marcado e os campos de palavras positivas e negativas receberam as seguintes informações:

a) positivo: vende mais, gratuito, solução, impressiona, rápido e legal;
b) negativo: não vai, mais caro, não descerá, reclamam, falha, preço absurdo, não deve, defeito, não funcionam, instabilidade, problemas, defeito, uma bosta, não terá, polêmica desnecessária e vish.

Os resultados obtidos nesta execução do teste um são apresentados na Tabela 7.

Tabela 7 - Resultados do experimento 3 teste um

RESULTADOS DO EXPERIMENTO 3 (TESTE 1)		
Classificação	**Classificados manualmente**	**Explorator**
Positivo	6	6
Negativo	22	22
Indefinido	-	1

Como é possível observar na Tabela 7, os *tweets* classificados pela ferramenta apresentam um percentual de 98% de acerto. No Quadro 19, são apresentados *tweets* que foram classificados errados.

Quadro 19 - *Tweets* classsificados errados do experimento 3

a) "Encontrada solução para problema no HDMI do PlayStation 4 - http://tinyurl.com/l4m9cjd #gamegen";
b) "Você vai querer um PS4. Mas, por mais legal que pareça, você precisa ter um PlayStation 4 agora? http://bit.ly/1e3XK18";

No Quadro 19, o *tweet* "a" foi classificado como indefinido pois contém uma palavra positiva e outra negativa. O *tweet* "b" por sua vez está classificado como positivo, mas o mesmo é uma pergunta, então devia ser descartado.

Realizou-se um segundo teste utilizando os mesmos valores para os campos positivos e negativos, apenas alterando o campo `Buscar interior das palavras`, os resultados obtidos são apresentandos na Tabela 8.

Tabela 8 - Resultados do experimento 3 teste dois

RESULTADOS DO EXPERIMENTO 3 (TESTE 2)		
Classificação	**Classificados manualmente**	**Explorator**
Positivo	6	7
Negativo	22	20
Indefinido	-	0

Como é possível verificar na Tabela 8, neste experimento alcançou-se o percentual de acerto de 97%.

Realizando um terceiro teste, alterando somente o campo de palavras negativas, incrementando com as seguintes palavras: problema, reclamando e não funcionam. Os resultados obtidos nesta execução estão apresentados na Tabela 9.

Tabela 9 - Resultados do experimento 3 teste três

RESULTADOS DO EXPERIMENTO 3 (TESTE 3)		
Classificação	**Classificados manualmente**	**Explorator**
Positivo	6	6
Negativo	22	22
Indefinido	-	1

Como é possível observar na Tabela 9, os *tweets* classificados pela ferramenta apresentam um percentual de acerto de 98%.

3.4.4 Discussão dos resultados

O Quadro 20 apresenta de forma resumida as principais características dos trabalhos relacionados com as da ferramenta desenvolvida.

Quadro 20 - Característica dos trabalhos relacionados com esta ferramenta

características / trabalhos relacionados	SANTOS (2010)	RAMOS E BRÄSCHER (2009)	TORRES (2005)	SILVA E. (2010)	Explorator (2013)
técnica de agrupamento	-	X	-	-	-
técnica de classificação	*SVM*	-	Naive Bayes	*KnnFlex*	X
ferramentas de auxilio	-	SAS	-	R/tm	-
remoção *stopwords*	X	X	X	X	X
normalização	X	X	X	X	X
sinônimos	-	-	X	-	X
identificação de abreviaturas	-	-	-	-	X
identificação de palavras combinadas	-	-	-	-	X

A partir do Quadro 20, pode-se verificar que todos os trabalhos utilizão as técnicas de *stopwords* e normalização na etapa de pré-processamento, sendo que este projeto e o do Torres (2005), utilizaram a técnica de sinônimos para melhora a eficiencia desta etapa. Este projeto para melhorar o pré-processamento utilizou as técnicas de identificação de abreviaturas e palavras combinadas. Diferente dos trabalhos de Ramos e Bräscher (2009), Silva E. (2010), este projeto não utilizou nenhuma ferramenta para auxiliar nesta etapa.

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de uma ferramenta que implementa algumas das as etapas da mineração de textos, sendo elas: coleta, pré-processamento e análise da informação. Para realizar o pré-processamento, utilizou-se as técnicas de tokenização, *stopwords*, sinônimos, normalização que possuem o objetivo de limpar os dados e aumentar a qualidade dos mesmo. Utilizou-se palavras positivas e negativas informadas pelo usuário para classificar os *tweets* relevantes em positivo, negativo ou indefinido. Por fim, a etapa de análise das informações, apresenta os *tweets* classificados para o usuário.

A ferramenta foi implementada utilizando a linguagem de programação C# da plataforma .NET, fazendo uso das bibliotecas LinqToTwitter e JQuery, auxiliando na coleta dos *tweets* do Twitter e manipulação dos campos html, respectivamente. A ferramenta possui uma interface web para que o usuário possa realizar as seguintes operações: cadastrar sinônimos, iniciar processo de coleta dos dados, iniciar processo de MT e visualizar a classificação dos *tweets.*

Os objetivos deste trabalho foram alcançados, onde a ferramenta apresentou um percentual de acerto acima de 80%. Entretanto, o sucesso na classificação depende do conhecimento que o usuário possui sobre a base, pois se o usuário informar palavras positivas e negativas que não existam na base de dados, os *tweets* não serão classificados.

4.1 EXTENSÕES

Algumas sugestões de extensões deste trabalho são listadas a seguir:

a) disponibilizar a coleta dos dados para outras redes sociais, exemplo: Facebook, Linkedin;
a) adicionar novos recursos ao pré-processamento como, por exemplo: correção ortográfica, redução do léxico, detecção automática de sinônimos, nomes truncados e *parsing* (Análise sintática);
b) utilizar outras técnicas de normalização, por exemplo: método de *Stemmer* S, método de *Lovins* e *Lemmatization*;
c) análise da base de dados antes de iniciar processo de MT, para identificar novas *stopwords*;
d) criar serviços que disponibilizam a etapa de pré-processamento, indexação e classificação. Assim o usuário pode utilizar somente a etapa que melhor lhe atende;
e) criar novas interfaces de acesso, por exemplo: *mobile*, *desktop*.

f) criar uma interface onde o usuário pode informar novas *stopwords*, palavras normalizadas, palavras abreviadas, palavras combinadas;

REFERÊNCIAS BIBLIOGRÁFICAS

ARANHA, Christian N. **Uma abordagem de pré-processamento automático para mineração de textos em português**: sob o enfoque da inteligência computacional. 2007. 144 f. Tese (Doutorado em Engenharia Elétrica) - Programa de Pós-Graduação em Engenharia Elétrica, Pontifíca Universidade Católica do Rio de Janeiro, Rio de Janeiro.

ARANHA, Christian; PASSOS, Emmanuel. A tecnologia de mineração de textos. **RESI - Revista Eletrônica de Sistemas de Informação**, Rio de Janeiro, v. 5, n. 2, 2006. Disponível em: <http://revistas.facecla.com.br/index.php/reinfo/article/viewFile/171/66>. Acesso em: 11 dez. 2013.

BRACKETT, Michael H. **The data warehouse challange**: taming data chaos. New York: John Wiley & Sons, 1996.

CARRILHO JUNIOR, João R. **Desenvolvimento de uma metodologia para mineração de textos**. 2007. 96 f. Dissertação (Mestre em Engenharia Elétrica) - Programa de Pós-Graduação em Engenharia Elétrica, Pontifíca Universidade Católica do Rio de Janeiro, Rio de Janeiro.

CASTANHEIRA, Luciana G. **Aplicação de técnicas de mineração de dados em problemas de classificação de padrões**. 2008. 91 f. Dissertação (Mestrado em Engenharia Elétrica) - Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte.

FERREIRA, E. B. A. **Analise de sentimento em redes sociais utilizando influência das palavras**. 2010. 59 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife.

GOLDSCHMIDT, Ronaldo; PASSOS, Emmanuel. **Data mining**: um guia prático. Rio de Janeiro: Elsiever, 2005.

LAROSE, Daniel T. **Discovering knowledge in data**: an introduction to data mining. New Jersey: John Wiley & Sons, 2005.

MONTEIRO, Lêda O.; GOMES, Igor R.; OLIVEIRA, Thiago. Etapas do processo de mineração de textos - uma abordagem aplicada a textos em português do Brasil. In: CONGRESSO DA SBC, 26., 2006, Campo Grande. **Anais...** Campo Grande: SBC, 2006. p. 78-81. Disponível em: <http://www.natalnet.br/sbc2006/pdf/arq0166.pdf>. Acesso em: 11 dez. 2013.

RAMOS, Hélia S. C.; BRÄSCHER, Marisa. Aplicação da descoberta de conhecimento em textos para o apoio à construção de indicadores infométricos para a área de C&T. **Ciência da Informação**, Brasília, v. 38, n. 2, p. 56-86, mai./ago. 2009. Disponível em: <http://www.scielo.br/pdf/ci/v38n2/05.pdf>. Acesso em: 11 dez. 2013.

SANTOS, Carlos E. **Data mining aplicado a campanhas de marketing**. 2008. 86 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Departamento de Ciências Exatas e Tecnológicas, Centro Universitário Feevale, Novo Hamburgo.

SANTOS, Leandro M. **Protótipo para mineração de opinião em redes sociais**: estudo de casos selecionados usando o twitter. 2010. 102 f. Trabalho de Conclusão de Curso (Bacharelado de Ciência da Computação) - Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras.

SIDNEY, Christiane F. **Aplicação de mineração de dados no banco de dados do zoneamento ecológico de Minas Gerais**. 2010. 60 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas da Informação) - Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras.

SILVA, Elcelina R. C. C. **Técnicas de data e text mining para anotação de um arquivo digital**. 2010. 89 f. Dissertação (Mestrado em Engenharia Electrónica e Telecomunicações – Especialização Sistemas de Informações) – Departamento de Electrónica Telecomunicações e Informática, Universidade de Aveio, Aveiro.

SILVA, Nelson G. R. **BestChoice: classificação de sentimentos em ferramentas de expressão de opinião**. 2010. 45 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife.

TORRES, José A. S. **Proposta de arquitetura para construção de perfis de usuário através da mineração da web**. 2005. 88 f. Trabalho de Conclusão de Curso (Bacharelado em Análise de Sistemas) – Departamento de Ciências Exatas e da Terra, Universidade do Estado da Bahia, Salvador.

TWITTER. **Streaming API Documentation**. [S.I.], 2012. Disponível em: <http://dev.twitter.com/docs/streaming-apis>. Acesso em: 11 dez. 2013.

UBER, José L. **Descoberta de conhecimento com o uso de text mining aplicada ao SAC**. 2004. 54 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

VIERA, Sylvio A. G. **Aplicação de máquinas de vetores de suporte na investigação da atividade gênica do câncer do colo de intestino**. 2011. 77 f. Dissertação (Mestrado em Nanociências) - Pró-Reitoria de Pós-Graduação, Pesquisa e Extensão, Centro Universitário Franciscano, Santa Maria.