ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Paulo Henrique de Sousa

DESENVOLVIMENTO DE UM PROTÓTIPO DE CLASSIFICAÇÃO AUTOMÁTICA DE OPINIÕES

Palmas - TO

2013

Paulo Henrique de Sousa

DESENVOLVIMENTO DE UM PROTÓTIPO DE CLASSIFICAÇÃO AUTOMÁTICA DE OPINIÕES

Trabalho de Conclusão de Curso (TCC) elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.Sc. Fernando Luiz de Oliveira.

Palmas - TO

2013

DESENVOLVIMENTO DE UM PROTÓTIPO DE CLASSIFICAÇÃO AUTOMÁTICA DE OPINIÕES

Trabalho de Conclusão de Curso (TCC) elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.Sc. Fernando Luiz de Oliveira.

Aprovada em: junho de 2013.

BANCA EXAMINADORA

Prof. M.Sc. Cristina D'Ornellas Filipakis Souza

Centro Universitário Luterano de Palmas

Prof. M.Sc. Fabiano Fagundes

Centro Universitário Luterano de Palmas

Prof. M.Sc. Fernando Luiz de Oliveira

Centro Universitário Luterano de Palmas

Palmas - TO

2013

Uma passagem acadêmica tão importante e especial como eu considero a minha, não seria possível, senão, através da influência de pessoas admiráveis que fizeram e fazem de suas vidas, exemplos de força e coragem a serem seguidos. Por isto, dedico este trabalho aos meus pais, Rubens Ribeiro e Maria Rita Lopes, aos quais devo todas as minhas conquistas pessoais e profissionais, principalmente a conclusão desta graduação. Nenhuma pessoa me apoiou tanto quanto vocês nos desafios que me foram impostos.

Dedico esta vitória também aos meus avôs Paulino Lopes, Rosa Lopes e Noemes Ribeiro, pela admiração que tenho por cada um e por terem me ajudado tanto na conclusão deste curso. Sinto orgulho de ser o primeiro, dentre os netos e netas de vocês, a concluir a graduação.

AGRADECIMENTOS

Agradeço a Deus por me tornar capaz de saber o quão bom é viver o momento que estou vivendo. Obrigado Senhor, por ter me dado forças para finalizar este curso numa área pela qual tenho tanto apreço e que me permitiu conhecer tantos amigos dedicados e inteligentíssimos.

Agradeço aos meus familiares por todo o diálogo e auxílio oferecido para me fazer entender coisas que dificilmente se aprende de forma sutil mundo afora. Agradeço principalmente aos meus pais por todo o amor e apoio, e por terem me permitido estudar no curso que eu desejava. Agradeço aos meus irmãos, Victor Job e Cinara Kariny, pela amizade verdadeira construída ao longo dos anos, nos quais compartilhamos muitos momentos prazerosos e felizes. Cinara, não é segredo que você pela boa comunicação, pela força de vontade e mania de organização, sempre influenciou positivamente na minha vida. Victor, eu te admiro pela sua bondade e pela sua alegria que muito me contagia a cada vez que conversamos.

Agradeço à minha namorada, Hedielsa Niágara, por todo o amor, por toda a vontade em me ajudar e por ser uma pessoa tão especial e presente na minha vida. Niágara, agradeço por cada segundo que você tem estado comigo.

Agradeço também ao Fernando Luiz de Oliveira que teve papel fundamental na minha formação e por ser o meu orientador das disciplinas de TCC I e de TCC II. Tenho consideração por esse excelente professor que, além do conhecimento, sabe o momento certo de fazer certas brincadeiras para animar os seus alunos.

Agradeço ao amigo Michael Schuenck dos Santos, meu colega de trabalho, que também já foi meu professor, por toda a atenção, dedicação e disposição em ajudar sempre que possível. Não existe uma forma fácil de agradecê-lo por toda a motivação que me foi transmitida.

Agradeço à Parcilene Fernandes de Brito pela amizade e pelas melhores aulas que eu já assisti nas disciplinas de Gestão Tecnológica e de Sociedade e Contemporaneidade. É admirável a excelência profissional e o compromisso com o ensino adotado por ela.

Agradeço também aos demais professores e professoras, Madianita Bogo, Fabiano Fagundes, Cristina D'Onellas Filipakis, Jackson Gomes e Edeilson Milhomem, por tanto terem me ensinado nestes últimos anos, inclusive fora da sala de aula.

Agradeço também aos amigos Rafael Gonçalves e Neuziron Aguiar, meus colegas de turma, por toda a amizade, por compartilharem vários momentos difíceis que enfrentamos e por todas as vitórias que foram conquistadas em conjunto.

RESUMO

As mídias sociais são meios de comunicação muito utilizados pela sociedade para demonstrar satisfação e insatisfação sobre empresas, marcas e produtos. Existem inúmeras mídias sociais disponíveis na *web* dentre as quais pode-se citar o *Twitter*, o *Facebook* e o *Google Plus*. Estas mídias recebem diariamente diversas postagens contendo opiniões sobre assuntos do interesse de seus usuários. Tal situação torna oportuno o interesse das instituições em analisar o conteúdo disponível na *web.* Geralmente, o resultado da análise destas opiniões provê vantagem competitiva para as empresas. Mas, existe uma dificuldade em analisar muitas opiniões. Com a finalidade de analisar um conjunto de opiniões que podem ser classificadas automaticamente como favorável ou não, a classificação automática de opiniões tem se tornado uma área promissora oferecendo recursos importantes para as empresas. Com o objetivo de prover um mecanismo de mineração de opiniões que pode ser adotado pelas empresas, este trabalho apresenta conceitos na área de mineração de dados e nas suas subáreas. O estudo destes conceitos possibilitou o desenvolvimento de um protótipo de classificação automática de opiniões que é apresentado neste trabalho.

PALAVRAS-CHAVE: MINERAÇÃO DE DADOS, CLASSIFICAÇÃO DE OPINIÕES, ANÁLISE DE MÍDIAS SOCIAIS.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API – Application Programming Interface

ARFF – Attribute-Relation File Format

DM – Data Mining

DW – Data Warehouse

ETL – Extract, Transform, Load

IE – Information Extraction

GPL – General Public License

IR – Information Retrieval

JSON – JavaScript Object Notation

JVM – Java Virtual Machine

KDD – Knowledge Discovery in Databases

KDT – Knowledge Discovery in Textual *Databases*

SUMÁRIO

INTRODUÇÃO

Existem inúmeros documentos e páginas compartilhadas na web abordando os mais variados contextos e assuntos. Tais registros são armazenados e devem ser aproveitados como fonte de informação nas tomadas de decisão das pessoas e empresas. No entanto, devido ao grande volume de dados, muitas informações ficam ocultas e não são utilizadas devidamente, o que dificulta a descoberta de relações comerciais para ganhar vantagem competitiva.

Os gestores, ao identificarem a existência de tamanha quantidade de conteúdo disponível na web, passaram a analisar a possibilidade de obter informações relevantes neste meio e que pudessem indicar a necessidade de mudanças em seus negócios. Contudo, analisar todo esse contingente de dados não é uma tarefa simples e, por isso, não é possível fazê-lo sem o auxílio de tecnologias capazes de processar os dados em tempo hábil, mantendo a veracidade das informações.

A web é muito utilizada para divulgar opiniões de produtos e marcas, favoráveis ou desfavoráveis, produzidas pela própria empresa, como também por seus clientes. Por isso, é importante categorizar tais opiniões para observar a visão das pessoas em relação aos produtos e serviços comerciais. É necessário saber se uma opinião qualquer é favorável ou não a um determinado produto. Sabe-se ainda que grandes empresas já têm funcionários atuando na função de analista de mídias sociais (pessoas que trabalham examinando o que é dito sobre a empresa na web, principalmente nas mídias sociais), sendo estes os responsáveis pela intermediação da empresa com a comunidade na *web*, utilizando, para isto, ferramentas de buscas na internet, assim como o Bing e o Google.

Por causa da quantidade de conteúdo disponível na web, a análise de documentos textuais deixou de ser realizada apenas pela capacidade de discernimento humano e passou a ser realizada de forma automatizada. Tal processo fez com que a análise contasse com o auxílio de máquinas que proporcionaram um cruzamento de dados mais abrangente. Assim, a intenção de extrair informações importantes em dados textuais colocou em evidência o conceito de "*Text Mining"* que, segundo Sullivan (2000, online) é "(...) o estudo e a prática de extrair informação de textos usando os princípios da linguística computacional".

As técnicas de mineração de textos têm sido utilizadas em diversas áreas do conhecimento. Porém, mesmo com o auxílio de ferramentas capazes de processar a enorme

quantidade de dados textuais disponíveis, existe um problema com a questão da Recuperação da Informação (RI). Isto porque nem sempre é possível recuperar informações relevantes a partir do(s) parâmetro(s) de pesquisa utilizado(s) na busca de conhecimento em dados textuais.

Geralmente, as pessoas necessitam de meios que auxiliem na avaliação dos itens de consumo. Devido à quantidade e à peculiaridade dos produtos disponíveis no mercado é necessário buscar opiniões de terceiros para obter uma concepção sobre o quão apropriado um determinado produto ou marca é para o consumidor. Deve existir uma forma rápida para verificar se uma empresa e/ou mercadoria estipulada é aprovada por outros consumidores, destacando principalmente, as vantagens e as desvantagens.

Além da necessidade dos clientes, as empresas necessitam saber sobre a visão dos consumidores em relação a elas e em relação aos produtos que elas oferecem. Sabe-se que o mercado é competitivo e, por isso, é imprescindível a existência de meios para identificar as adversidades comerciais. Em vista disso, um sistema que avalie várias opiniões sobre uma mercadoria qualquer, constatando se são favoráveis ou desfavoráveis poderá auxiliar nas tomadas de decisão das empresas.

Interpretar opiniões de uma grande massa populacional ainda é uma tarefa complexa. Primeiro por se tratar de opiniões que devem ser consultadas a partir de um conjunto de características que nem sempre estarão presentes na *web*, incluindo o contexto, as entidades envolvidas, entre outras. Além disso, as informações estão espalhadas em diversas mídias sociais e devem ser armazenadas num único lugar para serem classificadas.

Com o intuito de resolver tal problema, pretende-se desenvolver o protótipo[1] de uma ferramenta de mineração de opiniões que possibilite classificar em favorável ou desfavorável uma opinião ou qualquer outro texto sobre um determinado assunto. O trabalho em questão possui dois principais objetivos, sendo o primeiro a utilização do conceito de *Text Mining* para interpretar informações contidas nas postagens dos usuários que possam ser avaliadas e o segundo referente à utilização de técnicas de classificação para descobrir se são opiniões favoráveis ou não. A principal funcionalidade do protótipo prevista é a realização da classificação automática[2] de opiniões. As opiniões utilizadas serão obtidas em 3 mídias sociais: *Facebook*, *Twitter*, e *Google Plus*.

---

[1] **Protótipo**: Para qualquer ocorrência do termo "protótipo" no decorrer desta monografia, estará sendo referenciado o protótipo de classificação automática de opiniões que foi desenvolvido no decorrer deste trabalho.

[2] **Classificação automática**: tarefa relacionada ao reconhecimento automático de padrões onde são aplicadas técnicas de mineração de dados a fim de classificar um conjunto de dados.

Nas próximas seções serão apresentados o referencial teórico (seção que aborda o processo de descoberta de conhecimento, mineração de dados e mineração de textos); em seguida (seção 2) são apresentados os materiais e métodos utilizados, tornando possível o desenvolvimento do protótipo de classificação automática de opiniões. A seção 3 apresenta os resultados obtidos e, por fim, na seção 4 são expostas as considerações finais deste trabalho.

1 REFERENCIAL TEÓRICO

Este capítulo aborda os conceitos e tecnologias que serão utilizadas para a realização deste trabalho. Dentre os temas abordados será dado um enfoque maior sobre as técnicas de *Data Mining* e *Text Mining*. Elas são essenciais para a compreensão e desenvolvimento do protótipo proposto.

1.1. Processo de Descoberta de Conhecimento (*KDD*)

O processo de descoberta de conhecimento tem a função de explorar e extrair informações importantes, armazenadas nos bancos de dados, propiciando uma melhor gestão nas empresas. O processo *KDD* é muito utilizado nos casos em que existam muitos dados armazenados e é humanamente impossível a detecção de padrões sem o auxílio de tecnologias de extração do conhecimento.

De acordo com Fayyad (1996, p. 40-42) o processo *KDD* corresponde a um segmento de várias etapas, interativo e iterativo, para identificação de padrões compreensíveis, válidos, novos e potencialmente úteis, a partir de grandes conjuntos de dados. A Figura 1, a seguir, apresenta as etapas que compõem o processo *KDD*.

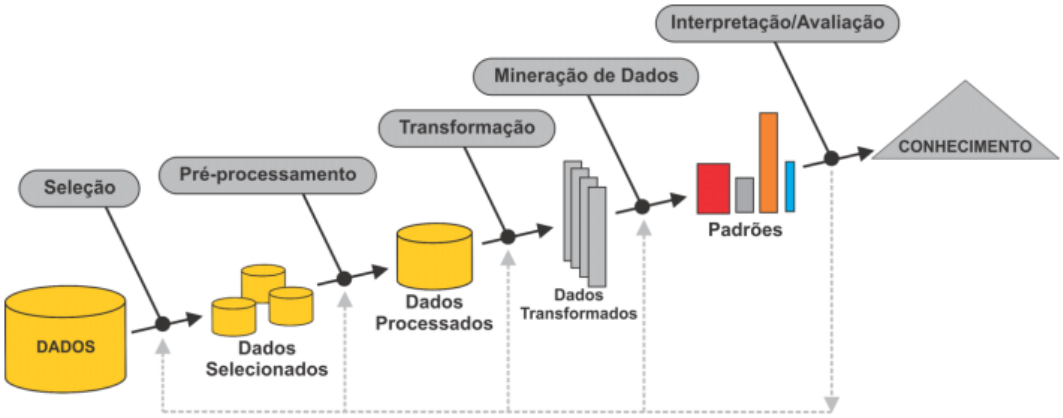

Figura 1 - Visão geral das etapas do processo de *KDD* (FAYYAD *apud* SILVA, 2012, p. 22).

O processo *KDD* tem como objetivo a obtenção do conhecimento. Como pode ser visualizado na Figura 1, são necessários cinco passos para concluir o processo**.** A seguir são explicadas cada uma destas etapas:

- **Seleção** – conforme Apolinário e Silva (2007, p. 24), a etapa de seleção no processo *KDD* resume-se em "(...) identificar e filtrar nos dados os termos mais representativos para as tarefas de mineração". Nesta etapa é utilizada a base de dados que contém o conjunto de registros armazenados de forma estruturada que serão utilizados para a descoberta de conhecimento. Geralmente, as bases de dados das empresas armazenam conteúdo sobre todos os setores, produtos e clientes. Porém, nem todos os dados são relevantes para a mineração de dados. Assim, a seleção de dados é a etapa onde são selecionados os dados relevantes para a mineração de dados;
- **Pré-processamento** – o processamento é a etapa onde se realiza a limpeza dos dados, retirando informações desnecessárias. Conforme Apolinário e Silva (2007, p. 24) este passo consiste na "remoção de dados incompletos, eliminação de tuplas repetidas, integração de dados heterogêneos, eliminação de termos considerados irrelevantes, eliminação de sufixos e prefixos (...)". O processo utiliza as técnicas de *Stemming* e de remoção de *Stopwords* (que serão apresentadas na seção sobre técnicas de *Text Mining*) deste trabalho. Caso esta etapa não seja realizada, o processo *KDD* pode ser comprometido pelo excesso de informações desorganizadas;
- **Transformação** – existem muitas informações nas bases de dados que não podem ser interpretadas diretamente pela máquina e, por isso, é necessário que tais dados sejam enriquecidos com informações relevantes que podem ser identificadas com a intervenção humana. De acordo com Barion e Lago (2008), o propósito principal desta fase é "(...) facilitar a utilização das técnicas de mineração de dados". Por exemplo, avaliando o código de área de um telefone é possível deduzir a localidade do mesmo e isto amplia os resultados da mineração de dados;
- **Mineração de Dados** – a mineração de dados consiste na principal etapa do processo *KDD* que possibilita "(...) uma capacidade sofisticada de pesquisa de dados que usa algoritmos estatísticos para descobrir padrões e correlações de dados" (RYGIELSKI, *et al*, 2002, p. 48, tradução nossa). Nesta etapa é realizada a descoberta de padrões, processo pelo qual são analisados os dados disponíveis para encontrar por meio de associações, categorizações entre outras técnicas, os padrões que possam ser utilizados para auxiliar nas tomadas de decisão da alta gestão;
- **Interpretação/Avaliação** – nesta etapa chega-se ao resultado da descoberta de conhecimento através de visualização e representação do conhecimento alcançado durante o processo (SILVA, 2012, p. 23). Nesta fase são interpretados os padrões

detectados e é avaliada a colaboração dos mesmos como auxílio para os gestores. Utiliza-se nesta etapa a técnica de sumarização (meio pelo qual se apresenta um conjunto de informações) para expor os resultados. Após esta etapa obtêm-se o conhecimento proporcionado pelo processo *KDD*.

Além das etapas supracitadas, pode ser adotada a criação de um *Data Warehouse* (*DW*) para uma melhor organização dos dados. Segundo Inmon (1997, p.33), "um *Data Warehouse* é uma coleção de dados orientada por assuntos; integrada; variante no tempo e não volátil, que objetiva dar suporte aos processos de tomada de decisão". A construção do *DW* conta com o auxílio do processo ETL para a alimentação dos dados e, este é composto pelas etapas de extração, transformação (análoga à limpeza de dados do processo *KDD*) e de carga de dados.

Dentre as etapas do processo *KDD*, será dado um enfoque maior à etapa de *Data Mining*, pois esta é importante para o desenvolvimento deste trabalho.

1.2. *Data Mining*

A mineração de dados (*Data Mining*) é uma das etapas mais importantes do processo *KDD*, possibilitando que os especialistas concentrem esforços na análise dos padrões descobertos na etapa de mineração para identificar se os padrões são úteis ou não (CAMILO; SILVA, 2009, p. 8). Isto porque é nesta etapa que os dados são minerados, ou seja, é onde ocorre o "(...) processo de análise de conjuntos de dados que têm por objetivo a descoberta de padrões interessantes e que possam representar informações úteis" (BRUSSO, 2002, p. 16).

1.2.1. Tarefas

A mineração de dados é realizada por meio de tarefas divididas em dois padrões primários: **preditivas** e **descritivas**. As tarefas preditivas são denominadas como tarefas "supervisionadas". Dentre as tarefas preditivas podem-se destacar as seguintes: classificação, regressão, detecção de desvios, agrupamento, associação e sumarização. Elas são utilizadas na identificação de padrões e tendências com base no conhecimento descoberto numa coleção de dados.

No caso das tarefas descritivas é considerada a necessidade do entendimento dos dados, visto que os padrões ainda são desconhecidos. Dentre as tarefas descritivas podem-se citar as seguintes: árvores de decisão, regras de associação, algoritmos genéticos, redes neurais artificiais. Tais tarefas detalham concisamente os dados disponíveis, buscando por

padrões desconhecidos e, por este motivo, são caracterizadas como "não supervisionadas" dispensando a adição de conhecimento humano sobre um contexto estabelecido (GOELZER, 2007, p. 22, 30).

1.2.2. Técnicas de *Data Mining*

A mineração de dados conta com um conjunto de técnicas de *Data Mining* que podem ser aplicadas tanto em banco de dados quanto em *Data Warehouses*. Cada técnica dessas possui uma abordagem particular em relação ao modo de resolução do problema. Ainda assim, as técnicas possuem características em comum, por exemplo, a intenção de prover a descoberta de conhecimento nas bases de dados. A seguir são apresentadas as principais técnicas de *Data Mining*:

- **Árvores de Decisão:** "A árvore de decisão consiste de uma hierarquia de nós internos e externos que são conectados por ramos" (SOBRAL, 2003, p. 21). É um meio pelo qual o conhecimento é organizado em nós que são acessados pela verificação de condicionais acima de cada nó. Para Santos *et al* (2007, p. 3)

> "Árvore de decisão é uma maneira de representação do conhecimento descoberto, através de um conjunto de nós de decisão (nós internos), ramos e nós-folhas que configuram um modelo de classificação, onde seus nós de decisão representam: os nomes dos atributos; as folhas: as diferentes classes e os ramos dos critérios de decisão".

- **Redes Neurais:** trata-se de uma técnica de previsão para mineração de dados que lida com processamento de dados de forma análoga ao cérebro humano. Segundo Hastie; *et al* (2008, p. 411, tradução nossa) "uma rede neural é uma regressão de duas fases ou modelo de classificação, tipicamente representado por um diagrama de rede". Semelhante ao cérebro que é composto por milhões de neurônios conectados entre si por meio de sinapses, as redes neurais utilizam inúmeros simuladores de neurônios conectados similarmente ao cérebro humano (BRAGA, 2012, p. 27). Assim como os cérebros humanos, as redes neurais são capazes de aprender, armazenar conhecimentos e adaptar a novas situações (NAGOA, 2003, p. 25). Além disso, "... podem ser utilizadas para resolver problemas de estimação (com resultados contínuos)" (RYGIELSKI, *et al*, *op. cit*., p. 499, tradução nossa).
- **Indução de Regras:** esta técnica consiste no "uso de ferramental matemático e estatístico, que visam o desenvolvimento de relacionamentos a partir dos dados apresentados. Tipicamente, são criadas correspondências do tipo 'se-então', baseadas

em relações causais detectadas nas variáveis. Cada relacionamento 'se-então' extraído é chamado de 'regra'" (LAZZAROTTO; *et al,* 2006, p. 111).

- **Análise de Agrupamento:** "a Análise de Agrupamento (*Cluster*), também chamada de segmentação de dados, tem uma variedade de objetivos. Todos relacionados com o agrupamento ou segmentação de uma coleção de objetos em subconjuntos ou grupos" (HASTIE; *et al*, 2008, p. 501).

1.2.3. Algoritmos de *Data Mining*

Os algoritmos de mineração de dados ou algoritmos de *Data Mining* são essenciais para a análise dos dados de um banco de dados. Podem ser definidos como "… um conjunto de heurística e cálculos que cria um modelo de mineração de dados" (MICROSOFT, 2012, online). Segundo Rygielski *et al* (2002, p. 487, tradução nossa) "com a aplicação de algoritmos avançados, a mineração de dados revela conhecimento em uma grande quantidade de dados e aponta possíveis relações entre os dados". Os algoritmos de mineração de dados atendem as necessidades das técnicas de mineração. A seguir são apresentados alguns algoritmos de mineração:

- **Algoritmo Apriori**: é um algoritmo que pode ser utilizado na mineração de dados para o estudo de regras de associação.
- **Algoritmo CURE:** O Algoritmo *Clustering Using Representatives* (CURE) trata-se de um algoritmo hierárquico aglomerativo que trabalha de forma mista para calcular a distância entre dois *clusters* (GUHA, 1998 *apud* GONZÁLEZ, 2010)**.**
- **Algoritmo GSP (*Generalized Sequential Patterns*):** é um algoritmo que lida com padrões sequenciais e/ou sequenciais, sendo que, uma sequência ou padrão é representado pela seguinte expressão $< I_1;... ; In >$, onde cada I_i é um aglomerado de itens.
- **Algoritmos Genéticos:** "(...) são abordagens de busca probabilística baseados nos processos evolucionários" (FALCONE, 2004, p. 39) e "que fornecem um mecanismo de busca paralela e adaptativa baseado no princípio de sobrevivência dos mais aptos e na reprodução" (PACHECO, 2009, p. 1).
- ***K-Means*:** o algoritmo *K-means* é considerado como um algoritmo não supervisionado por não necessitar da intervenção humana e tem por objetivo o fornecimento de classificações conforme os próprios dados. Segundo Hastie *et al* (2008, p. 509, tradução nossa) este algoritmo agrupa os dados iterativamente e "(...) destina-se a situações em que todas as variáveis são quantitativas e possuem a distância euclidiana ao quadrado" conforme a equação a seguir:

$$d(x_i, x_i') = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = || x_i - x_{i'} ||^2$$

- **Máquinas de Vetores de Suporte:** as Máquinas de Vetores de Suporte (SVMs) correspondem a um tipo de algoritmo criado para as redes neurais, fundamentado pelo princípio do aprendizado estatístico. Estabelece um conjunto de princípios que devem ser adotados na obtenção de classificadores com boa generalização, para ser estabelecido pelas suas capacidades de prever corretamente classes de novos dados do mesmo domínio em que ocorreu o aprendizado (LORENA; CARVALHO, 2007, p. 1).

1.3. *Processo de Descoberta de Conhecimento em Textos (KDT)*

O processo *KDT* é similar ao processo *KDD*, inclusive nas etapas de mineração e interpretação/avaliação (SILVA, 2012, p. 24). No entanto, o *KDT* baseia-se em "técnicas específicas para tratamento de textos que devem ser utilizadas a fim de se obter conhecimentos implícitos em banco de dados textuais" (BARION; LAGO, 2008, p. 126).

No processo *KDT* os dados são cruzados a procura de padrões a fim de obter o conhecimento. Diferente da descoberta de conhecimento em dados estruturados, o processo *KDT* requer uma série de etapas de processamento para a estruturação dos dados. Apenas após os dados serem pré-processados é possível realizar a análise dos mesmos. A Figura 2 apresenta um fluxo com ambos os passos do processo *KDT* e *KDD* em que é realizada uma comparação entre as mesmas.

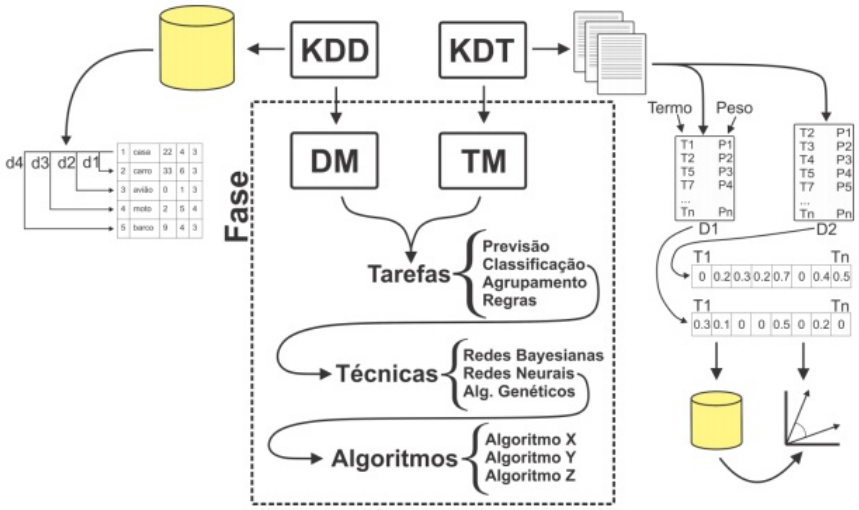

Figura 2 - Detalhamento e diferenciação dos processos de *KDD* e *KDT* (SILVA, 2012, p. 25)

Como podem ser observados na Figura 2 os processos apresentados (KDD e *KDT*) possuem etapas parecidas, que se diferenciam apenas no pré-processamento e na seleção dos dados. Sabe-se que a mineração de dados lida com dados estruturados e, por isso, necessita passar por uma conversão. Camilo e Silva (2009, p. 5) afirmam que, geralmente, "(...) antes de se aplicar os algoritmos de mineração é necessário explorar, conhecer e preparar os dados". Por esse motivo, a mineração de textos possui uma etapa diferenciada.

As fases do processo de descoberta de conhecimento executam tarefas de mineração de dados (previsão, classificação, agrupamento e regras) que são apoiadas por um conjunto de técnicas que são realizadas com a utilização de algoritmos (SILVA, 2012, p. 22). A seguir é apresentado o fluxo do processo *KDT* na Figura 3.

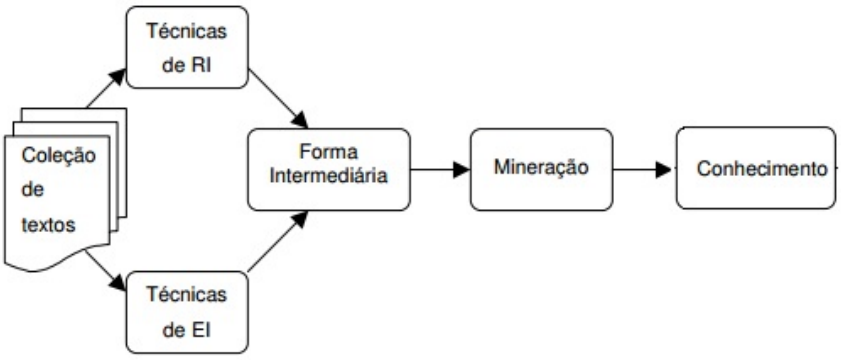

Figura 3 - Processo *KDT* – Adaptado de (CORREA, 2003, p. 6)

Antes de realizar a etapa de *Text Mining* é necessário transformar os dados disponíveis numa forma intermediária que pode ser um conjunto de radicais de palavras. Existem dois modos disponíveis para pesquisar informação nos documentos textuais: por meio das Técnicas de Recuperação da Informação (RI) ou por meio das Técnicas de Extração da Informação (IE), tal como explicado abaixo.

- **Recuperação de Informação:** a recuperação da informação é o campo da informática que lida com a parte de pesquisa de conteúdo a partir de termos informados pelos usuários. Segundo Cardoso (2000, p. 33), a RI consiste numa "(...) subárea da ciência da computação que estuda o armazenamento e recuperação automática de documentos, que são objetos de dados, geralmente textos".
- **Extração da informação:** A extração da informação é um meio de análise de "(...) um texto ou um conjunto de textos com a meta de extrair fatos apresentados nos mesmos" (SCARINCI, 2001, p. 18). Segundo o mesmo autor (2001, p. 16), a extração da

informação pode extrair informação de documentos de acordo com critérios predefinidos. É uma atividade relacionada à busca e relacionamento de informações retiradas de documentos textuais, que tem como objetivo a extração de itens predeterminados, ignorando dados que não pertencem ao domínio.

Após os dados estarem numa forma intermediária é realizada a etapa de mineração na qual serão utilizadas as técnicas de *Text Mining*. A próxima seção apresenta as principais técnicas de *Text Mining* que constituem a etapa mais importante do processo *KDT*.

1.4. *Text Mining*

A mineração de texto, diferente da mineração de dados, utiliza conteúdo não estruturado ou parcialmente estruturado e contempla um conjunto de técnicas usadas "(...) para navegar, organizar, achar e descobrir informação em bases textuais" (ARANHA; PASSOS, 2006, p. 2). A utilização de *Text Mining* revela informações importantes contidas em documentos textuais. A Figura 4 demonstra o conjunto dessas áreas.

Figura 4 - Multidisciplinaridade da mineração de textos (SOARES, 2008, p. 35)

Como pode ser visto na Figura 4, *Text Mining* caracteriza-se como uma área interdisciplinar, que abrange "(...) recuperação de informação, compreensão de texto, extração de informações, *clustering*, categorização, visualização, a tecnologia de banco de dados, aprendizado de máquina e mineração de dados" (TAN, 1999, p. 1, tradução nossa). Nas próximas seções são apresentadas as principais técnicas de mineração de textos.

1.4.1. Tokenização

Quando se trata de mineração de dados, a tokenização corresponde à tarefa de subdividir o texto por meio de delimitadores literais como: espaço, quebra de linha, ponto, vírgula, tabulações, entre outros. A tarefa de tokenização tem como resultado um vetor que contém as palavras do texto.

Como o texto puro não pode ser analisado pela máquina, é necessário subdividi-lo para analisá-lo por partes. Com a tokenização é possível, por exemplo, realizar a contagem dos termos, obtendo um melhor resultado da análise. Na Figura 5 é apresentada uma metodologia de identificação de *tokens* elaborada por Konchady (2006 *apud* SOARES, 2008, p. 45).

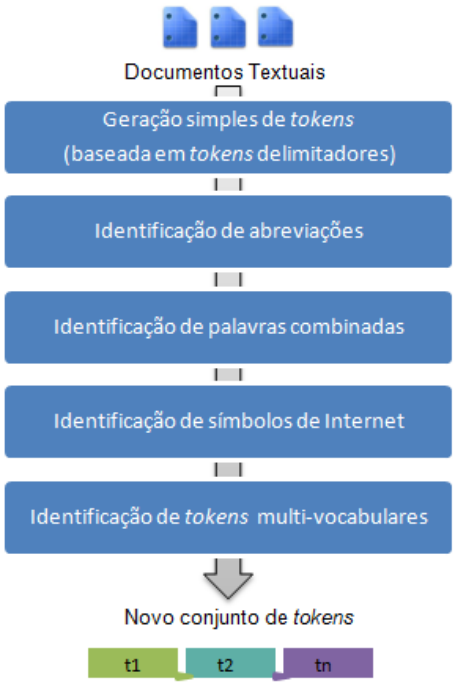

Figura 5 – Metodologia de identificação de *tokens* – Adaptado de (KONCHADY *apud* SOARES, 2008, p. 45)

Conforme pode ser observado na Figura 5, não basta apenas realizar uma divisão genérica do texto pelos delimitadores. É necessário realizar alguns passos para identificar e tratar as exceções como: palavras compostas, símbolos da internet, abreviações, números e outros. A seguir é apresentado o resultado da técnica de tokenização aplicada na seguinte frase:

"*O Bacharel em Sistemas de Informação é capacitado para desenvolver trabalhos nas áreas de Informática e Computação*"

Resultado:

[“*O*”, “*Bacharel*”, “*em*”, “*Sistemas*”, “*de*”, “*Informação*”, “*é*”, “*capacitado*”, “*para*”, “*desenvolver*”, “*trabalhos*”, “*nas*”, “*áreas*”, “*de*”, “*Informática*”, “*e*”, “*Computação*”]

Conforme apresentado acima, a tokenização da frase de exemplo resultou na quantidade de 17 *tokens*. No resultado, as palavras sublinhadas são consideradas palavras de baixo valor significativo e não tem importância para a mineração, tal situação será explicada na seção sobre *Stopwords*. Embora no exemplo não tenha sido utilizado nenhum delimitador diferente do “espaço”, podem-se destacar os seguintes delimitadores: () <>!-?.;’- “|.

1.4.2. *Stemming*

O conteúdo de qualquer documento textual é composto por uma gama de palavras que têm significado semelhante a outra. Algumas destas se diferenciam apenas pelas suas variações textuais (derivação de afixos e outros) o que não constitui outro valor semântico para a análise do texto ao ser comparado com outras palavras derivadas do mesmo radical. Nestes casos, utiliza-se o conceito de *Stemming* que é o processo de reduzir as palavras ao seu radical, retirando afixos, desinências (indicadores de gênero e número), e vogais temáticas. Com isto, todas as flexões de palavras poderão ser contabilizadas como se fossem uma só. Na Tabela 1 são apresentados alguns exemplos de radicais de palavras.

Tabela 1: Exemplos de radicais de palavras

Palavra	Radical
Carro	Car
Carrão	Car
Sonho	Sonh
Sonhador	Sonh
Terra	Terr
Terreiro	Terr

Para obter os radicais das palavras é necessária a utilização de algoritmos destinados à função de remover os afixos das palavras levando em conta o idioma das mesmas. A seguir é apresentado o algoritmo de Orengo, que é um dos mais conhecidos da língua portuguesa (SILVA, 2004 apud VIEIRA, 2007, online).

- **Algoritmo de Orengo**

Poucos são os algoritmos disponíveis para a radicalização de palavras na língua portuguesa. Entre os algoritmos disponíveis encontram-se o algoritmo de Orengo, que é constituído por 199 regras de remoção de sufixos, que são distribuídas em 8 passos e podem atender a exceções (PESSOTTI, 2008, p. 14). A seguir, na

Tabela 2, são apresentadas as etapas do algoritmo.

Tabela 2: Etapas do Algoritmo de Orengo (VIEIRA, 2007, online)

Etapa	Descrição
1. Redução do plural	Remove-se o final -s indicativo de plural de palavras que não se constituem em exceções à regra, realizando modificações, quando necessário.
2. Redução do feminino	Remove-se o final -a de palavras femininas com base nos sufixos mais comuns.
3. Redução adverbial	Remove-se o final -mente de palavras que não se constituem em exceção.
4. Redução do aumentativo/diminutivo	Removem-se os indicadores de aumentativo e diminutivo, mais comuns.
5. Redução nominal	Removem-se 61 sufixos possíveis para substantivos e adjetivos.
6. Redução verbal	Reduzem-se as formas verbais aos seus radicais.
7. Remoção de vogais	Removem-se as vogais “a”, “e” e “o” das palavras que não foram tratadas pelos dois passos anteriores.
8. Remoção de acentos	Removem-se os sinais diacríticos das palavras.

Como pode ser visto na Tabela 2, são executadas primeiramente as etapas mais genéricas, isto é, verificam-se anteriormente as regras que envolvem a extremidade da palavra. Como a remoção dos assentos é uma etapa que não coloca em risco a perda de dados essenciais para representação do termo, ela é realizada por último. Na Figura 6 é apresentado o fluxo do algoritmo de Orengo:

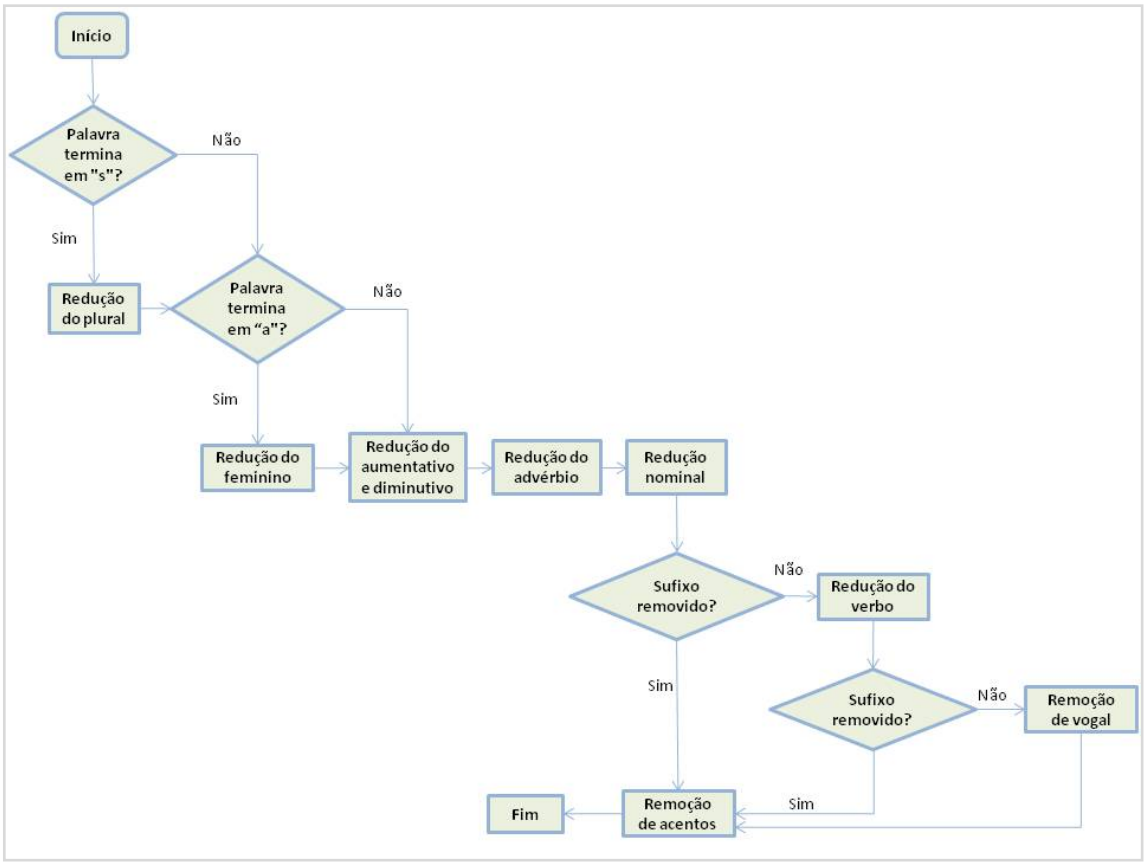

Figura 6 - Fluxo do algoritmo de Orengo (ORENGO *apud* VIERA, 2007, online)

O fluxo do algoritmo é representado por asserções usuais da lógica de predicados. É realizada uma função de radicalização para cada conectivo do fluxo. Os condicionais são utilizados para revelar a próxima função a ser executada até que o fluxo seja finalizado.

1.4.3. *Stopwords*

No decorrer de um documento textual existem várias palavras que não possuem valor significativo algum. São palavras que aparecem com frequência e podem encontrar-se nas seguintes classes gramaticais: artigos, preposições, conjunções, pronomes e advérbios. A seguir são demonstrados alguns exemplos de *stopwords*.

Tabela 3: Exemplos de *Stopwords* (BETTIO, 2007, p. 35)

A	com	dos	já	também	sem	estão
O	não	como	está	só	mesmo	você
Que	uma	mas	seu	pelo	aos	tinha
E	os	foi	sua	pela	ter	foram
Do	no	ao	ou	até	seus	essa
Da	se	ele	ser	isso	quem	num
Em	na	das	quando	ela	nas	nem
Um	por	tem	muito	entre	me	suas
Para	mais	à	há	era	esse	meu
Qual	essas	tu	minhas	nossa	estes	isto
Será	esses	te	teu	nossos	estas	aquilo
Nós	pelas	vocês	tua	nossas	aquele	havia
tenho	este	vos	teus	dela	aquela	seja
Lhe	fosse	lhes	tuas	delas	aqueles	pelos
deles	dele	meus	nosso	esta	aquelas	elas
numa	têm	minha	às	a	de	cujo
A	com	dos	já	também	sem	estão
O	não	como	está	só	mesmo	você

A remoção das *stopwords* é uma tarefa muito importante na mineração de textos. É utilizada para que não haja informações irrelevantes no processamento das tarefas de mineração. Como pode ser verificado na **Tabela 3**, a partir destas palavras, não é possível abstrair conceitos do texto. Logo estas palavras não acrescentam benefícios na análise dos dados. A seguir é demonstrado um exemplo de remoção de *Stopwords*:

"O profissional de Sistemas de Informação é preparado para investigar conceitos e técnicas de informática, contribuindo na solução de problemas de tratamento de informações nas organizações"

Resultado:

"profissional – Sistemas - Informação - preparado - investigar - conceitos - técnicas - informática - contribuindo - solução – problemas – tratamento - informações - organizações"

1.4.4. Sumarização

A sumarização é o processo de restringir os termos existentes na coleção de documentos textuais de forma a manter apenas os termos que estiverem mais relacionados com o conteúdo. Mesmo fora da computação, a sumarização é um conceito utilizado todos os dias pelas pessoas.

Quando é realizado o procedimento de sumarização o texto passa a ser representado por um conjunto de conceitos que indexa o conteúdo do documento para o processo de recuperação da informação, ou seja, o texto será referenciado por sumários. De acordo com Rino e Pardo (2003, p. 207) os "sumários são textos produzidos a partir de textos ou de suas correspondentes representações, podendo servir de indexadores ou substitutos dos mesmos".

Para Rino e Pardo (2003, p. 203), a sumarização automática tem o intuito de

> "simular as principais características da tarefa humana: identificar segmentos relevantes de um texto e compô-los, para produzir os sumários correspondentes. Sumários, neste contexto, são sempre textos resumidos. Existem diversos modelos de sumarização automática, que envolvem o conhecimento linguístico ou a manipulação estatística ou empírica das informações textuais".

A técnica de sumarização opera removendo do texto as partes consideradas como supérfluas e/ou diminuindo abordagens muito detalhadas, evidenciando desta forma, os trechos mais relevantes que se tornarão os sumários.

1.4.5. Associação

Associação é, para a área de descoberta de conhecimento, um meio pelo qual um determinado documento se relaciona com outro. Dentre as tarefas de mineração de dados, a associação é um conceito muito importante para a descoberta de conhecimento.

A análise das regras de associação "(...) surgiu como uma ferramenta popular para a mineração em bases de dados comerciais. O objetivo é encontrar os valores conjuntos das variáveis $X = (X_1, X_2, ..., X_p)$ que aparecem com mais frequência na base de dados" (HASTIE; *et al*, 2008, p. 487, tradução nossa). Assim, a associação entre documentos é identificada sobretudo pela ocorrência de termos em comum nos documentos. Se não fosse tal regra muitos padrões existentes nos documentos não seriam identificados.

1.4.6. Classificação

No que diz respeito ao processo *KDD*, a classificação é uma tarefa de mineração de dados que tem como objetivo associar ou classificar elementos às suas classes correspondentes, determinando automaticamente uma classe para cada novo objeto.

Para que seja possível executar a tarefa de classificação deve existir uma base treinada contendo as classes possíveis. Para isso, é necessário que um especialista experiente da área forneça a base inicial contendo registros classificados. Em alguns casos os registros iniciais podem não contemplar todo o contingente de elementos. Por isto, esta tarefa pode

possuir uma margem de erro que deve diminuir de acordo com a abrangência da base treinada.

Segundo Camilo e Silva (2009, p. 9) a etapa de classificação visa "(...) identificar a qual classe um determinado registro pertence. Nesta tarefa, o modelo analisa o conjunto de registros fornecidos, com cada registro já contendo a indicação à qual classe pertence, a fim de 'aprender' como classificar um novo registro".

1.4.7. Clusterização

Clusterização ou agrupamento é um meio de agrupar documentos que tenham características em comum. Sempre que existir uma similaridade entre dois documentos eles estarão relacionados de algum modo e por isso podem pertencer ao mesmo grupo. A clusterização é realizada por meio de algoritmos que "(...) são ferramentas úteis para a mineração de dados, compressão, aprendizagem não supervisionada, estimação de probabilidades, e outras tarefas no aprendizado de máquina e estatística" (HAMERLY; ELKAN, 2012, p. 281, tradução nossa). A seguir são apresentados os principais conceitos da clusterização:

- ***Cluster*:** trata-se de um grupo de objetos, entre outros grupos, cujos dados devem ser os mais similares possíveis dentro de cada grupo.
- **Centroide:** trata-se de um modo útil para sintetizar grupos e validar estes grupos em um novo conjunto de dados.
- **Similaridade:** "O método de similaridade calcula e define se dois objetos são semelhantes, e sendo semelhantes podem ser agrupados juntos, sendo que um dos elementos é um centroide (...)" (BARREIRA; SOUZA, 2011, p. 221).

Na próxima seção são apresentados os matérias e métodos que foram utilizados no decorrer do desenvolvimento deste trabalho.

2 MATERIAIS E MÉTODOS

Este capítulo apresenta os materiais e métodos utilizados durante o desenvolvimento do protótipo de classificação automática de opiniões proposto neste trabalho. Este protótipo é composto de uma biblioteca que disponibiliza recursos de mineração de textos e também de um sistema web que utiliza esta biblioteca para realizar a mineração de opiniões advindas de três mídias sociais: *Facebook*, *Twitter* e *Google Plus*. A biblioteca supracitada utiliza uma parte da *Weka API* e foi desenvolvida no *NetBeans* (*IDE* de desenvolvimento que será apresentado na seção 2.4).

2.1. *Weka API*

A *Weka API* é uma composição de bibliotecas que fornecem recursos de mineração de dados que podem ser integrados a outros sistemas. Para utilizá-la não é preciso comprar uma licença, visto que o projeto utiliza a licença *GNU General Public License* (*GPL*) para garantir a liberdade de compartilhamento e alteração da mesma.

Ela possui algoritmos de classificação, clusterização, associação, além de disponibilizar de recursos para realizar o pré-processamento dos dados. A *API* utiliza uma base de dados do tipo *Attribute-Relation File Format* (*ARFF*). Este formato de documento armazena uma lista de instâncias e de propriedades. As propriedades são especificadas no topo do documento e os registros no decorrer do documento (BOUCKAERT; e*t all*, 2013, p. 171-176, tradução nossa).

Embora a *Weka API* possua um conjunto de algoritmos para mineração de opiniões, optou-se por utilizá-la apenas para manusear a base de palavras classificadas com os respectivos valores semânticos para identificar a polaridade[3] das mesmas. Optou-se por não utilizar os algoritmos de classificação disponíveis na *Weka API* porque o funcionamento dos mesmos é baseado em aprendizado de máquina. Isto significa que é necessário treinar um conjunto de registros e utilizá-los como base para realizar as novas classificações. No entanto, como o protótipo alvo deste trabalho não possui nenhum contexto específico, identificou-se que a classificação encontrava-se instável e possuía uma margem de erro alta.

[3] **Polaridade**: o termo polaridade será utilizado no decorrer desta monografia para expressar sobre o valor semântico de uma palavra que pode ser considerado como positivo ou negativo.

Dentre os modelos de classificadores testados na *Weka API*, foram verificados o algoritmo *SVM* disponibilizado pelo *libSVM* (uma biblioteca Java que deve ser adicionada separadamente) e também o algoritmo *k-nearest neighbor algorithm* (*KNN*) que utiliza como técnica de aprendizagem o método *Lazy Learning*. Tanto o algoritmo *SVM* quanto o *KNN* requerem uma base de registros que devem ser treinados previamente. Esses registros podem estar armazenados numa base *ARFF* ou em algum banco de dados como o *Oracle* e *MySQL*. Para utilizar a base dos registros treinados deve existir um atributo que armazene as opiniões e outro atributo para especificar a classe de cada opinião. Em ambos deve-se utilizar o método ***buildClassifier*** passando como parâmetro a base de treinamento. Para realizar uma classificação num registro específico deve-se utilizar o método ***classifyInstance*** que recebe como parâmetro a instância que deve ser classificada.

Embora os algoritmos providos pela *Weka API* conseguissem carregar a base de treinamento e realizar a classificação para novos registros, devido às inúmeras inconsistências na classificação dos registros, incluindo registros com opiniões curtas e diretas, decidiu-se desenvolver um algoritmo de classificação próprio baseando-se em palavras com valores semânticos pré-definidos. A próxima seção apresenta o *framework* utilizado no desenvolvimento do protótipo.

2.2. *Grails*

Grails é um *Framework web Open Source e Full-stack* (ambiente configurado com os principais recursos necessários para o desenvolvimento de um sistema *web* que funcione de forma unificada). Ele é executado no *Java Virtual Machine* (*JVM*) e adota o modelo de desenvolvimento *Model-view-controller* (*MVC*). O *MVC* é um modelo de organização da arquitetura de sistemas web que divide o desenvolvimento em três camadas: *model*, *view* e *controller* e tem como principal objetivo a separação da lógica, da persistência de dados e da visualização da aplicação. Mais detalhes do *MVC* são apresentados na Figura 7.

O *Grails* é composto por um *Servlet Container* (servidor utilizado internamente para executar as aplicações desenvolvidas) e pelo *Spring* (*framework* que realiza o mapeamento dos dados e que é responsável pela convenção dos projetos).

O *Framework Grails* utiliza o paradigma *Convention Over Configuration* (*CoC*). O *CoC*, em português, programação por convenção, é um modo de desenvolvimento que tem por objetivo diminuir a necessidade de tomada de decisão por meio do desenvolvedor. Neste paradigma são adotadas todas as configurações que são comumente utilizadas nos sistemas e

caso haja necessidade, o desenvolvedor pode realizar as alterações desejadas (NETO, 2010, online).

O *Grails* utiliza o *Groovy* como sua linguagem de programação padrão, mas também permite a utilização e até mesmo a intercalação com a linguagem *Java*. Isso é possível pelo fato de toda a codificação ser compilada para *Bytecode Java* e por ser interpretada pela *JVM*. Na Figura 7 pode ser observada a arquitetura do *Grails*.

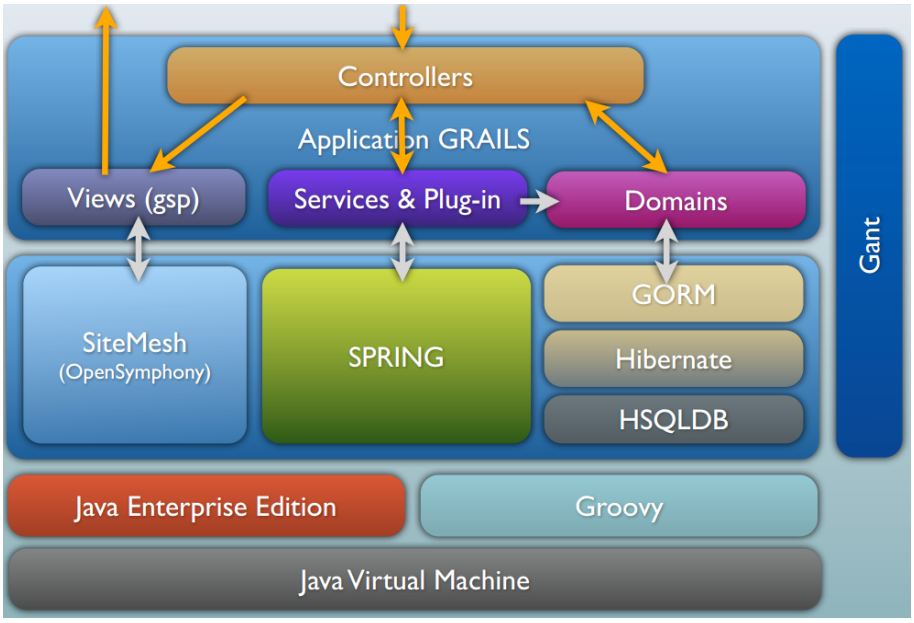

Figura 7 – Arquitetura do *Grails* (TORRIELLI, 2010)

Como pode ser visto na Figura 7 o *Grails Framework* é composto por muitos componentes que juntos proporcionam um ambiente integrado com os principais recursos necessários para o desenvolvimento de um sistema. A seguir será definido cada um destes componentes.

- ***Models (Domains):*** este conceito compreende o conjunto das entidades que existem num determinado sistema. Nele são definidas as propriedades e as operações básicas de um sistema. Cada *Model* representa uma tabela no banco de dados. Além disso, nos *domains* pode ser especificado o mapeamento da classe e das propriedades com as tabelas do banco. Uma das formas para se criar um *Model* é utilizando o seguinte comando: *$ grails create-domain-class Sample*.

  Após a execução desse comando, será criada a classe *Sample* no arquivo *SampleClass*. Outra opção importante nos *Models* são as *constraints*, parte do código onde é possível especificar regras de validação na inserção dos dados.

- ***Views*:** páginas que são criadas para serem apresentadas para o usuário. São conhecidas por *Groovy Server Pages* (GSP). Geralmente, acionam-se estas páginas por meios de *controllers* que processam uma chamada do cliente e retorna algum parâmetro para uma *view* de nome correspondente. Por exemplo, uma *controller* de nome **PessoaController,** por padrão, sempre retornará o resultado para a *view* **pessoa.gsp**, exceto se houver algum redirecionamento para outra *controller* ou *view* no decorrer da execução.
- ***Controllers*:** as *controllers* interceptam as solicitações do usuário. A partir delas é possível acessar as *domains*, os *services* e as funcionalidades disponibilizadas pelos *plugins* instalados para a aplicação. Segundo Klein (2009, p. 41, tradução nossa) "As classes do controlador são os despachantes de uma aplicação *Grails*. Todos os pedidos do navegador vêm através de um controlador". Após cada processamento de um *controller* é renderizada uma *view* do sistema.
- ***Services & Plug-in*:** geralmente, são serviços que realizam atividades que resolvem um determinado problema ou realizam uma determinada operação. Não é aconselhável manter na controller uma lógica que não seja específica de uma única entidade (classe) do sistema. Nestes casos devem-se utilizar os *Services*. Isso evita que o *controller* fique muito complexo e facilita o reuso da lógica (KLEIN, 2009, p. 108, tradução nossa). No caso dos *Plug-ins*, não é necessário codificar nenhuma funcionalidade. Eles são disponibilizados no site do *Grails* e, portanto, é necessário apenas instalá-los e utilizá-los.
- ***SiteMesh*:** muitos sistemas utilizam alguma forma de organização de *templates* o que é muito importante para que algumas partes do *layout* possam ser reutilizadas sem que seja necessário a duplicação de código de interface. No *Grails* essa técnica é conhecida por *SiteMesh*. Conforme Judd (2012, p. 92), o *SiteMesh* é um *Framework* de *layout* que implementa o padrão *design decorator* (padrão que pode ser utilizado para estender funcionalidades de um objeto) para renderização do HTML. A utilização do *SiteMesh* possibilita não apenas a incorporação de conteúdo numa determinada página, mas sim uma fusão de dados. Na Figura 8 pode ser observada a estrutura do *layout* do *Grails*.

```
1  <html>
2      <head>
3      <title><strong><g:layoutTitle default="Grails" /></strong></title>
4      <link rel="stylesheet" href="${resource(dir:'css',file:'main.css')}" />
5      <link rel="shortcut icon" href="${resource(dir:'images',file:'favicon.ico')}"
6          type="image/x-icon" />
7          <g:layoutHead />
8          <g:javascript library="application" />
9      </head>
10     <body>
11         <div id="spinner" style="display:none;">
12             <img src="${resource(dir:'images',file:'spinner.gif')}" alt="Spinner" />
13         </div>
14         <div id="grailsLogo">
15             <g:layoutBody />
16     </body>
17 </html>
```

Figura 8 - Código do Layout Padrão do *Grails*

No código apresentado tem-se as seguintes *tags* **g:layoutTitle, g:layoutHead, g:layoutBody** que são as utilizadas para indicar a inclusão de um *template*. Pode-se observar, por exemplo, a linha 3 do código onde foi definida a palavra "*Grails*" como título padrão. O documento que utilizar este *layout* deverá conter uma seção para o título e caso padrão este não seja definido, será utilizada a definição do *layout*.

- ***Spring*:** o *Spring Framework* fornece suporte às configurações de infraestrutura que podem ser feitas em nível de aplicação, ou seja, os desenvolvedores não precisarão se preocupar com configurações sobre as tecnologias específicas do ambiente de implantação. Ele inclui injeção de dependência, suporte à programação orientada a aspectos, transações, cache, validação, formatação, suporte ao *Hibernate* e ao *Quartz* (serviço de agendamento que pode ser utilizado por qualquer aplicação Java) (SPRING, 2013, online).
- ***GORM*:** recurso para mapeamento objeto relacional de aplicações *Grails* que utiliza o Hibernate 3. A Figura 9 contém um exemplo de código utilizando o *GORM*.

```
1 def book = Book.findByTitle("Groovy in Action")
2 book
3   .addToAuthors(name:"Dierk Koenig")
4   .addToAuthors(name:"Guillaume LaForge")
5   .save()
```

Figura 9 – Exemplo de utilização do GORM

Como pode ser visto no exemplo, na linha 1 é consultada uma instancia da classe ***Book*** cujo título seja "*Groovy in Action*" e a instancia é atribuída ao objeto ***book***. Nas linhas 3 e 4 são criadas duas instâncias da classe ***Author*** e as mesmas são

vinculadas ao objeto ***book***. A instrução da linha 5 faz com que os objetos sejam armazenados no banco.

- ***Hibernate*:** *Framework* que propicia o mapeamento entre os campos de uma base de dados ao modelo-objeto de aplicações Java. Ao utilizá-lo pode-se optar por utilizar *SQL* ou *Hibernate Query Language* (HQL) que é uma linguagem cujas operações são realizadas com base nos objetos da aplicação.
- ***HSQLDB*:** é o banco de dados padrão do *Grails*. Ao iniciar uma aplicação, se nenhuma configuração for realizada no arquivo ***Datasource.Groovy*** que fica na pasta **Grails-app\conf** dentro da aplicação, os dados serão armazenados no *HSQLDB*. Porém, na próxima vez que o sistema for iniciado os dados não estarão mais disponíveis. No protótipo não foi utilizado um banco de dados para armazenar as opiniões. Elas são carregadas a partir de consultas nas *APIs* do *Facebook*, *Twitter* e *Google Plus*.
- ***Gant***: ferramenta utilizada no acionamento de tarefas de script *Ant* (biblioteca *Java* utilizada na construção de aplicações Java e no gerenciamento de dependências). Diferente do *Ant*, o *Gant* utiliza a linguagem *Groovy* em vez de *XML* para especificar o que deve ser feito na aplicação.
- ***Java Enterprise Edition***: plataforma de programação que possibilita o funcionamento de aplicações desenvolvidas em linguagens cujo compilador de *Bytecodes* tenha suporte à plataforma Java. Dentre estas linguagens pode-se mencionar o *Java*, o *Scala*, o *Clojure* e o *Groovy*.
- ***Groovy*:** linguagem de Programação que é utilizada no desenvolvimento de sistemas baseados no *Grails*. Apesar do *Grails* também aceitar a linguagem de programação *Java*, o *Groovy* possui muitas vantagens em relação ao *Java*, entre elas pode-se citar os seguintes: *Spread Operator, Elvis Operator, Safe Navigation, Getters e Setters* implícitos. Abaixo é exposta a definição destes conceitos.
    - ***Spread Operator*:** recurso utilizado para acionar operações numa única instrução para todo um conjunto de elementos pertencentes a um determinado objeto. Este recurso é, geralmente, utilizado em *collections* sendo necessário utilizar apenas um asterisco antes da chamada do método (colecaoDeObjetos*.metodo()). O retorno da operação é um *array* contendo o retorno de cada chamada ao método (JUDD, 2012, p. 52).

- ***Elvis Operator*:** operador ternário simplificado que utiliza a instrução "**?:**" para verificar se um objeto é falso ou nulo e, em caso afirmativo, retorna um valor determinado, caso contrário, retorna o próprio objeto verificado, conforme o seguinte exemplo:
  def displayName = user.name ?: "Anonymous" (JUDD, 2012, p. 52).
- ***Safe Navigation***: verifica se um objeto é *null* antes de acessar suas propriedades ou seus métodos, evitando-se assim, a ocorrência de um *NullPointerException*. Utiliza-se este conceito conforme o exemplo a seguir: objeto?.propriedade. Dessa forma, o sistema apenas acessa a propriedade se o objeto não for igual a *null* (JUDD, 2012, p. 54).
- ***Getters e Setters*:** no *Groovy* não é necessário definir os *Getters* e *Setters*. A partir do momento em que as propriedades são definidas nos *models*, estes métodos são criados implicitamente.

- ***Java Virtual Machine*:** máquina virtual que executa aplicações *Java*. Ela converte os *Bytecodes* em código de máquina permitindo que uma mesma aplicação seja executada em diversos tipos de hardware desde que exista uma versão apropriada da *JVM* para o hardware respectivo.

2.2.1. *Jetty*

O *Jetty* é o *Servlet Container* padrão do *Grails.* Ele é um projeto *open-source* livre para distribuição e uso comercial que foi desenvolvido em *Java* que provê um servidor HTTP, um cliente HTTP e um *Servlet Container*. Ele é muito utilizado pelos desenvolvedores por ser de fácil configuração, eliminando desperdício de tempo nos testes (ECLIPSE, 2013, online).

O *Jetty* é concorrente do *Apache Tomcat* (*Servlet Container*) e ficou bastante conhecido por ter sido utilizado no *JBoss*. O *Jetty* é disponibilizado pela fundação Eclipse e as suas versões anteriores são mantidas pelo *Codehaus* (ambiente colaborativo para a construção de projetos de código aberto). O *Jetty* foi utilizado em todos os testes realizados.

2.3. *Highcharts JS.*

Highcharts JS é uma biblioteca de gráficos escritos em *HTML5/JavaScript* puro. O produto foi lançado no final de 2009 e desenvolvido por uma empresa norueguesa chamada *Highsoft Solutions AS* e fundada por Torstein Hønsi. Esta biblioteca dá suporte para a criação de gráficos das seguintes formas: "linha, spline área, areaspline, coluna, barra, pizza, dispersão, medidores angulares, arearange, areasplinerange, columnrange, bolha, gráfico de

caixa, barras de erro, funil, cachoeira e tipos de gráficos polares" (HIGHCHARTS, 2003, online, tradução nossa).

A biblioteca *Highcharts JS* foi utilizada na *view* principal do protótipo de classificação de opiniões para apresentar em forma de gráfico as opiniões agrupadas em positivas, muito positivas, negativas, muito negativas e neutras. Sem a utilização destes gráficos a análise dos resultados poderia se tornar dispendiosa e/ou pouco notável.

2.4. *NetBeans*

O *NetBeans* é um ambiente integrado para desenvolvimento de software (*IDE*) para a plataforma Java. Ele foi utilizado durante a etapa de desenvolvimento do protótipo. O *NetBeans* permite o desenvolvimento para "*Java desktop*, móvel e aplicações web, além de fornecer ótimas ferramentas para *PHP* e C/C++. Ele é gratuito e de código aberto e tem uma grande comunidade de usuários e desenvolvedores ao redor do mundo" (NETBEANS, 2013, online).

No desenvolvimento do protótipo foi utilizado o *NetBeans* na versão 7.3. Para tanto, foi necessário instalar o *plugin* "*Groovy* e *Grails"* que é o módulo que possibilita criar e editar aplicações utilizando o *Grails Framework.* A vantagem de utilizar uma *IDE* como o *NetBeans*, é que todos os scripts do *Grails* podem ser acionados sem a necessidade do desenvolvedor digitar os comandos no *Shell*. Isto acorre porque ao instalar o *plugin* "*Groovy e Grails*", os *scripts* serão vinculados à ações devidas nos menus e botões do *NetBeans.*

2.5. Mídias Sociais

As Mídias Sociais são meios que viabilizam a interação social entre pessoas. Trata-se de um conceito que surgiu antes mesmo das tecnologias da informação, incluindo a internet. São formas de comunicação cujo conteúdo é criado de muitos para muitos. Conforme Newson (2009, p. 49, tradução nossa), o termo mídia social foi criado por Chris Shipley, co-fundador da *Guidewire Group* (empresa situada em San Francisco que realizava pesquisas e publicava relatórios sobre tendências de tecnologia). Shipley utilizou o termo mídia social para representar ferramentas e utilidades que permitam a comunicação da informação, participação e colaboração.

Segundo o mesmo autor, Shipley enquadrou no conceito das mídias sociais as redes sociais e profissionais, as *wikis*, os *podcasting* e *video casting*, os *virtual worlds* e o social *bookmarking*. Embora as mídias sociais já existissem a muito tempo, as mídias sociais recentes estão proporcionando uma comunicação mais abrangente. Nestas, o usuário pode

interagir com seu conteúdo, compartilhando, comentando e acionando outras interações específicas de cada mídia.

Neste trabalho foram utilizadas três Mídias Sociais para provimento do conteúdo necessário para a execução do protótipo de mineração de automática de opiniões proposto. Num primeiro momento foi utilizado apenas o *Facebook*. Depois foram utilizados no protótipo o *Twitter* e o *Google Plus*. Os três disponibilizam suas *APIs* para realização de consultas de postagens e outras operações.

Conforme a abrangência do conceito de mídia social definido por Shipley, as redes sociais (sites de relacionamentos utilizados por pessoas e por organizações que, geralmente, possuem interesses em comum) fazem parte da mídia social. Isto significa que o *Facebook*, o *Twitter* e o *Google Plus* por serem redes sociais são também mídias sociais (como já estavam sendo tratadas anteriormente neste trabalho). No entanto, estas mídias sociais serão referidas no decorrer do texto deste trabalho apenas como redes sociais. Esta escolha foi baseada na ideia de utilizar o termo mais específico. Além das redes sociais utilizadas no protótipo existem outras redes sociais disponíveis na *web*: *MySpace* (rede social sobre músicas), *LikendIn* (rede social profissional), *Instagram* (rede social de fotos), entre outras. A seguir são apresentadas cada uma das Redes Sociais utilizadas.

2.5.1. *Facebook*

O *Facebook* é uma rede social fundada em 2004, disponível em 70 idiomas e que possui cerca de 1.11 bilhões de usuários. Em 2012 a receita do *Facebook* chegou a $5.09 bilhões (SMITH, 2013, online). O *Facebook* foi fundado por Mark Zuckerberg e por seus amigos de faculdade Eduardo Saverin, Dustin Moskovitz e Chris Hughes. Na época os fundadores estudavam em *Havard*, Estados Unidos e eram colegas de quarto.

Em maio de 2007 o *Facebook* passou a manter a sua *API* aberta aos desenvolvedores independentes, permitindo que os mesmos desenvolvessem suas aplicações utilizando dados do *Facebook* (SBARAI, 2013, online). No *Facebook* são disponibilizadas 7 *APIs* aos desenvolvedores: *Graph API*[4]*, Facebook Query Language (FQL)*, *Open Graph*, *Dialogs*, *Chat*, *Localization and translation* e *Ads API*. Neste trabalho foi utilizado o *Graph API* dado

[4] *Graph API:* meio principal para obter e adicionar dados no *Facebook.* Mais informações sobre esta *API* podem ser obtidas na página <*https://developers.facebook.com/docs/reference/api/*>

o fato de que, dentre as suas funcionalidades, existe a opção de busca na linha do tempo (*timeline*[5]) dos usuários.

2.5.2. *Twitter*

O *Twitter* é uma rede social e um servidor de *microblogging* fundada por Jack Dorsey no ano 2006. O *Twitter* é composto por pequenas postagens de informação chamadas *Tweets*, sendo possível visualizar fotos, vídeos e conversas diretamente nos *Tweets* (*TWITTER*, 2013, online).

No total, o *Twitter* possui cerca 500 milhões de usuários. Porém, apenas 200 milhões são considerados usuários ativos mensalmente. O *Twitter* processa em média 400 milhões de *Tweets* por dia e já conta com mais de 170 bilhões de *Tweets* recebidos desde quando foi lançado (SMITH, 2013, online).

2.5.3. *Google Plus*

Rede social fundada no ano de 2011 pelo *Google Inc*. O *Google Plus* (*Google+*) que conta com 343 milhões de usuários ativos, recebe mensalmente acessos de 20 milhões de usuários realizados através de dispositivos móveis (SMITH, 2013, online).

O *Google+* é a grande aposta do *Google* para superar o *Facebook*. Para tanto, conta com novos recursos para atrair mais usuários: Círculos (grupos de amigos), *Sparks* (sugestões de conteúdo), *Hangouts* (chat por vídeo) e *Huddles* (chat em grupo).

2.6. APIs das redes sociais utilizadas

Nas *APIs* utilizadas foi adotado o formato *JavaScript Object Notation* (*JSON*[6]*)* para retornar o conteúdo consultado. Na Figura 10 pode ser observada a função que foi utilizada para realizar a leitura das informações retornadas pelas *API*s. A função recebe uma URL como parâmetro e retorna um objeto *JSON*.

---

[5] *Timeline*: exposição organizada de informações relativas ao perfil de um usuário do *Facebook* (inclui textos, imagens e vídeos).

[6] *JSON*: notação de objetos JavaScript muito utilizado na troca de dados entre sistemas. Para mais informações pode-se consultar no site <*http://json.org/json-pt.html*>

```
1   public static JSONObject readJsonFromUrl(String url)
2                         throws IOException, JSONException {
3     InputStream is = new URL(url).openStream();
4     try {
5       BufferedReader rd =
6             new BufferedReader
7                     (new InputStreamReader(is, Charset.forName("UTF-8")));
8       String jsonText = readAll(rd);
9       JSONObject json = new JSONObject(jsonText);
10      return json;
11    } finally {
12      is.close();
13    }
14  }
```

Figura 10 - Função para leitura de *JSON*

O método apresentado na Figura 10 realiza a leitura de um texto a partir de um fluxo de caracteres de entrada e depois o converte para um objeto *JSON*. Pelo fato desta função acessar a página dos resultados de cada *API* o tempo de resposta do método depende principalmente da velocidade de conexão disponível. Após a execução desta função é necessário converter os dados obtidos em *JSON* para uma lista do *Groovy* que contenha os itens que foram retornados.

Para adquirir uma previsão do tamanho do retorno das *APIs*, utilizou-se o recurso *Developer Tools* (opção *Network*), disponível no navegador *Google Chrome*. Foi identificado que cada retorno ocupa em média de 10kb a 25kb. A variação se deve principalmente pela quantidade de postagens relacionadas com a pesquisa.

No que diz respeito ao retorno dos dados das *APIs* utilizadas no trabalho, constatou-se que, cada uma delas possui uma estrutura própria para o retorno dos dados, ou seja, o conjunto de dados de cada propriedade presente no *JSON* é apresentado de uma forma única em cada rede social escolhida. Por isso, cada uma das etapas de consulta e conversão de dados foi executada individualmente.

2.7. METODOLOGIA

A metodologia adotada neste trabalho envolveu inúmeras fases que incluíram atividades relacionadas à pesquisa, à escrita da monografia e ao desenvolvimento do protótipo de classificação automática de opiniões. Estas fases serão descritas a seguir:

- Na primeira fase foram realizadas pesquisas bibliográficas de natureza aplicada com o intuito de fornecer conhecimento para desenvolver o protótipo de categorização automática de opiniões. Nesta etapa, foram realizadas pesquisas sobre os principais

conceitos que seriam utilizados para a execução do trabalho. As pesquisas foram realizadas em documentos acadêmicos tais como artigos, monografias, dissertações, teses e livros. Após a realização desta atividade, organizou-se a estrutura do referencial teórico e foi realizada a escrita da revisão bibliográfica. Vale ressaltar que, dentre os conceitos estudados, destaca-se a mineração de dados e a mineração de textos, visto que compreendem as teorias que deram margem à aplicação dos demais conceitos e técnicas apresentados no trabalho.

- A segunda fase envolveu o estudo da ferramenta de mineração de dados *Weka API*. O objetivo era estudá-la e, se fosse o caso, utilizá-la (totalmente ou parcialmente) no trabalho. Além de verificar a possibilidade de utilizar a *Weka API* no protótipo, foram estudados os algoritmos providos e a forma de utilizá-los. Nesta fase também foi realizado um estudo sobre como criar uma base de dados do tipo *ARFF*. O objetivo era armazenar as opiniões pré-classificadas para posterior treinamento do algoritmo *SVM* ou *KNN*.
- A terceira fase envolveu uma pesquisa sobre a forma de obtenção do conteúdo que seria classificado. Para tanto, foram escolhidas três redes sociais das quais se pretendia obter as postagens para compor a base de opiniões do protótipo de classificação. Nesta pesquisa foi verificada a existência de *APIs* que disponibilizam o conteúdo para ser utilizado pelo protótipo.

  As *APIs* foram descobertas nas páginas de desenvolvedores das redes sociais. Para escolher quais utilizar dentre as *APIs* disponíveis, foi realizado um estudo sobre as funcionalidades de cada *API* para verificar as que atendiam a necessidade do protótipo. As *APIs* verificadas funcionam através de requisições HTTP e retornam o resultado do processamento em *JSON*. Por isso, foi preciso estudar uma forma de ler os dados que eram retornados. Outra etapa consistiu num estudo sobre como converter os dados obtidos para que pudessem ser utilizados pelo protótipo. Ao pesquisar sobre o tema identificou-se que existia a biblioteca ***google-gson*** que realiza a conversão de *JSON* para *Java*.
- Na sequência, foi desenvolvido o protótipo da aplicação que, para facilitar, é composto por subfases:
  - Inicialmente foi desenvolvida uma biblioteca de mineração de dados que serve como base para o protótipo. Nela foi desenvolvido um algoritmo de classificação e ainda outros métodos para organização do texto. A biblioteca foi desenvolvida na plataforma Java, utilizando o JDK 1.7.

- Nesta biblioteca foi utilizado um conjunto de palavras que serviram como base para a realização da classificação. Estas palavras foram selecionadas no decorrer do desenvolvimento do protótipo e adicionadas a 14 arquivos de texto. Optou-se por armazená-las em arquivos de texto por ser uma forma em que o conteúdo pode ser acessado e alterado com facilidade. Foram criados os seguintes arquivos para armazenar as palavras: *stopwords.txt* (arquivo que contêm as *stopwords*), *classifiedWords.arff* (arquivo que contêm as palavras pré-classificadas em positivas ou negativas), *compoundWords.txt* (arquivo que contêm palavras compostas e expressões idiomáticas), *subjectiviesWords.txt* (arquivo que contém palavras subjetivas), *denialWords.txt* (arquivo que contêm as palavras que negam uma determinada afirmação), *pronounsAndArticles.txt* (contêm pronomes e artigos), *afirmativesWords.txt* (contêm palavras afirmativas), *upWords.txt* (contêm palavras que aumentam o valor semântico de outra palavra), *downWords.txt* (contêm palavras que diminuem o valor semântico de outra palavra), *positiveWeakWords.txt* (contem palavras positivas cujo valor semântico seja baixo), *negativeWeakWords.txt* (contem palavras negativas cujo valor semântico seja baixo), *positiveStrongWords.txt* (contem palavras positivas cujo valor semântico seja alto) *e negativeStrongWords.txt* (contêm palavras negativas cujo valor semântico seja alto) e entitiesIgnore.txt (contêm palavras que são desconsideradas na realização da classificação).
- Para utilizar os arquivos supracitados foi criada uma classe que realiza a leitura dos mesmos, no caso a classe **TextReader.** Também foi criada a classe **TextDataBase** que implementa as funcionalidades de verificação de palavras (funções onde são comparadas as palavras presentes numa determinada opinião com outras palavras presentes na base de palavras – arquivos de palavras criados – a fim de identificar características de cada palavra). Das funcionalidades implementadas pode-se citar as utilizadas para identificar se uma palavra é uma *stopword*, um termo subjetivo, uma negação, entre outras.
- Para processar o texto das "opiniões" (postagens advindas das redes sociais escolhidas) foi criada a classe **WordProcessor**. Nela foram desenvolvidas as funcionalidades necessárias para organizar os textos antes e durante a classificação. Dentre as funcionalidades desenvolvidas estão a remoção de

stopwords, o tratamento de expressões idiomáticas, a verificação de possível igualdade ou flexão entre palavras, entre outras.

- Na sequência, foi desenvolvida uma aplicação *web* utilizando o *framework Grails*. Esta aplicação foi desenvolvida na linguagem *Groovy*, utilizando o *Grails* na versão 2.1.1. A primeira parte a ser desenvolvida nesta aplicação foi a consulta das opiniões utilizando as *APIs* das redes sociais determinadas. Posteriormente foi realizada a integração da biblioteca de mineração desenvolvida.
- Para organizar os dados de uma postagem foi criada a classe *Post* com as seguintes propriedades: *message*, *user* (identificador do usuário), *username* (nome do usuário), *URL* (*URL* do perfil), *profileImageURL* (*URL* da imagem do perfil), *date* (data da postagem) e *predicao* (classificação obtida pela postagem). Após isto, foi desenvolvida a interface da aplicação (*view* **classificar.gsp**) pela qual o usuário realiza as buscas.
- A interface da aplicação *Grails* foi implementada para apresentar os resultados do seguinte modo: na parte inferior da aplicação é demonstrada a lista de opiniões utilizadas na classificação automática. Para possibilitar a filtragem dessa lista foi implementada a funcionalidade de limitar a exibição dos resultados por tipo de classificação da opinião (todas, positivas, negativas e neutras). Esta opção de filtro foi criada utilizando as funções *hide* e *show* disponibilizadas no *JQuery*.
- Ainda para compor a interface do sistema, foi realizado um estudo sobre a biblioteca *Highcharts JS*. O estudo foi motivado pela necessidade de criar um gráfico para exibir a quantidade de opiniões encontradas em cada classe. Os dados exibidos na *view* **classificar.gsp** são enviados pelo **ClassificadorController.** Entre os dados enviados estão um *ArrayList* de objetos **Post**, a quantidade de opiniões consultadas e a quantidades de opiniões agrupadas em cada classe.

- Após a conclusão do desenvolvimento do protótipo, iniciou-se a fase de testes. Foram realizados testes tanto com opiniões intencionais (opiniões que foram escritas especificamente para se caracterizar em alguma classe), quanto com opiniões obtidas por meio das *APIs* das redes sociais.

3 RESULTADOS E DISCUSSÃO

Este capítulo contém informações sobre os resultados obtidos na realização deste trabalho. Para tanto, neste é apresentada a forma de obtenção da base de opiniões utilizada, a arquitetura do protótipo desenvolvido, as dificuldades e as soluções adotadas. Na sequência, será abordado o processo de criação da base de opiniões, disponibilizada a partir das redes sociais escolhidas (*Facebook*, *Google Plus* e *Twitter*).

3.1. BASE DE OPINIÕES

A base de opiniões é consultada em tempo real utilizando as *APIs* do *Facebook*, do *Twitter* e do *Google Plus*, a partir de um ou mais termos inseridos em um campo de pesquisa disponível na página inicial do protótipo desenvolvido. Na Figura 11 é apresentada a forma pela qual as informações são obtidas.

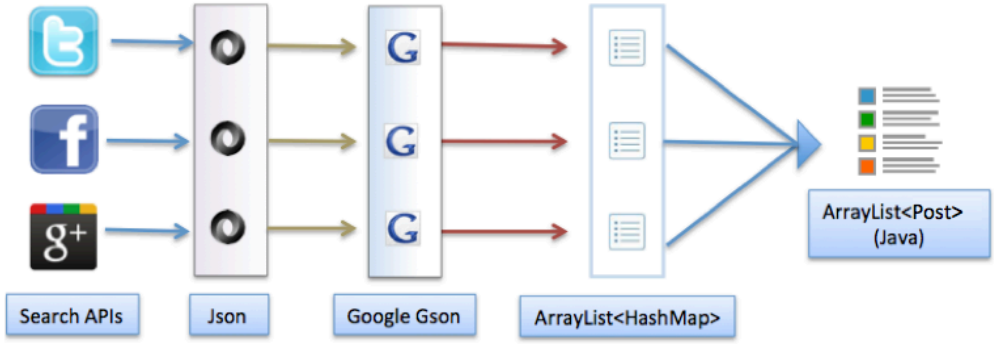

Figura 11 - Sequência de obtenção da base de opiniões

Utilizando as *APIs* de busca disponibilizadas pelas redes sociais utilizadas consegue-se obter as informações no formato *JSON*. Este formato pode ser interpretado, por exemplo, por um *browser* utilizando a linguagem *JavaScript* (linguagem para programação *client-side* em navegadores web). Porém, para o *Groovy*, o arquivo *JSON* não passa de uma *string* e, por isso, deve ser convertido para um objeto *Java.* Uma das formas de realizar esta conversão é utilizando o *Google Gson* (uma *API* fornecida pelo *Google* que possibilita a *conversão entre objetos Java e JSON*). Para converter um *JSON* para *Groovy* utilizando *Google Gson* deve-se informar uma classe *Groovy* que corresponda com a arquitetura do arquivo *JSON*. No caso, foi utilizada uma classe genérica na conversão do *JSON,* a classe *Map.* Posteriormente, o

conteúdo de interesse foi inserido num *ArrayList.* A Figura 12 demonstra o código utilizado na realização destas etapas.

```
1  JSONObject jsonT = JsonReader.readJsonFromUrl(urlJsonTwitter);
2  JSONObject jsonG = JsonReader.readJsonFromUrl(urlJsonGooglePlus);
3  JSONObject jsonF = JsonReader.readJsonFromUrl(urlJsonFacebook);
4
5  ArrayList<HashMap> dadosFacebook = new Gson().fromJson(jsonT.toString(), Map.class)?.get("data");
6  ArrayList<HashMap> dadosTwitter = new Gson().fromJson(jsonG.toString(), Map.class)?.get("results");
7  ArrayList<HashMap> dadosGooglePluss = new Gson().fromJson(jsonF.toString(), Map.class)?.get("items");
8
9  ArrayList<Post> opinions = new ArrayList<Post>();
10
11 dadosTwitter?.each {
12     Post post = new Post()
13     post.message = it.get("text")?.toString()
14     post.user = it.get("from_user")?.toString()
15     post.username = it.get("from_user_name")?.toString()
16     post.url = "https://twitter.com/" + post.user
17     post.profileImageURL = it.get("profile_image_url")?.toString()
18     post.date = it.get("created_at")?.toString()
19     opinions.add(post);
20 }
21 dadosGooglePluss?.each {
22     Post post = new Post()
23     post.message = it.get("title").toString()
24     post.user = it.get("from_user")?.toString()
25     post.url = it.get("url")?.toString()
26     post.username = it.get("actor")?.get("displayName")?.toString()
27     post.profileImageURL = it.get("actor")?.get("image")?.get("url")?.toString()
28     post.date = it.get("published")?.toString()
29     opinions.add(post);
30 }
31 dadosFacebook?.each {
32     Post post = new Post()
33     post.message = it.get("message").toString()
34     post.username = it.get("from")?.get("name")?.toString()
35     post.url = "https://www.facebook.com/" + it.get("from")?.get("id")?.toString()
36     post.profileImageURL = "http://graph.facebook.com/"+ it.get("from")?.get("id")?.toString() +"/picture"
37     post.date = it.get("created_time")?.toString()
38     opinions.add(post);
39 }
```

Figura 12 - Código utilizado para obter os dados das redes sociais

As instruções da linha 1 à 3 dizem respeito à leitura do *JSON* a partir das *URLs* disponibilizadas pelas *APIs*. Foi necessário realizar um cadastro de desenvolvedor para utilizar a *API* do *Google+*. Após o cadastro o *Google* fornece uma chave que deve ser adicionada à *URL* referente à *API*, a fim de medir e controlar a utilização dos recursos advinda da aplicação cadastrada. No caso do *Twitter* e do *Facebook,* no decorrer do período de desenvolvimento, não foi necessário realizar nenhum cadastro. Porém, após a finalização do desenvolvimento do protótipo o *Twitter* atualizou a sua *API* e também as suas regras de utilização. Esta atualização fez com que fosse necessário realizar um cadastro de desenvolvedor do *Twitter* e alterar o código relacionado nas consultas.

Cada uma das *APIs* utilizadas organiza as informações numa propriedade diferente o que não caracteriza uma mudança na técnica de obtenção dos dados, mas apenas uma mudança no modo de acessar a propriedade onde os dados foram organizados.

As postagens são armazenadas primeiramente em uma lista de *HashMaps*. Isto porque cada postagem contém, além da mensagem publicada, outras informações como: autor, data, *URL* do perfil, comentários entre outros. O *HashMap* organiza os dados com uma chave para cada valor, o que difere de outras classes *Maps* pois ela não permite a existência de chaves duplicadas. Neste ponto, é importante ressaltar que as propriedades que armazenam as postagens disponibilizadas pelas *APIs* não são as mesmas, como pode ser visto na linha 13, 23 e 33 do código apresentado, sendo que uma utiliza a propriedade '*text'* (*Twitter*), outra '*title'* (*Google Plus*) e a última '*message'* (*Facebook*).

Na sequência do código, na linha 9, é definida a lista que agrupará as informações consultadas e, por fim, as listas com as informações de cada mídia social são percorridas e inseridas na lista que será utilizada na mineração dos dados. Foi utilizada a classe **Post** para organizar as postagens e simplificar o acesso aos dados. Na próxima seção será apresentada a arquitetura do protótipo de mineração de opiniões automática que foi desenvolvido.

3.2. ARQUITETURA DA APLICAÇÃO

Nesta seção serão apresentadas todas as partes que constituem a arquitetura do protótipo de mineração de opiniões automática desenvolvido neste trabalho. Serão apresentadas as classes e bases de dados utilizadas, além de abordar uma visão geral sobre a arquitetura adotada. Na Figura 13 é apresentada a arquitetura do protótipo.

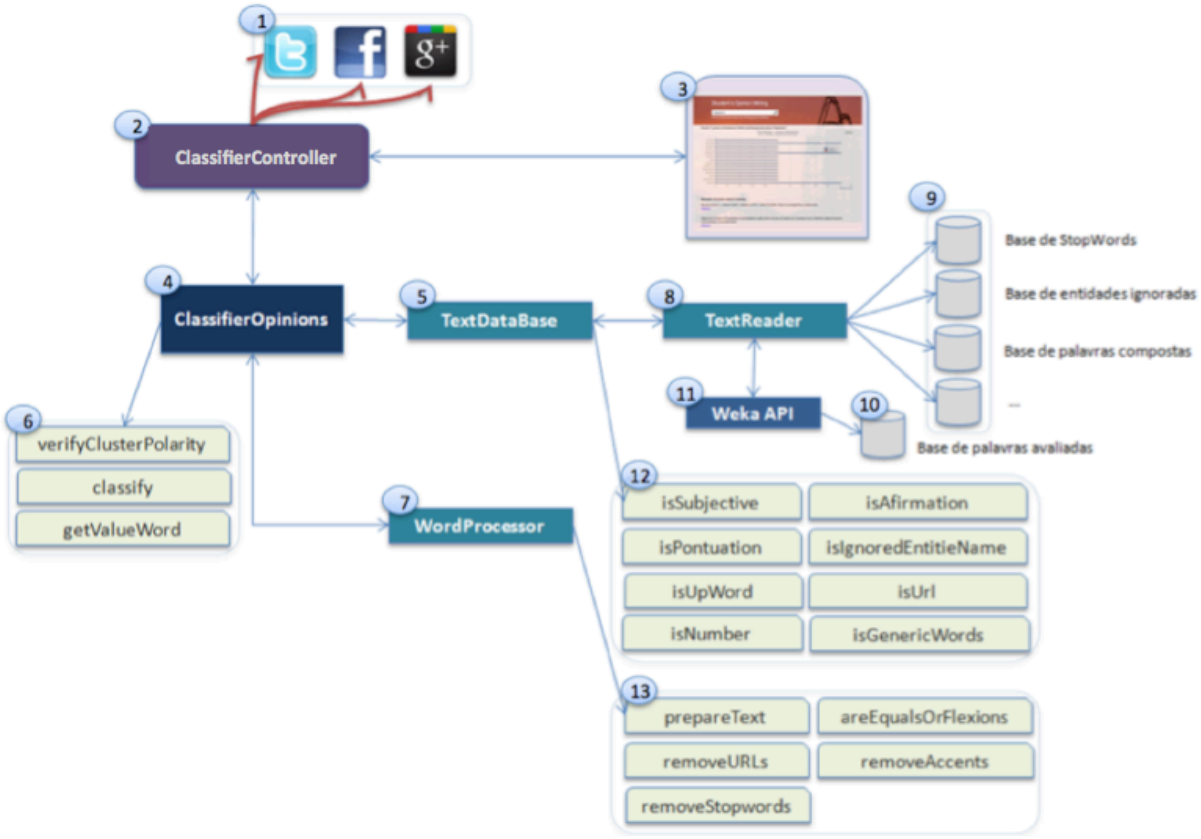

Figura 13 - Arquitetura do protótipo de classificação automática de opiniões

A arquitetura apresentada na Figura 13 pode ser organizada em 4 partes para obter um melhor entendimento: base de opiniões, biblioteca de mineração, aplicação *Grails*, base de palavras utilizadas como apoio na classificação das opiniões. Abaixo serão descritos a utilidade e modo de funcionamento das mesmas:

- **Aplicação *Grails*:** aplicação desenvolvida para realizar a mineração das opiniões advindas das redes sociais que compõem a base de opiniões. A aplicação será explicada na sessão 3.2.1.5.
- **Base de opiniões**: conforme explicado na seção anterior, a base de opiniões é obtida através das *APIs* do *Twitter, Facebook e Google Plus* que permitem a consulta de postagens através de termos relacionados enviados para as *APIs*. A obtenção dessas opiniões é realizada no item 2 da Figura 13 e será explicada no tópico que aborda sobre a aplicação *Grails*.
- **Biblioteca de mineração**: conjunto de métodos e arquivos que provê as funcionalidades necessárias para o funcionamento da aplicação de mineração de opiniões. A biblioteca em questão corresponde à agregação dos itens indicados na

Figura 13 do item de número 4 ao item de número 13. Esta parte será melhor explicada na seção 4.2.1.

- **Base de palavras:** foram utilizadas muitas palavras como base para a etapa de classificação. Estas palavras foram separadas em arquivos distintos. Dentre estes arquivos estão o arquivo que contêm as palavras pré-classificadas e o arquivo de *stopwords* (necessário para remover palavras desnecessárias para a classificação) cujo conceito foi abordado na seção1.4.3.

3.2.1. *Biblioteca de mineração*

A biblioteca de mineração é o núcleo do protótipo. Ela implementa as funcionalidades necessárias para realizar as classificações das opiniões. É importante ressaltar que, nesta fase, não foram utilizados nenhum dos algoritmos disponibilizados pela *Weka API*. Isto porque estes algoritmos realizam a classificação utilizando a técnica de aprendizagem de máquina. Esta técnica requer que um conjunto de opiniões seja classificado anteriormente para que seja possível realizar as novas classificações. Esta restrição dificulta a classificação de opiniões das quais não se conhece o contexto. Assim, como a biblioteca de mineração terá que trabalhar com contextos desconhecidos, deve funcionar da forma mais abrangente possível. Por isto foi utilizado um conjunto de palavras pré-classificadas em positivas ou negativas a fim de se enquadrarem em qualquer contexto.

Para uma melhor explicação, a biblioteca de mineração foi dividida em 4 partes (classes) principais: ***ClassifierOptions***, ***TextDataBase***, ***TextReader***, ***WordProcessor***. Estas classes serão explicadas abaixo:

3.2.1.1. ClassifierOptions

Corresponde à classe principal da biblioteca que contêm os métodos para carregar as palavras-base, obter a nota de uma palavra isolada e realizar a classificação de uma opinião. Para que seja possível a classificação, o método responsável percorre cada opinião e a converte para um vetor de palavras utilizando a técnica de tokenização. Porém, apesar da separação, as palavras não são classificadas separadamente. São analisadas, ademais, algumas palavras que precedem uma palavra que esteja em processo de avaliação.

Para tanto, decidiu-se de forma empírica utilizar as 3 palavras anteriores para tentar obter uma melhor classificação. Escolheu-se por não analisar a relação de uma determinada palavra com mais palavras do texto por acreditar que, conforme uma palavra se distancia de outra, menor será o nível de relação entre ambas. Outra análise realizada incide em verificar

se a palavra anterior ou a próxima correspondem a uma “pontuação” o que significaria a finalização de um raciocínio ou de um assunto. Esta verificação influência para que a classificação tenha uma margem de erro menor.

O método que realiza a classificação das opiniões verifica também se as palavras são negações, se aumentam ou diminuem o significado de uma palavra e se são subjetivas. A Figura 14 contém uma parte do código que realiza a classificação das opiniões.

```
1  for (int w = 0; w < text.length; w++) {
2      String strTerm = text[w].toLowerCase();
3      boolean denial = getTextDB().isDenial(strTerm);
4      if (denial) {
5          if ((numTestsRevertDenial > 0)
6                  && (numTestsRevertDenial < 3 || getTextDB().isPontuation(text[w]))) {
7              numTestsRevertDenial = 0;
8          } else {
9              numTestsRevertDenial = 3;
10         }
11     }
12     if (numTestsRevertDenial == 0) {
13         denial = false;
14     } else {
15         denial = true;
16         numTestsRevertDenial--;
17     }
18     if (!textDB.isStopWords(strTerm)) {
19         for (int i = 1; i < getTrainData().numAttributes(); i++) {
20             String strAtributeName = CharFilter.replaceSpecial(getTrainData().attribute(i).name());
21             if (TextProcess.areEqualsOrFlexions(strTerm, strAtributeName)) {
22                 double note =
23                         getAvaliativeNote
24                                 (strTerm, i, denial,
25                                 getTextDB().isSubjective(strTerm), getTextDB().isUpWord(strTerm)),
26                                 getTextDB().isDownWord(strTerm));
27                 if (note != 0) {
28                     denial = false;
29                     subjective = false;
30                     upward = false;
31                     downward = false;
32                 }
33                 neutral = false;
34             }
35         }
36     }
37 }
```

Figura 14 - Trecho de código que percorre um vetor com palavras e as classifica

Do modo que o classificador foi implementado, o método **getAvaliativeNote** chamado a partir da linha 23 até a 26 da Figura 14 pode prever (tratar) inclusive a “negação sobre negação” o que significa que o protótipo não analisa somente as palavras separadamente, mas faz verificações que podem reduzir bastante as possível falhas na classificação. Por exemplo, as palavras identificadas como interrogação estão sendo caracterizadas como neutras. Isto acontece porque, quando houver uma interrogação, possivelmente o texto corresponderá a uma pergunta e não a uma opinião.

3.2.1.2. TextDataBase

O **TextDataBase** é uma classe utilizada para realizar verificações entre palavras. É também a classe responsável pelo acesso a todas as bases de palavras criadas. Dentre os dados que podem ser acessados por meio desta classe estão as palavras pré-avaliadas (positivas e negativas), as *stopwords*, as vogais e artigos, palavras afirmativas, palavras agregadoras de valor sintático, palavras depreciadoras de valor sintático, palavras compostas, palavras subjetivas, negações, pontuações e outras. Nesta classe é utilizada uma parte da *Weka API* responsável pela leitura das palavras pré-classificadas em negativas ou positivas. Toda a base de palavras foi organizada em arquivos de texto. A Figura 15 demonstra um dos métodos da classe **TextDataBase.**

```
1  /**
2   * Verifica se o termo é uma negação.
3   * @author Paulo H.
4   * @param term
5   * @return boolean
6   */
7  public boolean isDenial(String term)  throws IOException{
8      if(denialWords == null){
9          denialWords  = new ArrayList();
10         String[] denials = textReader.readFile(getPathDenialWords()).split(";");
11         denialWords.addAll(Arrays.asList(denials));
12     }
13     return denialWords.contains(CharFilter.removerAcentuacao(term.toLowerCase()));
14 }
```

Figura 15 - Método para identificação de palavras que invertam o sentido de outras

O funcionamento do método ocorre da seguinte forma: na primeira vez que ele for acionado serão carregadas as palavras que invertem o sentido de outras palavras. Dentre as palavras utilizadas para compor esta base encontram-se: nao, jamais, nunca, de forma alguma, entre outras. Após consultar as palavras, elas são armazenadas numa lista denominada **denialWords.**

Depois de carregar estas palavras, o sistema verifica se o termo passado como parâmetro está contido na lista **denialWords**. Este procedimento é muito importante por evitar que o sistema classifique uma opinião baseado apenas no valor semântico das palavras que compõe a base das palavras pré-classificadas.

3.2.1.3. TextReader

A classe **TextReader** foi criada para realizar a leitura dos arquivos de texto que compõem a base de palavras. Foram implementados 2 métodos nessa classe, sendo que ambos realizam a leitura de arquivos. O primeiro método, **readFile**, é específico para leitura de

arquivos que não são muito grandes. No segundo método, utilizado para ler grandes arquivos de texto, a alocação de memória para o *buffer* (espaço de memória utilizada para escrita e leitura de dados) é maior, e o processo é realizado utilizando múltiplas *threads* (segmentos de execução de um programa que podem ser executados concorrentemente). A seguir, na Figura 16, são apresentados os métodos da classe **TextReader**.

```
1  /**
2   * Realiza a leitura de um arquivo de texto pequeno.
3   * @author Paulo H.
4   * @param fileName
5   * @return String
6   */
7  public String readFile(String fileName) throws IOException{
8          BufferedReader in = new BufferedReader(new FileReader(fileName));
9          String str;
10         String strReturn = "";
11         while ((str = in.readLine()) != null) {
12             strReturn += str + ";";
13         }
14         in.close();
15         return strReturn;
16
17 }
18
19 /**
20  * Realiza a leitura de um arquivo de texto muito grande.
21  * @author Paulo H.
22  * @param aFile, inChannel
23  * @return void
24  */
25 public void readLargeFiles(RandomAccessFile aFile, FileChannel inChannel) throws IOException{
26         String ignored = "";
27         //create buffer with capacity of 48 bytes
28         ByteBuffer buf = ByteBuffer.allocate(48);
29         int bytesRead = inChannel.read(buf); //read into buffer.
30         while (bytesRead != -1) {
31           buf.flip();  //make buffer ready for read
32
33           while(buf.hasRemaining()){
34               ignored += (char) buf.get();
35           }
36           String[] igs = ignored.split("\n");
37           TextDataBase.ignoredsTree.addAll(Arrays.asList(igs));
38           ignored = "";
39           buf.clear(); //make buffer ready for writing
40           bytesRead = inChannel.read(buf);
41         }
42         aFile.close();
43 }
```

Figura 16 - Métodos da classe TextReader

O primeiro método realiza a leitura do início ao fim do arquivo, linha por linha. Como pode ser visto na linha 12 cada linha percorrida está sendo incrementada na variável **strReturn** que será retornada no fim da execução do método.

O método **readLargeFiles** realiza o acesso não sequencial e/ou aleatório ao arquivo. O parâmetro **inChannel** é utilizado para referenciar a posição atual do arquivo que é lido na linha 29. Do modo que o método foi implementado utilizando o método **allocate** conforme visto na linha 28, o escalonamento da leitura do arquivo está sendo gerenciado pela máquina virtual do *Java*. A leitura é realizada de 48 bytes por vez.

3.2.1.4. WordProcessor

Classe responsável por todas as operações de tratamento e organização de palavras. Entre as funcionalidades presentes estão a verificação de igualdade ou flexão entre uma palavra e outra; detecção de palavras compostas; remoção de *stopwords* e caracteres especiais, entre outros. A seguir, Figura 17, é apresentado o método de remoção de *stopwords*.

```
1  /**
2  * Remove as Stopwords do texto.
3  * @author Paulo H.
4  * @param text
5  * @return String
6  */
7  public String removeStopwords(String text) throws IOException{
8      String[] words = text.split(" ");
9      String saida = "";
10     for(int i = 0; i < words.length; i++){
11         if(textDataBase.isStopWords(words[i])){
12             words[i] = "";
13         }
14     }
15     for(int i = 0; i < words.length; i++){
16         saida += words[i] + " ";
17     }
18     return saida;
19 }
```

Figura 17 - Método de remoção de Stopwords

Conforme apresentado na Figura 17, o método recebe e retorna uma "String" e funciona da seguinte forma: ao ser acionado, o método realiza a etapa de tokenização e armazena o resultado em um *array* de *strings*. Este passo é realizado na linha 8. Em seguida, percorre a lista de palavras e verifica para cada palavra se a mesma é uma *stopword* e, caso seja verdade, a palavra é substituída por uma "String" vazia. Depois a lista de palavras é novamente percorrida para adicionar as palavras à variável que será retornada. Na Figura 18 é apresentado outro método responsável por identificar e organizar as palavras compostas.

```
1  /**
2   * Pesquisa e organiza as palávras compostas no texto.
3   *
4   * @author Paulo H.
5   * @param text
6   * @return String
7   */
8  public String organizeCompoundWords(String text) throws IOException {
9      String compoundWords = CharFilter.removerAcentuacao(textDataBase.getCompoundWords().toLowerCase());
10     for (String cw : compoundWords.split(";")) {
11         if (text.toLowerCase().contains(CharFilter.removerAcentuacao(cw))) {
12             text = text.replace(cw, cw.replace(" ", "##"));
13         }
14     }
15     return text;
16 }
```

Figura 18 - Método para encontrar e tratar as palavras compostas

O método demonstrado na Figura 18 permite tratar algumas palavras que juntas podem ter um sentido positivo ou negativo. O método trata além das palavras compostas as expressões idiomáticas adicionadas no arquivo correspondente. Algumas das expressões utilizadas no protótipo são: “perde e ganha”, “menina dos meus olhos”, “pão e circo”, “feito nas coxas”, “deixar na mão”, “tirar de letra”, “com a faca e o queijo na mão”, “negócio da China”, entre outros.

Na linha 9 do código na Figura 18 as palavras compostas (expressões idiomáticas) são armazenadas na variável **compoundWords**. Depois, da linha 10 à 14 as palavras compostas são percorridas para que seja verificada a existência de alguma delas no texto recebido como parâmetro, caso exista alguma destas palavras, as mesmas terão os espaços entre si preenchidos com “##” (duas cerquilhas). A partir disto, a expressão será tratada como se fosse apenas uma palavra.

3.2.1.5. Aplicação *Grails*

A aplicação *Grails* é um protótipo *web* com a qual o usuário interage. Nela são utilizadas as *APIs* do *Facebook*, do *Twitter* e do *Google Plus* para consultar as postagens relacionadas com o(s) termo(s) de pesquisa. Para realizar a classificação das opiniões consultadas, a aplicação *Grails* utiliza a biblioteca de mineração de dados que foi desenvolvida neste trabalho.

O sistema é composto de 3 áreas apresentadas na Figura 19: a área de pesquisa (Área 1), gráfico representativo das opiniões encontradas (Área 2) e as opiniões obtidas das redes sociais que foram classificadas (Área 3). A Área 1 é composta pelo campo onde deve(m) ser escrito(s) o(s) termo(s) de pesquisa. A pesquisa é iniciada após o usuário acionar o botão ao lado do campo de pesquisa. Após o acionamento deste botão as opiniões serão pesquisadas através das *APIs* disponibilizadas pelo *Facebook*, *Twitter* e *Google Plus*. As pesquisas nestas *APIs* são feitas em temo de execução. Para tanto é necessário que o servidor da aplicação conecte-se à internet e o tempo de resposta está vinculado à conexão da internet. Foi utilizada como ícone do botão de pesquisa a imagem de uma alferça (ferramenta utilizada em diversos tipos de mineração). O objetivo de utilizar esta imagem é tentar criar uma identidade visual para o protótipo enfatizando o conceito de mineração de dados. O resultado da aplicação em funcionamento pode ser observado na Figura 19.

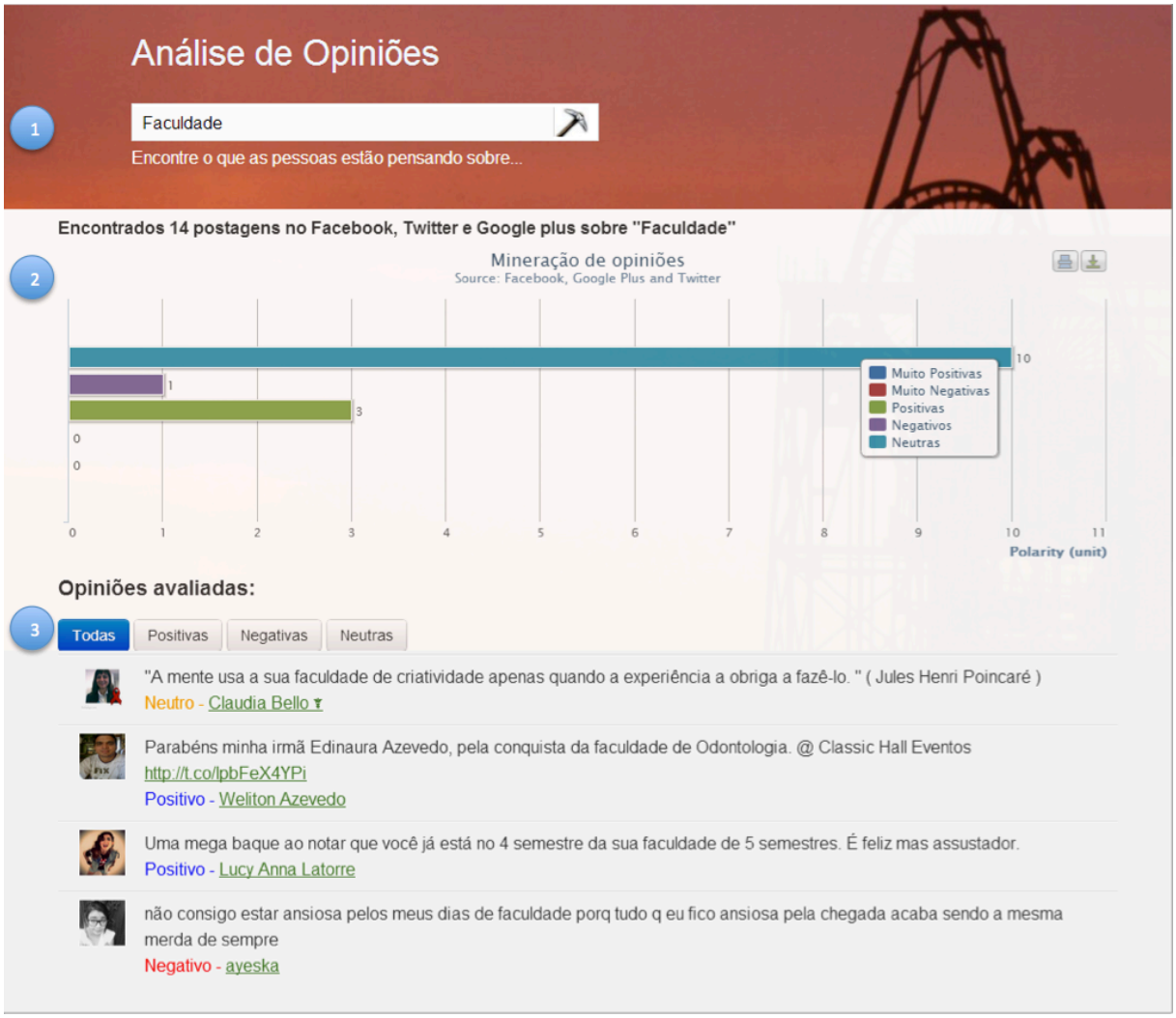

Figura 19 - Tela do protótipo

Durante o processo de classificação o sistema apresentará na Área 1, uma imagem para representar que a solicitação está sendo processada. Após o carregamento das opiniões, as mesmas serão classificadas e o sistema apresentará a quantidade de opiniões encontradas e um gráfico na Área 2 agrupando as opiniões que se enquadrem na situação positiva, muito positiva, negativa, muito negativa e neutra. Além do gráfico, serão listadas na Área 3 todas as opiniões que foram classificadas.

Cada item da listagem na Área 3 contém, além da opinião textual, o nome e a imagem do perfil da pessoa que postou o comentário, o link para o perfil e a classificação obtida. O usuário do sistema pode também solicitar a exibição de apenas um tipo de opinião que deve ser feita através de quatro botões disponibilizados acima das opiniões. As opções de listagem são: **todas** (todas as opiniões classificadas), **positivas** (as opiniões positivas e as muito positivas) e **negativas** (as opiniões negativas e as muito negativas). Logo abaixo, na Figura 20, é exibido o diagrama de atividades envolvendo o protótipo e a biblioteca de mineração.

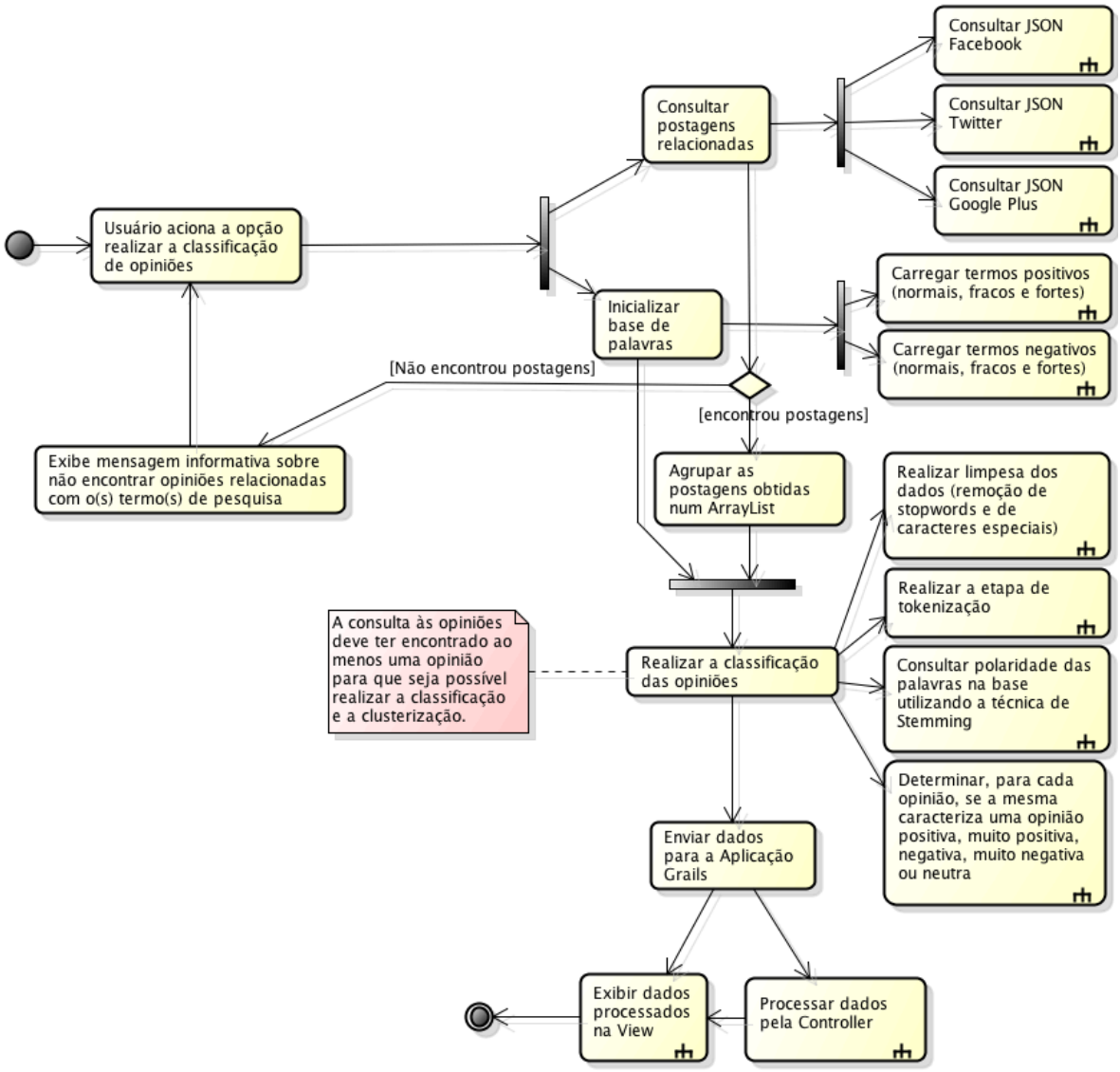

Figura 20 - Diagrama de atividades do protótipo de classificação de opiniões automática

O diagrama exibido na Figura 20 apresenta o fluxo do funcionamento do protótipo desenvolvido. Como pode ser visualizada no diagrama a primeira etapa consiste na ação do usuário solicitando a classificação das opiniões sobre um ou mais termos. A partir disto, a aplicação se comunica com as *APIs* das redes sociais escolhidas e em paralelo é carregada a base de palavras caso as mesma ainda não estejam na memória do servidor.

Posteriormente, as opiniões consultadas são agrupadas e enviadas para a classificação, momento em que será realizado um pré-processamento nos textos incluindo a remoção das *stopwords*.

Na sequência, é verificada a polaridade das palavras e o valor semântico das palavras adjacentes (para identificar se existem negações, subjetividade, entre outros) para definir se a opinião é positiva, negativa ou neutra.

3.3. Utilização do Protótipo e Testes Realizados

Para validação do protótipo foram realizados tanto testes com opiniões escritas intencionalmente quanto testes em opiniões obtidas em tempo real. 81% dos testes realizados com protótipo atenderam perfeitamente a classificação das opiniões. As principais falhas no protótipo foram ajustadas com a adição de novas palavras aos arquivos de palavras.

A fase de testes foi organizada no seguinte modo: primeiro foram testadas afirmações isoladas criadas manualmente utilizando algumas palavras que foram adicionadas à base de palavras pré-classificadas. Além destas opiniões, foram analisadas 200 opiniões advindas das redes sociais a partir de pesquisas aleatórias sobre cidades, países, instituições, etc. Nestas análises, o protótipo acertou em 81% dos casos. As situações em que o protótipo não acertou foram organizadas de acordo com o motivo do erro. Foram identificados 7 motivos: idiomas desconhecidos, erros ortográficos, ironias, gírias, expressões idiomáticas mal interpretadas, palavras compostas que não foram não previstas e equívocos do sistema. Os erros ocorreram em 19% das análises e a porcentagem geral dos tipos de ocorrência de erros pode ser visualizada na Figura 21.

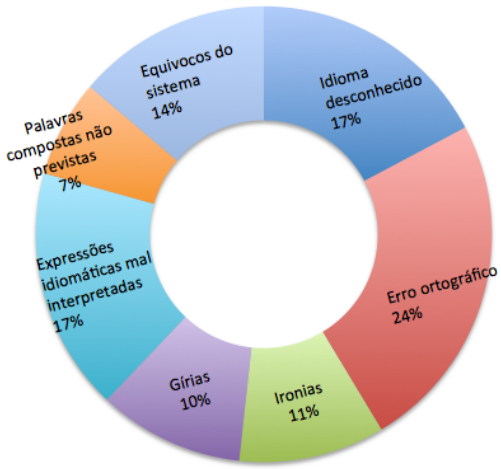

Figura 21 – Motivos dos erros identificados nos testes

Embora ocorreram falhas nos testes de opiniões em tempo real, não ocorreu nenhuma falha nos testes com opiniões elaboradas intencionalmente. Na elaboração destas opiniões,

optou-se pela seguintes estruturas de texto: (entidade + afirmação + adjetivo); (adjetivo ou verbo + complemento + entidade); (entidade + negação + adjetivo + complemento); (negação + verbo ou adjetivo); (adjetivo + entidade + interrogação); também foram realizadas combinações entre estas estruturas textuais, afim de identificar situações onde a opinião deve ser classificada em neutra. As opiniões que foram criadas são apresentadas na Tabela 4.

Tabela 4: Opiniões escritas manualmente para realização de testes

Opinião	Polaridade semântica
1. **A praia do prata é boa.** A polaridade foi definida como positiva devido à palavra "boa" estar inclusa na base de palavras positivas.	Positiva
2. **O avião da linha X é ruim.** A polaridade foi definida como positiva devido à palavra "ruim" estar inclusa na base de palavras negativas.	Negativa
3. **Adorei visitar o museu X.** A polaridade foi definida como muito positiva devido à palavra "adorei" estar inclusa na base de palavras muito positivas.	Muito positiva
4. **Odeio política baseada no equilíbrio do pão e circo.** A polaridade foi definida como muito negativa devido à palavra "odeio" estar inclusa na base de palavras muito negativas.	Muito negativa
5. **Gostei de morar no Rio de Janeiro.** A polaridade foi definida como positiva devido à palavra "gostei" estar inclusa na base de palavras positivas.	Positiva
6. **Não aprovei o novo sindico do prédio.** A polaridade foi definida como negativa devido à palavra "aprovei", que está sendo negada com a palavra "não", estar inclusa na base de palavras positivas.	Negativa
7. **O novo site da empresa X não ficou bom com as novas atualizações.** A polaridade foi definida como negativa devido à palavra "bom", que está sendo negada com a palavra "não", estar inclusa na base de palavras positivas.	Negativa
8. **Não gostei da cidade A, no entanto, amei a cidade B.** Neste caso existe uma situação que é a negação de uma palavra positiva (gostei) e outra que se caracteriza como muito positiva devido a palavra amei. Por isso a opinião foi polarizada como positiva.	Positiva
9. **Seria muito bom se todas as pessoas tivessem onde morar.** Embora a frase tenha uma característica um pouco negativa, o protótipo leva em conta que a palavra "seria" neutraliza a palavra muito bom e como não existe outra palavra cuja polaridade não seja neutra, a opinião é caracterizada como neutra.	Neutra
10. **Prefiro o computador X que o computador Y.** Nenhuma das palavras foi inclusa nas bases de palavras e por isso a opinião foi considerada neutra.	Neutra
11. **Se eu gosto de futebol? Com certeza não. Odeio futebol.** A primeira frase é categorizada como neutra por se tratar de uma interrogação. O complemento "Com certeza não.", não muda o sentido da palavra "odeio" pois o ponto seguido faz com que a negação não influencie as demais palavras. A palavra "odeio" está inclusa nas palavras muito negativas e por causa dela a polaridade foi classificada como muito negativa.	Muito negativa
12. **Se eu gosto de futebol? Com certeza não odeio o futebol.** A primeira frase é descartada na análise e a segunda é considerada positiva devido a negação da palavra "odeio".	Positiva
13. **A cidade de Brasília tem um clima muito agradável.** A palavra "agradável" foi inclusa na base de palavras positivas e por isso polarizada como positiva.	Positiva

14. Não é verdade que eu não me agradei do Camaro SS 2013. Nesta opinião ocorre a situação de negação sobre negação e o texto é categorizado em positivo de acordo com a polaridade da palavra “agradei”.	Positiva

Conforme visualizado na Tabela 4, da linha 6 à linha 8 foram iniciados os testes utilizando negações de palavras. Nestes testes as opiniões foram classificadas numa polaridade inversa à classificação definida para as palavras presentes no texto.

Na sequência, linha 9 e 10, foram realizados testes com a intenção de obter uma classificação neutra. Depois foram realizados testes para verificar se o protótipo está diferenciando opiniões de perguntas e também se ele está interpretando corretamente as demais pontuações utilizadas no texto.

Na última linha é realizado um teste sobre um texto que contêm uma negação sobre outra negação. Nestes casos foi verificado que o sistema funcionou de fato anulando ambas as negações. Também foram realizados testes utilizando as opiniões obtidas por meio das *APIs* das redes sociais escolhidas. Estas opiniões foram obtidas através de pesquisas aleatórias realizadas no protótipo. Os termos utilizados nas pesquisas foram: Galaxy S4, Renault Logan, Santander, Restaurante Street Grill, Dia dos namorados, Itaipava e Porto de Viamão. Os termos foram destacados em negrito no texto analisado. Na Tabela 5 são apresentadas algumas das opiniões utilizadas nos testes e a explicação de cada resultado.

Tabela 5: Exemplos de opiniões analisadas que foram obtidas nas redes sociais

a.		**Galaxi s3** e s4 são mesmo um espectáculo sem dúvida gosto e Parabéns à marca samsung Positivo - Luciano Costa Positivo por conter as palavras "gosto" e "parabéns".
b.		Pra variar a Sony lançou com pressa um console quadrado e feio, pra depois fazer a versão slim com mais calma. Esse tem o design de um **Renault Logan**, ou seja, nenhum... a versão slim vem no estilo Lamborghini. Negativo - André Mathias Negativo por conter a palavra "feio".
c.		Olha meu desempenho no Desafio **Santander**! Os 30 mil já são praticamente meus, mas ainda assim vou dar uma chance. Quem ousa fazer melhor?! Hein?! Hein?! Positivo - Welton Kelly Andrade O protótipo definiu a opinião como positiva após remover os caracteres especiais da palavra "melhor?!" e trata-la como positiva.
d.		Na contramão da felicidade a Cidade conta agora as horas para o triste fim da história do **Santander** Banespa entre nós !!! Negativo - Rudnei da Silveira O protótipo definiu a opinião como negativa por negar a palavra positiva "felicidade" com a palavra contramão e por conter a palavra negativa "triste".
e.		Hoje é o dia dos namorados, momento de compartilhar alegria e muito amor! Pensando nisso, o **Restaurante Street Grill** preparou pratos especiais para você oferecer a quem vc ama, venha comemorar esse dia conosco. O restaurante está localizado na Qd: 104 Sul ao lado do Bar Chopp Brahma. Muito Positivo - Chopp Brahma Foi classificado em muito positivo por conter as palavras "alegria" e a junção das palavras "muito + amor".
f.		E dai se eu vou passar o **Dia dos namorados** sem namorado?! Eu também não passo o dia do índio com o índio, nem o dia da árvore com uma árvore, muito menos o dia de finados com um defunto! KKKKK ;) Positivo - Antonio Carlos de Souza A opinião foi considerada como positiva por conter o símbolo ";)" e por conter os caracteres "kkkkk", na execução do sistema estes caracteres são convertidos para apenas 3 Ks ("kkk") e tratados como positivos.
g.		Como pode a AMBEV não ter controle de estoque, todo dia acontece falha de sistema, falha de comunicação interna, eu fico transtornada com isso, sempre deixando os clientes na mão. E a **ITAIPAVA** ganhou o mercado jogou o preço lá em cima, não faz mais nenhuma promoção, vive com os produtos in falta sem previsão pra chegar, não tem um material de ponto de venda. É nossas horas que tem que aparecer uma concorrente para engolir essas empresas que não tem capacidade técnica de atender suas demandas, e ficam colocando em risco os negócios dos seus clientes. Negativo - Jaqueline Cruz Opinião considerada negativa por conter a negação da palavra "controle"; por conter a palavra "falha".
h.		Não basta ter o carro do picolé, a kombi dos produtos de limpeza, o caminhão da liquigás, agora tem a kombi do peixe ¬¬ Fazem de tudo para você não dormir aqui em **Viamão** ... Neutro - Coisas que Viamão Fala Opinião neutra por não conter nenhuma palavra caracterizada como positiva ou negativa.

Para validar a classificação foi realizada uma soma entre as palavras positivas e muito positivas e também uma soma entre as palavras negativas e muito negativas e depois os valores de cada soma foram subtraídos a fim de identificar qual a polaridade prevaleceu.

Na validação também foram levadas em consideração as negações existentes antes de palavras com polaridade.

As *APIs* utilizadas não limitam os resultados a textos que contenham termos idênticos à pesquisa. Por exemplo, no item **a** os termos utilizados foram "Galaxy S4" mas, os termos encontrados no texto não seguem a mesma ordem dos termos da pesquisa. O mesmo acontece no item **h** mas encontra apenas um dos termos.

No geral, acredita-se que uma atualização nas bases de palavras do protótipo é um passo que deverá contribuir muito no protótipo já que 51% dos erros estão relacionados à palavras que não foram previstas nas bases de palavras.

4 CONSIDERAÇÕES FINAIS

No decorrer deste trabalho foram realizados estudos sobre a descoberta de conhecimento em bases de dados estruturadas e não estruturadas. Estes estudos abordaram as técnicas e os algoritmos de mineração de dados e as tarefas relacionadas com a mineração de textos em especial a tarefa de classificação. A partir dos conceitos estudados foi elaborado o referencial teórico do qual foram providos os fundamentos para o desenvolvimento do protótipo de classificação automática de opiniões.

O tema escolhido para este trabalho corresponde a uma área muito importante e promissora da mineração de dados. O tema aborda uma ferramenta que é de interesse de inúmeras empresas independente da área de atuação. O objetivo do trabalho foi desenvolver um protótipo que efetuasse a classificação de um conjunto de opiniões obtidas das principais redes de relacionamento e interação disponíveis na *web*. Para tanto, optou-se por recuperar as mensagens postadas no *Facebook*, no *Twitter* e no *Google Plus*.

Constatou-se que as mídias sociais escolhidas consistem em meios de relacionamento e interação amplamente utilizados. Estas redes sociais recebem diariamente opiniões e críticas sobre vários assuntos o que é muito válido para as empresas que desejam acompanhar a aprovação de suas decisões, marcas e produtos. Na intensão de melhorar este acompanhamento o trabalho em questão tem como propósito possibilitar que um usuário realize pesquisas sobre um termo qualquer e que lhe sejam retornados, caso existam, opiniões relacionadas com a pesquisa classificadas em cinco tipos de polaridade: positivo, muito positivo, negativo, muito negativo e neutro.

Para atingir o objetivo do trabalho foi desenvolvido um protótipo de classificação automática de opiniões para analisar e classificar as opiniões consultadas. Ao final do desenvolvimento foi possível obter um protótipo de classificação funcional. O protótipo incluiu além das funcionalidades propostas, um gráfico no qual as opiniões são agrupadas nas cinco classes de polaridade determinadas.

Todo o processo de classificação é realizado por uma biblioteca de mineração e desenvolve-la foi uma iniciativa muito válida, pois possibilitou a utilização do conhecimento adquirido nos estudos realizados sobre mineração de dados e suas subáreas. Embora 19% dos testes realizados não tenham obtido êxito, apenas 2% corresponderam a equívocos dos

sistemas e 12% das falhas poderiam ser corrigidas atualizando a base de dados, uma vez que correspondem à gírias, outros idiomas, palavras portuguesas não adicionadas à base de palavras, palavras compostas e expressões idiomáticas mal interpretadas.

O protótipo utiliza 6 tipos de palavras como base para realizar as classificações: pouco positiva, positiva, muito positiva, pouco negativa, negativa e muito negativa. Isto possibilita que a classificação seja realizada de forma mais relevante. Estas palavras poderiam também ser organizadas em mais tipos. Assim, outro trabalho futuro poderia envolver uma forma de pontuar estas palavras com um valor numérico que representasse a polaridade das mesmas além de encontrar uma forma de atualizar a base de palavras automaticamente.

O protótipo realiza as consultas e recebe as informações em um único retorno de cada *API*. Este fluxo é mais interessante em situações nas quais pretende-se realizar uma análise imediata sobre algum assunto. Porém, existe outro modo de realizar estas consultas utilizando a tecnologia de *streaming* (fluxo de distribuição de dados através de pacotes) para carregamento das informações. Com a utilização de *streaming* as pesquisas se manteriam ativas classificando novas postagens. Assim, utilizar *APIs* que ofereçam suporte a streaming poderia ser realizado em outro trabalho futuro.

5 REFERÊNCIAS BIBLIOGRÁFICAS

ABERNETHY, Michael. **Mineração de dados com *WEKA***: Introdução e regressão. DeveloperWorks, 2010, Disponível em: <http://www.ibm.com/developerworks/br/opensource/library/os-*Weka*1>. Acesso: 08 de mai. de 2013.

APOLINARIO, A. L.; SILVA, A. L. **Mantendo Listas de Controle de Acesso de Proxy Utilizando Classificador de Conteúdo Naive Bayes**. 58 p. Monografia (Bacharelado em Informática), Universidade Católica do Salvador, Salvador, 2007.

BARION, E. C. N.; LAGO, D. Mineração de Textos. **Revista de Ciências Exatas e Tecnologia**, Valinhos, SP, p. 123-140, 08 dez. 2008.

BARREIRA, R. G.; SOUZA, J. G. Proposta de uma ferramenta de notificação de conteúdo do *Twitter* baseado em técnicas de Extração Automática de Tópicos e Clustering. In: ENCONTRO DE COMPUTAÇÃO E INFORMÁTICA DO TOCANTINS, 13., 2011, Palmas. **Anais...** Palmas: CEULP/ULBRA, 2011. p. 219-227.

BETTIO, R. W. **Interrelação das Técnicas Term Extraction e Query Expansion Aplicadas na Recuperação de Documentos Textuais**. 99 p. Tese (Doutorado em Engenharia e Gestão do Conhecimento) – Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento, Universidade Federal de Santa Catarina, Florianópolis.

BOUCKAERT, R.; *et al*. ***WEKA* Manual**. 327 p. University of Waikato, Hamilton, New Zealand, 2013.

BRAGA, C. V. **Rede Neural e Regressão Linear:** Comparativo entre as Técnicas Aplicadas a um Caso Prático na Receita Federal**.** 96 p. Dissertação (Mestrado em Administração), Rio de Janeiro, 2010

BRUSSO, M. J. **Access Miner:** Uma proposta para a Extração de Regras de Associação Aplicada à Mineração do Uso da Web. 96 p. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2000.

CAMILO, C. O.; SILVA, J. C. **Mineração de Dados:** Conceitos, Tarefas, Métodos e Ferramentas. Instituto de Informática, Universidade Federal do Goiás, Goiânia, 2009, p. 29.

CORREA, A. C. G. **Recuperação de Documentos baseada em Informação Semântica no Ambiente AMMO**. 92 p. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Ciência da Computação, Departamento de Computação da Universidade Federal de São Carlos, São Carlos, Brasil, 2003.

ECLIPSE. **Jetty**. JETTY, 2013. Disponível em: <http://www.eclipse.org/jetty>. Acesso: 28 de mai. de 2013.

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From Data Mining to Knowledge Discovery in Databases. **Artificial Intelligence Magazine**, v. 17, n. 3, p. 37-54, 1996.

GOELZER, M. P. **Análise de Técnicas de Mineração de Dados Aplicada na Bioinformática na Descoberta de Drogas**. 121 p. Monografia (Bacharelado em Ciência da Computação), Universidade de Passo Fundo, Passo Fundo, 2007.

GONZÁLEZ, D. P. **Algoritmos de Agrupamiento basados en densidad y Validación de clusters.** 171 p. Tesis (Doctorado en Inteligencia Artificial), Universitat Jaume I, Castelló de la Plana, Espanã, 2010.

HAMERLY, Greg; ELKAN, Charles. Learning the k in k-means. In: conference on neural information processing systems (NIPS**)**, 17., 2003, San Diego. **Anais...** San Diego: University of California, Department of Computer Science and Engineering, 2003. p. 281-288.

HASTIE, T ; R TIBSHIRANI; J FRIEDMAN. **The Elements of Statistical Learning**. 2. ed. Stanford, California, Springer Series in Statistics, 2008. 739 p.

HIGHCHARTS, Solutions. **Highcharts JS.** Disponível em: <http://www.highcharts.com/>. Acesso: 06 de jun. de 2013.

INMON, W. H. **Como construir o *Data Warehouse***. 2. ed. Rio de Janeiro, Campus, 1997. 388 p.

JUDD, C.; NUSAIRAT J.; SHINGLER, V.; LAYKA, V. ***Beginning Groovy, Grails and Griffon***. 1. ed. *Nova York, United States of America, Apress,* 2012. 364 p.

KLEIN, D. ***Grails****: A Quick-Start guide. United States of America, Pragmatic Programmers*, 2009. 212 p.

LAZZAROTTO, L. L.; OLIVEIRA, A. P.; LAZZAROTTO, J. J. Aspectos Teóricos do Data Mining e Aplicação das Redes Neurais em Previsões de Preços Agropecuários. In: **Congresso da Sociedade Brasileira de Economia e Sociologia Rural**, 44., 2006, Fortaleza. **Anais...** Fortaleza: Universidade Federal do Ceará, 2006. p. 98-115.

LORENA, A. C.; CARVALHO, A. C. P. L. F. Uma introdução às *Support Vector Machines*. **Revista de Informática Teórica e Aplicada**, v. 14, p. 43-67, 2007.

MICROSOFT, Corporation. **Algoritmos de mineração de dados (Analysis Services – Mineração de Dados)**. SQL Server, 2012, Disponível em: <http://technet.microsoft.com/pt-br/library/ms175595>. Acesso: 23 de nov. de 2012.

NAGOA, M. E. **Uma Abordagem Baseada em Redes Neurais Artificiais para a Estimação de Densidade do Solo**. 134 p. Tese (Doutorado em Agronomia) – Área de Concentração em Energia na Agricultura, UNESP, BOCATTU, 2003.

NETBEANS. ***NetBeans IDE****: NetBeans IDE Features.* Disponível em: <https://netbeans.org/features/>. Acesso: 30 de mai. de 2013.

NETO, J. Conhecendo mais do *Framework Grails*. ***iMasters****,* 2010, Disponível em: <http://imasters.com.br/artigo/17545/desenvolvimento/conhecendo-mais-do-framework-grails/>. Acesso: 28 de mai. de 2013.

NEWSON, A.; HOUGHTON, D.; PATTEN, J. **Blogging and Other Social Media**: Exploiting the Technology and Protecting the Enterprise. Gower Publishing Ltd. 2008. 200 p.

PESSOTTI, H. C. **Desenvolvimento de um *Framework* para Classificação de Doenças Pulmonares Difusas apoiado por Técnicas de Recuperação de Imagens Baseada em Conteúdo e Recuperação de Informação Textual.** 41p. Monografia (Bacharelado em Informática Biomédica), Faculdade de Filosofia Ciências e Letras, Faculdade de Medicina de Ribeirão Preto, Universidade de São Paulo, Ribeirão Preto, 2008.

RINO, L. H. M.; PARDO. T. A. S. **A Sumarização Automática de Textos**: Principais Características e Metodologias. In: Congresso da Sociedade Brasileira de Computação, 23., 2003, Campinas, SP. **Anais...** Campinas: III Jornada de Minicursos de Inteligência Artificial. p. 203-245.

RYGIELSKI, C.; *et al*. Data mining techniques for customer relationship management. **Technology in Society**, Vol 24, p. 483-502, out 2002.

SANTOS, C. B.; SCANDELARI, L.; CARVALHO, D. R.; GOMES, J. C.; VAZ, M. S. M. G. Data Mining para Classificação das Funções de Uma Instituição Pública a Partir das Semelhanças Entre Suas Competências. In: SIMPÓSIO DE ENGENHARIA DA PRODUÇÃO: GESTÃO DE DESEMPENHO EM SISTEMAS PRODUTIVOS, 14, 2007, Bauru, SP. **Anais...** Bauru: USP, 2007. p. 1-12.

SBARAI, R. ***Facebook*** **alcança 73 milhões de usuários no Brasil**. Veja**.** Disponível em: <http://veja.abril.com.br/noticia/vida-digital/facebook-alcanca-73-milhoes-de-usuarios-no-brasil>. Acesso: 31 mai de 2013.

SILVA, T. N. **Uma Arquitetura para Descoberta de Conhecimento a Partir de Bases Textuais**. 78p. Monografia (Bacharelado em Tecnologias da Informação e Comunicação), Universidade Federal de Santa Catarina, Araranguá, 2012.

SMITH, C. **How Many People Use the Top Social Media, Apps & Services?**. Digital Marketing Ramblings. Disponível em: <http://expandedramblings.com/index.php/resource-how-many-people-use-the-top-social-media>. Acesso: 30 de mai de 2013.

SOARES, F. A. **Mineração de Textos na Coleta Inteligente de Dados na Web.** 120p. Dissertação (Mestrado em Engenharia Elétrica), Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Programa de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC-Rio, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2008.

SOBRAL, A. P. B. **Previsão de Carga Horária – Uma Nova Abordagem por Árvore de Decisão**. 90p. Tese (Doutorado em Engenharia Elétrica), Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, Rio de Janeiro, 2003.

SPRING**.** **Spring Framework**. Pivotal, 2013. Disponível em: <http://www.springsource.org/spring-framework>. Acesso em: 28 de mai. de 2012.

SULLIVAN, D. The need for Text Mining in business intelligence. **DM Review**, 2000. Disponível em: <http://www.dmreview.com/specialreports/20040210/8100-1.html>. Acesso em: 19 de out. de 2012.

TAN, A. Text Mining**:** Promises And Challenges. In: **South East Asia Regional Computer Confederation**, Westin Stamford Hote, 1999. P. 1-7

TORRIELLI, J. Groovy and Grails: Langage de script basé sur Java appliqué dans un environnement JEE. In: EXPOSÉS INFORMATIQUE & RÉSEAU 2010, 2010, Seine-et-Marne, Paris: UNIVERSITÉ DE MARNE-LA-VALLÉE, 2010. 43 slides: color. Disponível em: <http://www-igm.univ-mlv.fr/~dr/XPOSE2009/*Groovy* _and_*Grails*>. Acesso em: 16 de mai de 2013.

TWITTER. **Sobre o Twitter**. Twitter, Inc. Disponível em: <https://twitter.com/about>. Acesso: 30 de mai de 2013.

VIERA, A.F.G. & VIRGIL, J. Uma revisão dos algoritmos de radicalização em língua portuguesa. *Information Research*, Vol 12, nº 3, april 2007, Paper 315. Disponível em: <http://InformationR.net/ir/12-3/paper315.html>. Acesso em: 19 de out. de 2012.