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

Rafael Gonçalves Barreira

ANÁLISE DE SENTIMENTOS COM RAPIDMINER

Palmas – TO

2013

Rafael Gonçalves Barreira

ANÁLISE DE SENTIMENTOS COM RAPIDMINER

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. Parcilene Fernandes de Brito

Palmas – TO

2013

Rafael Gonçalves Barreira

ANÁLISE DE SENTIMENTOS COM RAPIDMINER

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. Parcilene Fernandes de Brito

Aprovada em xxxxxxx de 2013.

BANCA EXAMINADORA

_____

Prof. M.Sc. Parcilene Fernandes Brito

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Edeilson Milhomem da Silva

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Fabiano Fagundes

Centro Universitário Luterano de Palmas

Palmas – TO

2013

RESUMO

O uso mais frequente da computação para a resolução de problemas que envolvam diferentes áreas de conhecimento tem propiciado o surgimento de novas subáreas na computação. Uma dessas subáreas tem relação com a utilização de conceitos e técnicas computacionais para alcançar uma compreensão coerente dos textos publicados em redes sociais virtuais. A Análise de Sentimentos é uma das áreas da computação que é responsável por lidar com esse tipo de situação. Este trabalho apresentará o conceito e a utilização da Análise de Sentimentos aplicado no Twitter, com o uso do ambiente de desenvolvimento RapidMiner.

PALAVRAS-CHAVE: Análise de Sentimentos, RapidMiner, SVM

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS FONTES

LISTA DE ABREVIATURAS

API – Application Program Interace

HTML – Hiper Text Markup Language

JSON – JavaScript Object Notation

SVM – Support Vector Machine

XML – eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Com o número cada vez maior de conteúdo publicado na internet, dos mais variados tipos e assuntos, há uma necessidade mais imediata em criar mecanismos que possibilitem uma sistematização. Isso porque há vários exemplos de empresas procurando saber onde há interesse sobre seus produtos e/ou serviços, bem como de seus concorrentes; além do interesse de políticos sobre a reação de suas ações; produtoras sobre aceitação de filmes produzidos; etc.

Analisar uma quantidade tão extensa de dados manualmente tornou-se uma tarefa não apenas complexa como também praticamente impossível de ser realizada se for considerado um dado tempo/espaço. Desta forma, foram definidos, na literatura, diversos meios para o processo de coleta e processamento de informações, que possibilitam que análises (por exemplo, identificar e classificar textos) sejam realizadas de forma mais rápida. Assim, com o crescimento exponencial do conteúdo produzido na *web*, aumentou também o interesse por meios que pudessem trabalhar com esse conteúdo de forma a extrair o máximo de informações possível para um dado contexto.

Para identificar e obter conteúdos relevantes, a extração de informação surgiu como uma área de estudo para prover meios para esse objetivo, bem como servir de base para outras áreas. A partir das informações obtidas, é possível extrair algum tipo de conhecimento que possa ter uma aplicação prática. E é nesse sentido que a Aprendizagem de Máquina tem um papel relevante, pois as técnicas embasadas nessa teoria têm como propósito identificar padrões em domínios complexos, e a partir destes possibilitar um meio para utilizar as informações contidas nesse domínio.

As pesquisas sobre identificação e utilização das informações obtidas geraram outras temáticas, algumas delas tendo como foco a busca por um entendimento mais aprofundado do teor do texto, de forma a identificar o sentimento do autor sobre um dado objeto (coisa, pessoa, produto etc.). Nesse cenário, surgiu a Análise de Sentimentos, que busca identificar os valores subjetivos do documento, o que possibilita que mais informações sejam inferidas, e que, por consequência, melhores ações sejam tomadas. Por exemplo, no caso de uma empresa que, ao descobrir que seu produto é comentado negativamente em uma rede social, poderá planejar modificações no produto de forma a melhorar sua aceitação.

Assim, a possibilidade de trabalhar com ferramentas que promovam uma análise do texto (ou o entendimento da opinião de um indivíduo ou grupo) de forma automatizada fez com que estabelecesse rapidamente uma vertente de pesquisa fortemente aplicada e necessária a atual realidade. Uma dessas ferramentas é o *RapidMiner*, um ambiente de *data mining*, que possui um grande acervo de funcionalidades, além do apoio de uma comunidade de usuários que provê informações quanto ao uso da ferramenta, e também contribui com uma constante agregação de novas funcionalidades.

Este trabalho tem como objetivo aplicar a técnica de Análise de Sentimentos em um conjunto de documentos de um domínio através do ambiente de desenvolvimento *RapidMiner*. Para atingir o objetivo proposto, foram estabelecido objetivos específicos, os seguintes: Realizar um estudo sobre os conceitos que embasam a técnica de Análise de Sentimentos; Aplicar as técnicas de Extração da Informação em um conjunto de documentos de um dado domínio; Utilizar o resultado obtido a partir Extração da dos algoritmos de aprendizagem de máquina presentes no *RapidMiner;* Informação e aplicá-lo em um algoritmo de Aprendizagem de Máquina; Apresentar as etapas envolvidas no processo de Análise de Sentimentos em um determinado domínio a partir da utilização dos métodos de classificação e Analisar os resultados obtidos de forma a verificar o desempenho do processo aplicado.

O objetivo desse trabalho foi guiado no intuito de resolver o seguinte questionamento: “Como aplicar os recursos disponíveis no *RapidMiner* para realizar a técnica de Análise de Sentimentos?” No qual há a seguinte hipótese, que norteou este trabalho: Se for possível mapear as características de uma opinião, em um contexto escolhido, para as funcionalidades disponibilizadas no ambiente de desenvolvimento *RapidMiner*, então é possível utilizar essa ferramenta para a implementação da técnica de Análise de Sentimento.

A necessidade de estudo e aplicação desse trabalho surgiu devido à quantidade de conteúdo produzido em redes sociais, portais de notícias etc. que gerou a necessidade de extrair informações desse meio como forma de entender melhor nichos de mercado em potencial, indivíduos e suas relações sociais, com produtos e com a mídia. Esses dados geram a possibilidade de haver diferentes manifestações de opinião, o que torna o ambiente mais interessante de trabalhar, tanto no sentido de complexidade quanto nos resultados que possam surgir. Facilitar meios para coletar essas informações é sempre vantajoso, o que pode resultar em informações relevantes em tempo hábil, no qual possa ser aplicado em uma tomada de decisão. Pois um interesse comercial do projeto serve como um indicador de como a solução proposta pode ser aplicado a um problema real.

A Análise de Sentimentos identifica os valores subjetivos de um documento, e através desses valores permite, por exemplo, que sejam descobertos os interesses do autor do documento sobre determinado assunto. Se for considerado esse assunto como um produto de uma empresa, o conhecimento sobre essa informação pode permitir que a empresa avalie a satisfação do cliente para o produto em questão. Além disso, através de técnicas de Análise de Sentimentos, também é possível acompanhar a evolução desse interesse, ou dos vários interesses, desse produto no decorrer do tempo, permitindo não apenas avaliar a satisfação do cliente, como também os resultados das ações tomadas com essa decisão.

Em cenários assim, a existência de ferramentas que auxiliem e até automatizem parte da tarefa ou também que permitem manipular os dados de uma forma mais ágil, faz com que seu uso possa ter uma grande influência nos resultados finais. Uma das ferramentas que pode ser utilizada nesse contexto é o RapidMiner, pois possui um grande número de funcionalidades, que vai desde a preparação de dados (limpeza de dados, conversão de tipos, etc.) até a visualização dos resultados, contendo ainda diversos algoritmos de Inteligência Artificial, além de possuir uma boa curva de aprendizado.

Este trabalho está dividido nas seguintes etapas: a partir da seção **Erro! Fonte de eferência não encontrada.** são apresentados os detalhes do trabalho, como o objetivo geral. Na seção 2 contém o referencial teórico necessário para o estudo e aplicação do trabalho, com o seguintes temas abordado: Extração de Informação, Aprendizado de Máquina e Análise de Sentimentos. Na seção 3 é apresentada a metodologia, apresentando melhor a ferramenta escolhida e os procedimentos utilizados. Na seção 4 tem-se os resultados deste trabalho, bem como a análise do mesmo. E na seção 5 são apresentadas as considerações finais obtidas ao término do trabalho. E, por fim, as referências bibliográficas utilizadas são apresentadas na seção 6.

2 REFERENCIAL TEÓRICO

Na seção 2.1 serão apresentados alguns conceitos sobre extração de informação, no qual são necessários para o projeto. Aprendizado de Máquina será apresentado na seção 2.2. Na seção 2.3 é apresentada a base teórica sobre Análise de Sentimentos.

2.1. Extração de Informação

Obter informações de uma fonte de dados tem complexidade de acordo com a estrutura dessa mesma fonte, por exemplo, extrair informação de documentos textuais, é mais complexo que extrair de um banco de dados relacional. Isso em relação ao fato de que os dados armazenados em um banco de dados são melhores organizados (tabelas, colunas, metadados, etc.) que textos, o que facilita localizar mais rapidamente os dados necessários. Como o objetivo deste trabalho será a aplicação em um conjunto de documentos textuais, é importante entender como trabalhar com esse tipo de informação, para evitar que informações relevantes sejam descartadas o que pode prejudicar o resultado da Análise de sentimentos. A Extração de Informação é uma área de estudo que pode ser aplicada para esse fim, pois tem como objetivo identificar e extrair informações de eventos ou relacionamentos em textos de linguagem natural, construindo ainda uma representação que organiza essas informações obtidas (GRISHMAN, 1997).

Uma definição semelhante é apresentada por Kushmerick e Thomas (2003, p. 1, tradução nossa): “uma forma superficial de processamento de documentos que envolvem o preenchimento de uma base de dados com valores automaticamente extraídos de documentos”. Esta definição transmite a ideia de que a Extração de Informação é mais voltada para a preparação de dados para serem utilizados em outros procedimentos. Apesar de sua aplicação neste trabalho ser nesse sentido, isso não significa que esse é o objetivo principal da Extração de Informação.

A Extração de Informação resulta em uma fonte de informação que pode ser tanto utilizada diretamente, como, por exemplo, obter o assunto relativo a um documento, quanto por outros sistemas, por exemplo, servir como uma informação pré-processada para *Data Mining* ou mesmo Análise de Sentimentos. Para chegar nesse resultado, as informações extraídas são geralmente organizadas em modelo de *templates*, isto é, estruturas que contêm

campos, que serão preenchidos pelos dados extraídos no processo de Extração de Informação (ÁLVAREZ, 2007, p. 41). A Figura 1 apresenta um exemplo de um *template*.

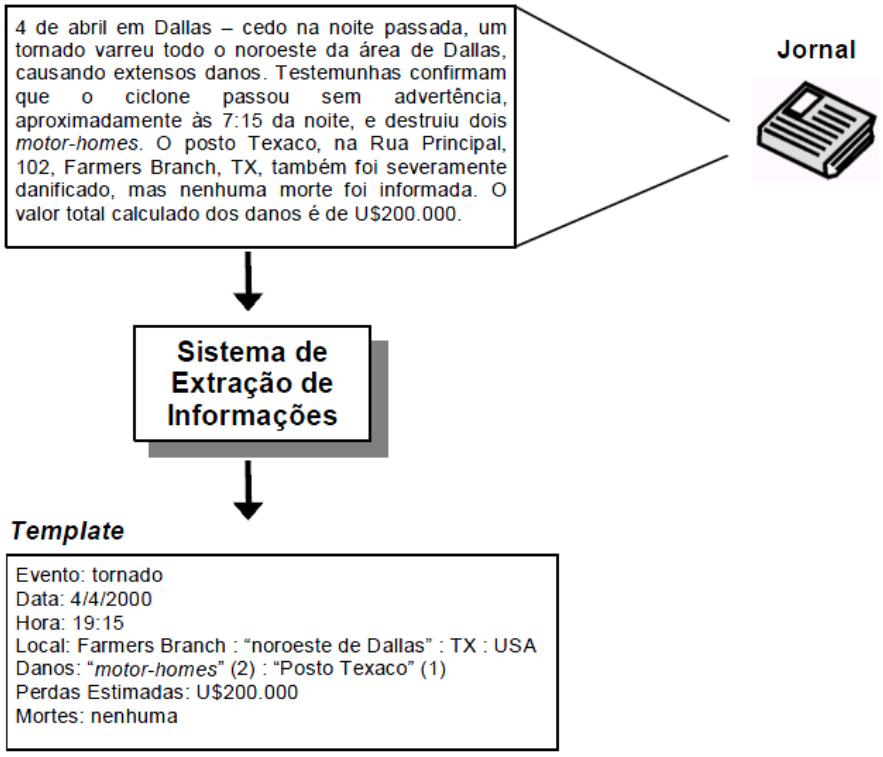

Figura 1: Processo de criação de um template (ZAMBENEDETTI, 2002, p. 26).

O processo de extração de informação é orientado a um objetivo, isto é, a extração é feita com base em características pré-definidas (GAIZAUSKAS e WILKS, 1998, p. 17). No exemplo da Figura 1, o *template* é formado por sete campos, ou seja, há sete objetivos na EI do exemplo.

A Extração de Informação pode ser definida ainda como o processo de preencher uma base de dados estruturada a partir de fontes não estruturadas ou texto puro (GAIZAUSKAS e WILKS, 1998, p. 17). Essas definições que foram apresentadas convergem para o seguinte ponto: obter informações relevantes de um conjunto de dados no qual não há, na maioria das vezes, informações extras, tais como metadados que facilitem a extração.

Um fator considerável durante o processo de Extração de Informação é a organização do texto de origem. Segundo Silva (2004, p. 8), os textos podem ser classificados em: estruturados, semiestruturados e não estruturados.

Um texto estruturado, que é o mais organizado dentre os três, possui sua estrutura predefinida, isto é, a localização de certas informações é facilitada por elementos encontrados no próprio texto. Como exemplo há os documentos HTML e XML. E no caso do HTML, a

estrutura do documento (as *tags*) disponibiliza informações extras, como, por exemplo, uma *tag* de negrito que indica relevância. Na Figura 2 há um exemplo de documento estruturado.

```
<div class="paint" id="curso-desc">
  <div>
    <p>
      " O Bacharel em Sistemas de Informação é capacitado para desenvolver
      trabalhos nas áreas de Informática e Computação, além de realizar
      pesquisas que tenham como foco o processo de inovação tecnológica.
      Concursos públicos em diversas áreas, mestrados e doutorados, gerências
      de centros de informática, consultorias em tecnologia, gerência de
      redes e desenvolvimento de sistemas são algumas das possibilidades
      desse profissional."
    </p>
    <p>
      <strong>Autorizado pela Portaria MEC nº 85, de 29/01/95.</strong>
    </p>
    <p>
      <strong>Reconhecido pela Portaria MEC nº 591 de 21/05/10.</strong>
    </p>
  </div>
</div>
```

Figura 2: Texto Estruturado: Documento HTML.

No exemplo da Figura 2, há o HTML da pagina que descreve o curso de Sistema de informação. Entre as informações extras que podem ser utilizadas, há: as *tags* "<*strong*>" que dão destaque ao conteúdo, o que pode significar uma informação relevante; atributos dos elementos, como o "id" do elemento " " que identifica ao identificar esse elemento pode facilitar o processo descrito na Figura 1.

Os documentos não estruturados não possuem uma organização que facilite a Extração de Informação. Nesse tipo de documento, encontra-se apenas texto em linguagem natural, como na Figura 3.

```
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. Ele também é capaz de
desenvolver sistemas de informação em concordância com as
estruturas organizacionais e com ênfase em informática e
sua aplicação, além de contribuir para o desenvolvimento
científico e tecnológico da área. Sendo assim, projetou-
se um curso de Sistemas de Informação capaz de formar um
profissional com um perfil científico e empreendedor, ou
seja, formar profissionais empreendedores, capazes de
projetar, implantar e gerenciar a infra-estrutura da
tecnologia da informação, envolvendo computadores e
comunicação de dados em sistemas organizacionais.
```

Figura 3: Texto não estruturado: Texto puro.

No texto apresentado na Figura 3 é possível identificar algumas informações relevantes, tais como:

- **Área de atuação:** Informática e computação.
- **Capacidade profissional:** desenvolver sistemas, contribuição científica e tecnológica.
- **Perfil do profissional:** empreendedor e cientifico.

Essas informações são facilmente identificadas quando o texto é lido por um ser humano. Mas essa tarefa não é fácil quando é executada por um computador, porque não há indicadores explícitos do que se trata o texto, diferentemente do exemplo apresentado na Figura 2.

E por fim há os textos semiestruturados, que consistem em uma versão intermediária entre os dois tipos anteriores. Nesse tipo de texto, há uma estrutura que organiza o conteúdo, mas não há uma obrigatoriedade de seguir essa mesma estrutura o tempo todo. As referências bibliográficas são um exemplo de texto semiestruturados, conforme visto na Figura 4

RUSSEL, Stuart, NORVIG, Peter. **Artificial intelligence**: a modern approach. Prentince-Hall, New Jersey. 1995

SILVA, Eduardo Fraga do Amaral e. **Um sistema para extração de informação em referências bibliográficas baseado em aprendizagem de máquina**. 2004. 94 p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Pernambuco, Recife, Pernambuco.

Figura 4: Texto semiestruturado: Referências Bibliográficas

Nos exemplos apresentados na Figura 4, há informações em comum, como os nomes dos autores e os títulos dos trabalhos, que são a primeira e a segunda informação em cada texto, respectivamente. Depois dessas informações, cada referência apresenta dados diferentes, e não há mais uma padronização entre os textos, assim enquanto uma identifica a editora do livro, cidade e ano da publicação, a outra apresenta o ano, o número de página, o tipo de trabalho apresentado e outras informações.

Ao trabalhar com documentos de texto, há muitas informações que não são relevantes, logo podem ser descartadas, ou informações que precisam ser pré-processadas para que tenha uma relevância maior no processo de Extração de Informação. Duas dessas técnicas são: remoção de *stop words* e o processo de *stemming*.

As *stop words* são palavras que não têm valor no contexto de um processamento de texto. Muitas vezes englobam-se nesse grupo: artigos, preposições, verbos de ligação etc., palavras que, na maioria das vezes, possuem apenas valor sintático, que podem ser

descartados pelo computador, pois não carregam nenhum tipo de significado que seja útil no contexto. Na Tabela 1 há um exemplo da remoção de *stop words*.

Tabela 1: Remoção de *stop words*.

Texto original	Texto após a remoção de *stop words*
O Curso de Sistemas de Informação tem como objetivo fornecer uma noção abrangente da área de Informática e Computação, acrescentada de uma formação que permita ao aluno gerir, administrar e, principalmente, criar empresas ou atividades de cunho comércio-empresarial que se utilizam dos conhecimentos advindos dessa área.	Curso Sistemas Informação objetivo fornecer noção abrangente área Informática Computação acrescentada formação permita aluno gerir administrar criar empresas atividades cunho comércio empresarial utilizam conhecimentos advindos dessa área.

O *stemming* refere-se ao processo de reduzir uma palavra a sua origem, seu radical (*stem* em inglês). Esse processo visa agrupar conceitos semelhantes que estão apenas escrito de forma diferente, como conjugação verbal. Na Tabela 2, há um exemplo do processo de *stemming*.

Tabela 2: *Stemming*.

Texto após a remoção de *stop words*	Texto após o *stemming*
Curso Sistemas Informação objetivo fornecer noção abrangente área Informática Computação acrescentada formação permita aluno gerir administrar criar empresas atividades cunho comércio empresarial utilizam conhecimentos advindos dessa área.	Curs sistem info objet fornec noção abrang área info comput acresce form permit alun ger admini cria empres ativi cunh comerc empres utiliz conhec advi essa area

No exemplo da Tabela 2 foi escolhido como texto base o resultado do processo anterior, com o objetivo de evitar processamento em palavras irrelevantes, que a remoção de *stop words* trata. Um dos resultados a ser mencionado no exemplo citado é a geração de

equivalência entre termos, como “empresarial” e “empresa”, que no contexto do texto realmente indicam termos equivalentes.

2.2. Aprendizagem de Máquina

Com o objetivo de obter novas soluções para problemas computacionais, diversos pesquisadores procuraram inspiração em modelos existentes na natureza. Um desses conceitos tem relação com a maneira de aprender dos animais em geral. Aprender é o ato de adquirir conhecimento ou habilidade em algo (HOUAISS, online), o que permite agrupar em uma definição tanto a capacidade de um animal selvagem aprender a caçar, quanto de um ser humano compreender fatos complexos da ciência.

O ato de adquirir conhecimento pode ser compreendido de diversas maneiras: por meio de indução ou por meio de dedução. No método dedutivo o conhecimento é formado a partir de algo geral até chegar as partes especializadas. Nas afirmações: “Todos os homens são mortais” e “Sócrates é um homem” pode-se deduzir, através das informações mais gerais, que “Sócrates é mortal”. Pelo método indutivo, o conhecimento é formado através das partes até chegar a algo genérico. Por exemplo, havendo as seguintes situações: “Todos os sapos são verdes” e “Todas as rã são verdes” induz-se que qualquer anfíbio possa ser verde. Para esse tipo de construção de conhecimento é importante ressaltar que ocorre o risco de haver generalização em excesso, como nos exemplos apresentados.

Ao trabalhar com a questão de aprendizagem na área de computação, muitas teorias têm como base a utilização do raciocínio indutivo. Esse tipo de aprendizado indutivo pode ser assim definido (MITCHEL, 1997, p. 2, tradução nossa): “um programa de computador aprende a partir de uma experiência E em relação a algumas classes de tarefa T e medidas de desempenho P, se esse desempenho na tarefa T, conforme medido por P, melhora a experiência E”. Os três elementos destacados por Mitchel (1997) (experiência, tarefa e medida de desempenho) norteiam a tarefa de “ensinar” algo a uma máquina, o que possibilitaria que um programa de computador pudesse “aprender” com a experiência.

Considerando como exemplo um sistema de reconhecimento facial, a tarefa T desempenhada é encontrar a presença de rosto em vídeos. A medida de desempenho P é obtida através da porcentagem de acertos ao identificar rostos. E a experiência E é uma base de dados de imagens que contenha dados para que o sistema aprenda a identificar o que é e o que não é um rosto.

A definição de aprendizado de máquina refere-se à capacidade de uma máquina de mudar sua estrutura, o seu funcionamento para melhorar seu desempenho, tendo como base

para essas ações as entradas do sistema ou alguma influência externa (NILSSON, 1998, p.1). É necessário apresentar aqui que nem toda forma de entrada pode ser considerada como um meio de aprendizagem, isto é, o aprendizado não está relacionado à saída produzida, e sim ao processo que resultou nessa saída. Por exemplo, em um sistema baseado em regras que receba como entrada o valor inteiro 1 (um) e retorne a palavra "azul", não pode ser considerado que o mesmo aprendeu, pois havia um regra para isso (se a entrada for 1 retornar azul) que não é alterada pelo sistema.

O processo de ensinar, também chamado de treinamento, pode ser assim sistematizado (RUSSEL e NORVIG, 1995, p. 528):

- **aprendizado supervisionado:** método de treinamento no qual o aprendizado é realizado baseado em um conjunto de entradas e saídas corretas. Por exemplo, o reconhecimento de imagens é um caso de aprendizado supervisionado, pois através de um conjunto inicial de imagens, com suas classes já identificadas, é possível identificar as características que separam os grupos, e com essa informação realizar futuras classificações com novas imagens.
- **aprendizado não supervisionado:** o treinamento é efetuado sem o uso de respostas corretas, isto é, aprender utilizando apenas os valores de entrada. Sistemas de recomendação de produtos são exemplos de aprendizado não supervisionado, pois o mesmo recomenda um novo produto a partir de um histórico de vendas, no qual houve a venda do produto recomendado e assim associar o produto a outros.
- **aprendizado por reforço:** nesse tipo de treinamento há um *feedback* sobre o resultado obtido, mas sem indicar qual é a saída desejada. A navegação robótica em um ambiente é um exemplo desse tipo de aprendizado. Ao encontrar um obstáculo, o robô recebe um reforço negativo, tendo então que modificar seu itinerário, ao encontrar um caminho sem obstáculo um reforço positivo é dado, fazendo com que o robô mantenha o curso.

Para este trabalho, serão utilizadas técnicas de aprendizado supervisionado, assunto detalhado a seguir.

2.2.1. Aprendizado supervisionado

Nesse tipo de treinamento, há um supervisor, ou tutor, que guia o processo de aprendizagem indicando quais são as saídas corretas, através de um conjunto de dados rotulados. Com as saídas desejadas e com a saída obtida pelo sistema, o supervisor realiza os ajustes necessários

para que o sistema aprenda aquela situação. Na Figura 5 há uma representação do processo de treinamento Supervisionado.

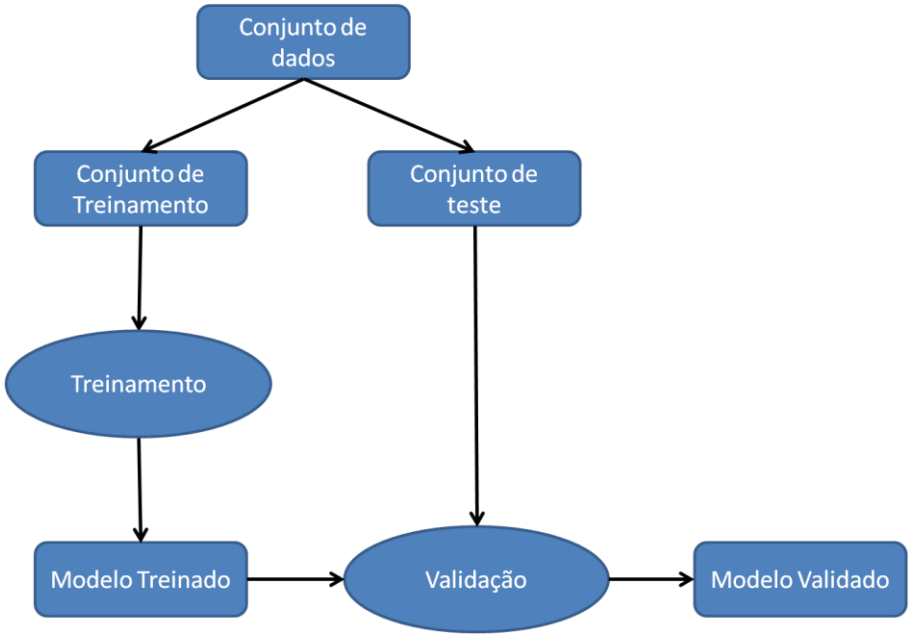

Figura 5: Processo de treinamento supervisionado.

De acordo com a Figura 5, os dados que serão utilizados no treinamento serão divididos em dois grupos: um conjunto de treinamento e um de teste. Ambos os conjuntos são compostos de dados rotulados, isto é, dados que já estão identificados a qual classe deve ser classificada. O conjunto de treinamento é utilizado pelo supervisor para extrair a relação entre entrada e saída que será utilizado para construir o modelo treinado. A próxima etapa, a validação, é realizada pelo supervisor para verificar a eficiência do treinamento, aplicando o modelo gerado ao conjunto de testes. No fim do processo, o modelo está treinado, podendo ser aplicado aos dados do domínio.

Uma abordagem de treinamento supervisionado são as técnicas de classificação no qual o modelo resultante é construir uma função classificadora f, que conforme Sebastiani (2002, p. 3), é definida por:

$$f: D \times C \rightarrow \{0,1\}$$

Onde D representa a coleção de elementos a serem classificados, C representa todas as categorias (ou classes) existentes. A função é a relação entre esses dois conjuntos, sendo que um resultado de valor 1 (um) indica que o documento pertence a uma classe, e o valor 0

(zero) indica que o documento não pertence a classe. A Tabela 3 mostra um exemplo dessa função na tarefa de classificar documentos em área de conhecimento.

Tabela 3: Exemplo de uma função de classificação

	Classe A	Classe B	Classe C
d_1	1	1	0
d_2	1	0	1
d_3	0	1	0
d_4	0	0	1
d_5	1	0	1

No exemplo apresentado pela Tabela 3, para a classe "A" foram classificados os documentos d_1, d_2 e d_5. Há contextos em que um documento pode ser classificado para duas ou mais categorias, como no caso da Tabela 3 que o documento d_1 também foi classificado para a classe B. Essa matriz de decisão (SEBASTIANI, 2002, p. 3) é o resultado da construção da função que tem como objetivo se aproximar da função f, sendo chamada de função classificadora (FELDMAN e SANGER, 2007, p. 66).

Support Vector Machine

Support Vector Machine (SVM) é uma técnica de aprendizado de máquina que é utilizada como uma técnica de classificação. Mais especificamente, o SVM utiliza de um hiperplano para separar um conjunto entre amostras positivas e negativas em um espaço n-dimensional, sendo que este método busca maximar a distância entre as superfícies separadas por esse hiperplano (VAPNIK, 1995 *apud* KINTO, 2011, p.8).

Cada dimensão desse espaço n-dimensional consiste em uma característica dos elementos que serão classificados pelo SVM. Por exemplo, em um caso onde se procura classificar uma se um paciente está com uma doença grave pelos sintomas apresentados, tais como febre, dores e entre outros, esse sintomas são as características analisadas, e cada doença é composta por um conjunto de valores sobre essas características, ou dimensões. Assim o hiperplano que irá dividir esse espaço indicara se essa o elemento apresentado faz parte ou não do conjunto desejado (que no caso é se o paciente está ou não com a doença). Este mesmo hiperplano é apresentado por Lorena e Carvalho (2007, p. 53) por:

$$f(x) = w \cdot x + b \ = 0$$

Onde **w** é o vetor pertencente ao conjunto de entradas, **x** um ponto no hiperplano e **b** é valor da *bias*, isto é, o valor que assumido como resultado não haja nenhuma entrada, influenciando em como a aprendizagem é realizada. Com isso, o hiperplano divide o espaço em dois grupos: com $f(x) > 0$ e $f(x) < 0$. Durante o treinamento do SVM, são encontrados diversos hiperplanos que separam as amostras, como apresentado na Figura 6.

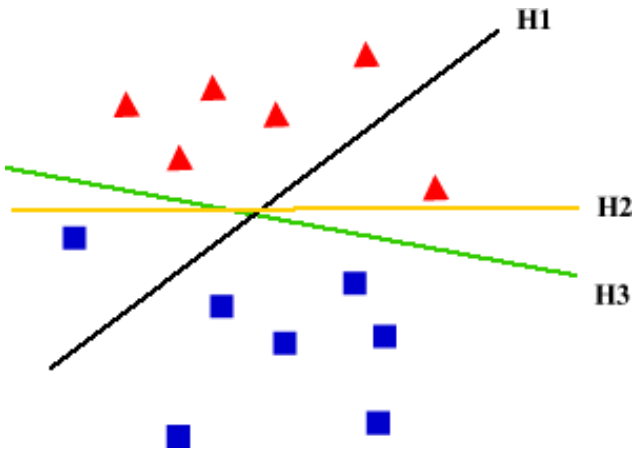

Figura 6: Amostras separadas por hiperplanos.

O hiperplano H_1 classificou incorretamente dois elementos, portanto não é adequado como uma solução para o problema, isto é, a função f'. Já o hiperplano H_2 classificou corretamente todos os elementos, mas o hiperplano H_3 teve um desempenho melhor, pois obteve uma função com uma distancia maior entre os conjuntos de dados. Os elementos que são utilizados como fronteiras dos conjuntos são chamados de *support vectors*, e na Figura 7 mostra os *support vectors* do conjunto da Figura 6.

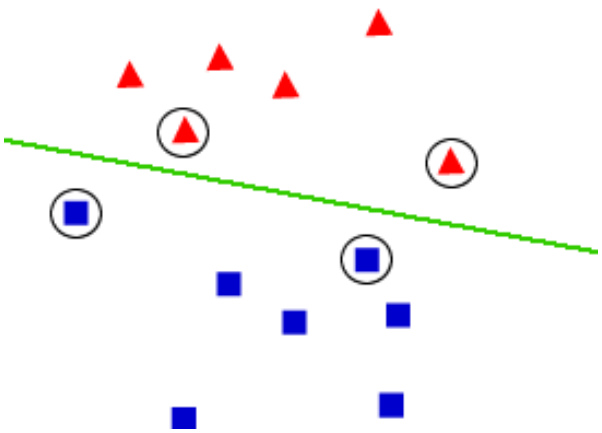

Figura 7: Amostras com os *support vectors* em destaque.

Os *support vectors* são o elementos mais importantes para o SVM, pois são eles que servem como base para a função classificadora. Elementos mais distantes da borda dos grupos estão mais fortemente relacionados à classe que se encontrada em relação a esses elementos que estão mais próximo do limite. A importância dos *support vectors* para a classificação está no fato de que esses elementos, ao serem os limítrofes dos conjuntos que se enquadram, fornece um meio de classificação para uma classe, que considere também a não-classificação de um elemento em outras classes.

Uma das tarefas no qual o SVM tem bons resultados é na classificação de texto e isso acontece devido às seguintes características (JOACHIM, 2007, p. 3):

- **Espaço de entrada de alta dimensão:** a tarefa de classificação é treinada com base nas características dos elementos a serem classificados. Havendo um número maior de características, geralmente aumenta o tempo gasto para o treinamento e a aplicação do classificador. No funcionamento do SVM o numero de características (dimensões) não tem um impacto muito grande na construção do hiperplano, o que possibilita usar um grande numero de características para classificação.
- **Poucas características irrelevantes:** na classificação de texto, há poucas características que são consideradas irrelevantes, isso porque os elementos presentes nos texto, seja letra, palavra ou frase, são geralmente usados como características, e por ser em um domínio de documentos de texto, uma grande parte dessas características são importantes. No SVM não há um limite para a quantidade de características, bem como numero de *support vectors*, o que não há a necessidade de ignorar características que são relevantes no domínio.
- **Vetores dos documentos são esparsos:** essa característica refere-se à representação vetorial de um documento, no qual um documento gera um vetor com muitos elementos e com poucas características que não são nulas, sendo que essa situação pode gerar erros no treinamento. Kivinen et al. (1995) descreve experimentos sobre algoritmos aditivos no qual conclui que esses algoritmos estão aptos para solucionar problemas tanto com documentos densos (possuem grande partes de todas as características existentes no domínio) quanto documentos esparsos (possui algumas das características existentes no domínio), e conforme Joachim (2007, p. 3) o funcionamento desses algoritmos tem semelhanças com os SVM. Isso significa que não há problemas de um treinamento excessivo com as SVM.

- **Maioria dos problemas é linearmente separável:** isso significa que a maioria dos problemas de classificação de texto tem duas classes, ou seja, classificam os documentos em dois grupos. E como apresentado anteriormente, o SVM é um classificador linear que é capaz de lidar com dados compostos por diversas características.

A Análise de Sentimentos, temática desse trabalho, pode ser trabalhada através de métodos de aprendizado supervisionado, com algoritmos SVM. Por exemplo, classificar um documento com conteúdo positivo ou negativo a um determinado assunto já é, de certa forma, um meio de verificar a opinião sobre uma determinada entidade.

2.3. Análise de Sentimentos

Segundo o dicionário Houaiss (online), a etimologia da palavra 'sentimento' expressa a "faculdade de receber as impressões, sensação, conhecimento, fato de saber qualquer coisa; opinião". De certa forma, esse significado está relacionado com as indicações da palavra 'sentimento' apresentadas por Pang (2006), que são divididas em dois grupos: o primeiro relaciona a palavra a uma atitude, pensamento ou julgamento advindo de um sentimento, ou seja, uma predileção, a uma visão específica ou a uma determinada noção, que seria uma espécie de parecer; e o segundo grupo, que pode indicar uma emoção, um sentimento refinado (sensibilidade delicada muitas vezes expressa na literatura), idealismo emocional, um sentimento romântico ou nostálgico beirando ao sentimentalismo.

Para Pang (2006), o significado da palavra 'sentimento' apresentado no primeiro grupo (atitude, julgamento e visão) foi o ponto que ele usou para concentrar seus trabalhos no que tange à análise de sentimentos. O trabalho em questão tem como base o sentido de 'sentimento' relacionado à 'opinião', assim, buscar-se-á uma análise de textos que tem um caráter opinativo e não meramente expressões acerca de fatos. Por exemplo, a seguinte sentença "Palmas é capital do Tocantins" expressa um fato, não um sentimento. Já na sentença "Palmas é a melhor capital do Brasil", tem-se um opinião acerca da cidade de Palmas, valorada de forma positiva. Acrescenta-se, ainda que a opinião refere-se à subjetividade, isto é, a opinião tem como base os valores pessoais de um indivíduo, os seus sentimentos e emoções. A existência dessa característica permite classificar documentos textuais em dois grupos (LIU, 2010, p. 1):

- **Fatos:** expressão objetiva sobre uma entidade, isto é mencionando apenas um fato sem considerar nenhum valor pessoal de quem emite a opinião. Exemplo: “Houve eleições municipais neste ano”.
- **Opiniões:** expressão subjetiva sobre uma entidade, isto é, a expressão tem como base valores pessoas de quem emite a opinião. Exemplo: “Não gostei dos resultados das eleições em algumas cidades”.

Na Análise de Sentimentos, para os documentos que se enquadram no segundo grupo principalmente, há alguns elementos que devem ser identificados, sendo que muitos desses elementos podem ser compostos por subelementos, ou seja, uma estrutura hierárquica de componentes. Liu (2011a, pg. 461-463) apresenta esses e outros componentes a seguir:

- **entidade:** uma entidade *e* representa o objeto no qual uma opinião pode ser referenciada, isto é, uma pessoa, uma empresa, um produto ou um serviço. É definida como *e: (T, W),* onde *T* é uma coleção de elementos com compõe essa entidade, organizados de forma hierárquica. *W* é uma coleção de atributos da entidade *e*. Sendo que cada componente pode ser visto como uma entidade, isto é, possuir um conjunto de subcomponentes e atributos.
- **aspecto:** o aspecto é o que caracteriza uma entidade, isto é, todos os seus componentes e atributos.
- **nome e expressão do aspecto:** o nome do aspecto é o termo que o designa, enquanto a expressão é a palavra ou frase usada no texto para se referir ao aspecto. Por exemplo, dada a opinião “O HD desse computador permite-me salvar tudo que preciso”, o aspecto mencionando é a capacidade de armazenamento do computador, sendo que a expressão utilizada foi o “HD”, o componente responsável pelo armazenamento.
- **nome e expressão da entidade:** o nome da entidade é o nome da entidade no qual a opinião se refere, e a expressão é a palavra ou frase utilizada no texto. Por exemplo, na opinião “Os Países Baixos são os melhores países para se viver”, o termo Países Baixos é tanto o nome quanto a expressão utilizada pela entidade.
- **titular/autor da opinião:** o titular da opinião é quem emite a opinião.

Para um melhor entendimento desses componentes, na Figura 8 é apresentado um exemplo de entidade, com seus elementos e características.

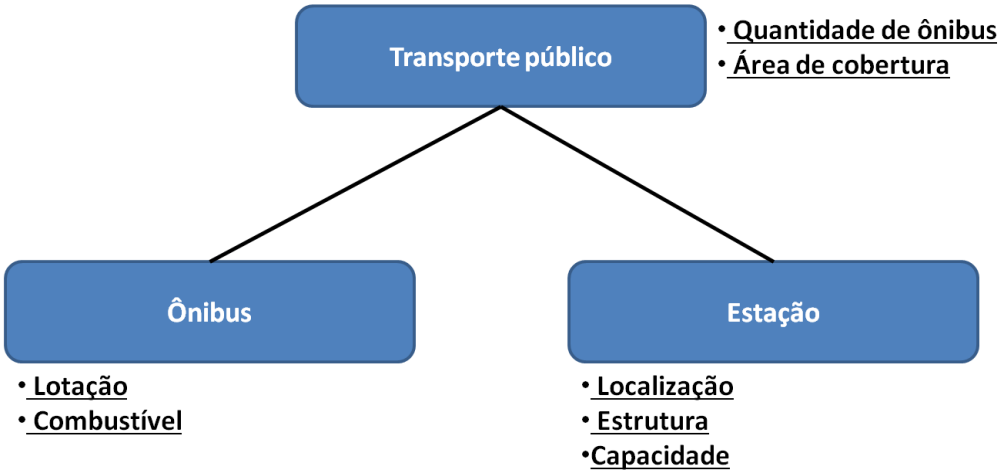

Figura 8: Exemplo de entidade.

Na Figura 8, a entidade exemplificada é o serviço de transporte público, tendo como atributos a quantidade de ônibus e a área de cobertura. O transporte público é composto por ônibus e estações, tendo cada um suas características próprias. Essa relação hierárquica possibilita, por exemplo, que uma opinião sobre os ônibus possa ser relevante para formar uma opinião sobre o serviço de transporte público. Além do mais, o caminho inverso (uma opinião sobre o transporte público) também é preponderante para a formação da opinião sobre as estações de ônibus, isto é, uma opinião de um nível mais alto pode agregar ou subtrair valor nas opiniões de suas entidades componentes.

A opinião é definida por Liu (2011b, p.5) como sendo um sentimento, emoção e/ou ponto de vista sobre uma entidade ou uma característica dessa entidade dada por um emissor, representando pela quíntupla:

$$(e_j, f_{jk}, so_{ijkl}, h_i, t_l)$$

Onde:

- $\mathbf{e_j}$: Entidade *j* que a opinião é direcionada.
- $\mathbf{f_{jk}}$: Característica *k* da entidade e_j no qual é o foco da opinião.
- $\mathbf{so_{ijkl}}$: valor (ou orientação) sentimental *i* sobre a característica f_{jk} da entidade e_j.
- $\mathbf{h_i}$: Emissor *i* da opinião.
- $\mathbf{t_l}$: Momento *l* em que a opinião foi expressada

Para exemplificar, tem-se uma opinião emitida por um cliente em um determinado *site* de *e-commerce* de produtos eletrônicos: "**Eu** recebi o meu ***Tablet Pandora*** esta manhã e ele é fantástico. O **peso**, o **tamanho** e a **qualidade da tela** são ótimos. Fazer pesquisas na internet

no *Tablet* é um processo rápido e posso organizar mais facilmente os artigos para meu doutorado".

No exemplo anterior a entidade e_j identificada é o *Tablet Pandora*, sendo que as características f_{jk} tratadas são o *peso*, tamanho e a qualidade da tela. A opinião é favorável, pois as características citadas foram qualificadas como "ótimas", além disso, é mencionada a facilidade de navegar na internet com o *tablet*, de organizar os trabalhos, além do autor da mensagem adjetivar a entidade como "fantástica". Nessa situação é dito que o valor sentimental so_{ijkl} é positivo. O emissor h_i da opinião é o próprio autor do comentário no site. O momento t_l da opinião é registrado na data da postagem do comentário e isso é relevante para o entendimento do intervalo de tempo na qual aquela qualificação positiva é válida.

A Análise de Sentimentos é uma área da computação que tem como o objetivo estudar esses tipos de informações presentes em textos (PAN, 2012, p. 12). Sendo que a quíntupla apresentada em Liu (2011) geralmente é utilizada como estrutura base desse estudo. No exemplo do comentário do *Tablet Pandora*, as expressões: "é fantástico", "são ótimos" e "pesquisar na internet é um processo rápido" são os indicadores do valor sentimental. Essas expressões muitas vezes contêm certas palavras-chaves (adjetivos em sua maior parte), como: "fantástico", "ótimo" e "horrível", que indicam a orientação sentimental da opinião.

Há também situações que o valor sentimental é expresso a partir de uma relação comparativa. Por exemplo: "O iPhone é melhor do que o Nokia 2300". Nesse caso, o valor sentimental é positivo para a entidade iPhone, e esse valor foi compreendido a partir da comparação com outro produto. Esse tipo de opinião, chamada de opinião comparativa não faz parte do foco desse trabalho e, portanto não será detalhada.

Ainda em relação à identificação do valor sentimental, há opiniões em que esse valor é neutro, não expressando nem uma orientação positiva e nem negativa, como em: "Fiquei surpreso com essa nova rede social". O termo "surpreso" não indica se o emissor sente algo positivo ou negativo em relação à entidade "rede social". Mesmo sem esse valor sentimental, ainda se trata de uma opinião, pois a mesma é formada com base em valores subjetivos.

2.3.1. Classificação de sentimentos

Uma das subáreas da Análise de Sentimentos é a classificação de sentimento, que trata de classificar documentos com base no valor sentimental do mesmo. Nesse tipo de processo geralmente há um contexto com apenas duas classes: uma para documentos com opiniões positivas e outra para negativas.

Para a classificação, deve ser elaborado um modelo que consiga identificar o que seja valor sentimental e qual tipo de valor ele possui. Para isso, técnicas de aprendizado de máquina são aplicadas, conforme apresentado em trabalhos realizados por Pang, Lee e Vaithyanathan (2002).

Utilizando técnicas de aprendizado supervisionado, o classificador irá aprender como identificar valores sentimentais através de um treinamento com exemplos. Sendo que, para treinar um classificador, é necessário rotular os dados e identificar quais características que serão usadas para essa tarefa.

Algumas das técnicas de seleção dessas características utilizadas na análise de sentimentos são as seguintes (LIU, 2006, pg. 470-471):

- **Frequência do termo:** nesse formato o documento é representado por uma coleção de termos *ou n-grams* com as respectivas frequências e/ou pesos.
- ***Part of speech* (parte do discurso):** realiza uma análise morfológica das palavras, identificando as respectivas funções gramaticais, podendo trabalhar tanto com um termo isolado quanto com um conjunto de termos.
- **Palavra-opinião e frase-opinião:** nesse método, os documentos são representados apenas pela palavra ou frase que transmite um valor sentimental. “bom”, “ruim”, “excelente” são exemplos de palavras-opinião.
- **Regras de opinião:** expressões mais complexas que as palavras-opinião e frases-opinião, que também transmite um valor subjetivo. Nos exemplos: “O futebol do fim de semana **diminuiu** meu **stress**” e “esse carro **gasta muito** combustível”, os termos destacado atribuem um valor sentimental positivo no primeiro caso e um valor negativo no segundo. No primeiro caso, os termos transmitem, quando sozinhos, uma opinião negativa (por exemplo, ‘stress’ é negativo), mas quando acrescenta-se a expressão “diminuir” antes da palavra “stress”, isso faz com que a opinião como um todo tenha um valor sentimental positivo.
- **Negações:** As negações (não, nenhum, nada, etc.) podem alterar o sentido de uma opinião (o valor sentimental), o que torna necessária haver um cuidado a parte com essa característica. Por exemplo, na frase “Eu gosto de açaí” há uma orientação positiva por causa da palavra “gosto”, e já na frase “Eu não gosto de café”, apesar de haver o mesmo termo, a frase tem um valor negativo em relação à entidade.
- **Dependência sintática:** Representar as relações entra a expressões que possuem dependência sintática, como por exemplo: “foi a imagem mais bonita **já** retratada pela

câmera” onde o valor sentimental e a características estão em uma oração e a entidade em outras, ligadas pela conjunção “já”.

Com as características selecionadas, parte-se para a etapa de realizar a classificação de fato. Uma variação da representação por frequência que é bastante utilizada é a *bag-of-word*, que consiste na coleção de termos e um valor que indique se o termo existe ou não no documento. A Figura 9 mostra a representação de documentos em relação aos termos: “receber”, “tablet”, “ótimo”, “ruim”, “manhã”, “tela”.

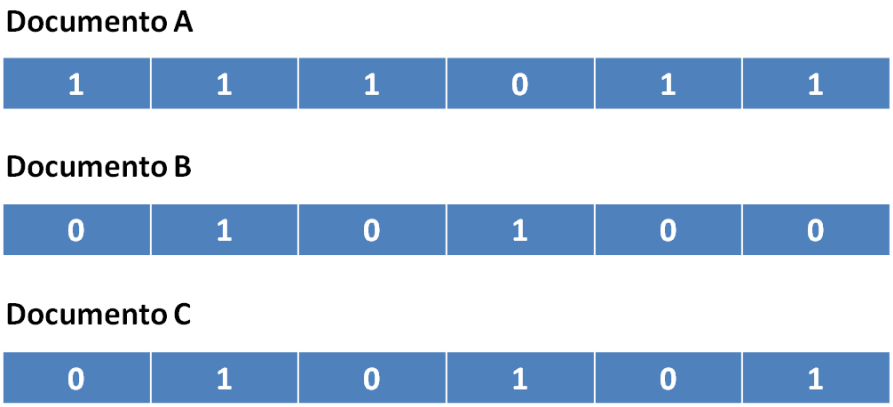

Figura 9: Representação dos documentos como *bag-of-words*.

Na Figura 9, os valores 1 (um) indicam que o termo aparece no documento ao menos uma vez, e 0 (zero) indica que o termo não apareceu no documento. Assim como apresentado na seção 2.2.1, é necessário rotular os dados, que nesse caso são os vetores, para que o supervisor possa identificar as diferenças entre os vetores e elaborar um classificador apropriado.

Considerando que o documento A seja rotulado como uma opinião positiva e os documentos B e C como opinião negativa, então a etapa de treinamento é iniciada.

Uma técnica que pode ser aplicada para realizar a classificação tem relação com os algoritmos SVM’s, pois, como apresentado anteriormente, classificam um conjunto de dados em dois grupos, o que pode ser aplicado à classificação de sentimentos, pois as categorias resultados podem ser identificadas como os valores sentimentais de positivo e negativo.

O hiperplano deverá trabalhar em um espaço 6-dimensional, pois são 6 características utilizadas para representar os documentos. Nessa situação, o SVM, ao ser treinado, deverá encontrar um hiperplano que consiga, no máximo possível, separar os elementos entre as

classes de valor sentimental positivo e negativo. Por exemplo, na Figura 10 há os hiperplanos do SVM gerados para apenas duas das características.

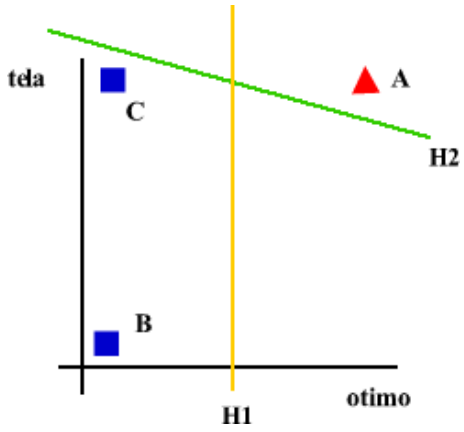

Figura 10: Possíveis hiperplanos em 2 das 6 características

Os elementos B e C da Figura 10 são pertencentes ao grupo de documentos com valor sentimental negativo, não havendo nenhum deles que possua o termo (característica) “ótimo”. Já o elemento A, no que qual possui os termos “tela” e “ótimo” faz parte da classe com valor sentimental positivo. Ambos os hiperplanos gerados (H1 e H2) conseguem separar os elemento em dois conjuntos, no qual o H1 teve um resultado melhor (a máxima distancia entre os *support vectors*) para essas duas características, o que não significa que esse foi o hiperplano gerado, pois há outras características no domínio, e todas devem consideradas.

Essa complexidade não se torna um grande problema para realizar a classificação de sentimentos, pois como foi apresentando na seção 0, os SVM não enfrentam esse problema de entradas com alta-dimensão, que nos exemplos das Figura 9 e Figura 10 consistia apenas de seis termos, que em relação a um domínio de aplicação real é uma quantidade ínfima.

O resultado do processo de treinamento de uma SVM para classificação de sentimento é um modelo que indica se um documento é considerado mais próximo dos documentos positivos ou dos documentos negativos. O que significa que a definição do sentimento vai depender das informações do domínio e de como os elementos do treinamento representa o domínio como o todo, o que é um problema existente nesse tipo de abordagem, a aprendizagem de máquina.

3 MATERIAIS E MÉTODOS

Esta seção apresenta a metodologia utilizada neste trabalho, desde a finalidade e natureza da pesquisa até os procedimentos utilizados no decorrer do trabalho.

Conforme os objetivos apresentados na seção **Erro! Fonte de referência não ncontrada.**, este trabalho é caracterizado como uma pesquisa aplicada, e por pesquisa aplicada entende-se a aplicação de conhecimentos existentes para resolver problemas reais, que neste caso consiste em aplicar as técnicas presentes no ambiente de desenvolvimento *RapidMiner* para realizar a Análise de Sentimentos em textos extraídos do Twitter.

Esse trabalho consiste em uma pesquisa de natureza quali-quantitativa. Isso significa que os resultados obtidos nos procedimentos realizados podem ser medidos através de indicadores quantificáveis, tais como taxas de erro, desempenho dos algoritmos etc., os quantificadores para este trabalho serão apresentados mais adiante. Essas informações possibilitam verificar se o objetivo do trabalho foi concluído, ou está próximo da conclusão, bem como serve como base para validar, ou não, a hipóteses proposta. E por ter sido um projeto que trabalha com análise de valores subjetivos, nem todos os resultados obtidos foram mensurados através de métodos estatísticos, sendo então analisados pelos pesquisadores.

Os materiais utilizados nesse trabalho são apresentados na seção 3.1 e todos os procedimentos utilizados serão apresentados na seção 3.2.

3.1. Materiais

Para a produção bibliográfica foram utilizados: teses, dissertações e artigos científicos. Também foram utilizados como fonte de pesquisa, de modo secundário e em casos específicos, os sites de conteúdo técnico, como no caso da documentação do RapidMiner, tendo como objetivo auxiliar na compreensão de certos conceitos envolvidos.

Para o desenvolvimento deste trabalho foi necessário o uso dos seguintes recursos: o ambiente RapidMiner (seus principais recursos são apresentados na seção 3.1.1); a ferramenta de desenvolvimento Eclipse, apresentada na seção 3.1.2; e a biblioteca de código Twitter4j, apresentada na seção Twitter4j.

3.1.1. RapidMiner

O RapidMiner[1] consiste em um ambiente de desenvolvimento *open-source* que fornece uma interface visual para realização de processos de *data mining*, *text mining*, aprendizado de máquina etc. Esse ambiente disponibiliza um grande conjunto de funcionalidades, chamadas de operadores, que lhe dá certa independência em relação a outros projetos, e ainda conta com um conjunto de extensões que fornece uma maior amplitude no que tange a contextos de utilização (por exemplo, para processamento web, para textos).

Entre as funcionalidades disponibilizadas, podem-se citar os seguintes grupos:

- **Importação e exportação de dados:** funcionalidades que permitem a leitura e gravação em diversos formatos de arquivo de texto (XML, CSV etc.), além de poder acessar serviços de banco de dados.
- **Transformação de dados:** funcionalidades para preparação dos dados nas outras etapas, isso inclui: conversão de tipos, normalização de dados, filtragem, ordenação, entre outras.
- **Classificação:** inclui diversas funcionalidades preparadas para realizar classificação de dados, entre as técnicas disponibilizadas, tem-se: Redes Neurais, SVM, Redes Bayesianas.
- ***Clustering*:** inclui funcionalidades para realização de *clustering*, no qual tem-se os algoritmos de k-means, DBScan etc.
- **Processamento de texto:** nesse grupo há funcionalidades voltadas para trabalhar com documentos de texto, tais como: filtragem (*stop words*, por termos específicos), *stemming*, geração de *tokens*, transformação de dados etc.
- **Web mining:** conjunto de funcionalidades voltadas para informação obtidas de paginas da web, por exemplo: leitura de paginas web e de *webservices*.

Em relação ao uso dessas funcionalidades, a Figura 11 apresenta o componente básico do RapidMiner, o operador.

[1] http://rapid-i.com/content/view/181/196/

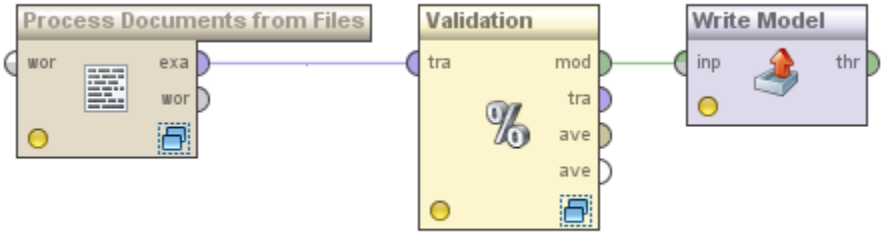

Figura 11: Exemplo de operadores do RapidMiner.

Um operador é um componente que pode ser visto como um sistema caixa-preta, ou seja, pode ser utilizado sem o conhecimento de como o processamento interno é realizado. No exemplo da Figura 11, a saída do operador "Process documents from Files" é utilizada como entrada do operador "Validation", responsável por treinar um modelo de dados a partir de um conjunto de dados, e logo em seguida, esse modelo é enviado para o operador "Write Model", que tem o objetivo de persistir esse modelo para aplicações futuras. O processo apresentado na Figura 11 é gerado a partir de um documento XML, que é apresentado na Figura 12.

```
<operator activated="true" class="process" compatibility="5.3.006" expanded="true" name="Process">
  <process expanded="true">
    <operator activated="true" class="text:process_document_from_file"
      compatibility="5.3.000" expanded="true" height="76" name="Process Documents from Files" width="90" x="112" y="75">
      <list key="text_directories"/>
      <process expanded="true">
        <connect from_port="document" to_port="document 1"/>
        <portSpacing port="source_document" spacing="0"/>
        <portSpacing port="sink_document 1" spacing="0"/>
        <portSpacing port="sink_document 2" spacing="0"/>
      </process>
    </operator>
    <operator activated="true" class="x_validation"
      compatibility="5.3.006" expanded="true" height="112" name="Validation" width="90" x="246" y="75">
      <process expanded="true">
        <portSpacing port="source_training" spacing="0"/>
        <portSpacing port="sink_model" spacing="0"/>
        <portSpacing port="sink_through 1" spacing="0"/>
      </process>
```

Figura 12: XML de um processo do RapidMiner.

No XML apresentado na Figura 12 cada operador é definido pelo elemento `<operator>`, como por exemplo, o operador "Validation" da Figura 11 é definido pelo elemento `<operator>` com o atributo "class" com o valor "x_validation". Além disso, a ferramenta é também uma biblioteca de código, o que permite utilizar os recursos do RapidMiner sem a necessidade de utilizar a ferramenta diretamente. Isso significa que um processo pode ser elaborado dentro do RapidMiner e ser executado por um outro sistema utilizando o XML que descreve o processo.

3.1.2. Eclipse IDE

O Eclipse[2] é um IDE (*Integrated Development Enviroment* – Ambiente Integrado de Desenvolvimento) para programação em Java que foi utilizado nesse trabalho para desenvolver a extensão do RapidMiner. O principal motivo da escolha da ferramenta foi a existência de um projeto *template* para o Eclipse, contendo diversas configurações pré-configuradas, o que facilitou o desenvolvimento das extensões.

3.1.3. Twitter4j

O Twitter4j[3] é uma biblioteca Java para a API do Twitter que encapsula os métodos da API, bem como os parâmetros e retornos em objetos Java, evitando lidar diretamente com a API, o que significa ganho em tempo e redução da complexidade de acesso aos dados do Twitter. Além do mapeamento dos recursos da API, o Twitter4j não necessita de outros recursos adicionais, o que em desenvolvimento de software evita problemas como dependência de código. O Twitter4j também disponibiliza métodos para tratar situações relacionadas com autenticação e autorização de acesso aos dados do Twitter.

3.1.4. Grails

O Grails[4] é um *framework* de desenvolvimento web que utiliza a linguagem Groovy, uma linguagem da plataforma Java. O Grails utiliza o padrão de projetos MVC (*model-view-controller*) o qual separa a camada de visualização de dados (*view*) da camada de definição e gerenciamento de dados (*model*), através de uma camada responsável por gerenciar as requisições e retorna os dados e visualizações. Essa separação de responsabilidade faz com que se desenvolvam, mais facilmente, aplicações que se utilizam de requisição assíncrona, que foi o caso da aplicação desenvolvida nesse trabalho.

O Grails também disponibiliza uma camada de persistência com um banco de dados em memória, que foi utilizado na etapa responsável pela comunicação de dados entre o acesso ao Twitter via Streaming API e a parte responsável pela Análise de Sentimento.

3.2. Procedimentos

Os procedimentos utilizados neste trabalho são divididos nas seguintes categorias:

---

[2] http://www.eclipse.org/

[3] http://twitter4j.org/

[4] http://www.grails.org/

- Levantamento bibliográfico;
- Utilização do RapidMiner;
- Definição do Domínio;
- Aplicação da Técnica de Análise de Sentimentos.

O resultado do levantamento bibliográfico realizado foi apresentado na seção 2, tendo como base os materiais expostos na seção 3.1. As demais etapas são apresentadas a seguir.

3.2.1. A utilização do ambiente de desenvolvimento RapidMiner

Para a compreensão das funcionalidades necessárias do RapidMiner, foram realizadas algumas tarefas de *data mining*, que são chamadas no RapidMiner de processos. Um processo pode executar uma ou mais funcionalidades, por exemplo, de leitura de dados, aplicação de algoritmos, processamento de resultados, não havendo uma limitação quanto às funcionalidades aplicadas, desde que haja compatibilidade entre elas. Nesta etapa do trabalho, foram criados processos de treinamento de técnicas de classificação de dados, como o SVM, além de transformação, exportação e importação de dados.

Com a escolha do domínio, que será apresentado na seção 3.2.2, e a necessidade de acesso às informações desse domínio, gerou a necessidade de estender o RapidMiner, com a criação de novas funcionalidades, para acessar os dados do domínio de forma desejada. Essas novas funcionalidades, que consistem em dois novos operadores que foram criados para o RapidMiner, são apresentadas na seção 4.2.

3.2.2. A definição do Domínio

O domínio escolhido para a aplicação de Análise de Sentimento foi a rede social e *microblogging* Twitter[5]. O Twitter consiste no compartilhamento de informações denominado *tweets*, contendo textos, links e *hashtags*, com no máximo 140 caracteres, o que, de certa forma, limita o assunto de um *tweet*. Ao postar um tweet, este é visto por todos os seguidores do autor do *tweet*, no qual há a possibilidade desses seguidores responder o *tweet* ou compartilhar para seus respectivos seguidores, ação chamada de *retweet*. Uma *hashtag* é utilizada para identificar uma palavra-chave em um *tweet* e é criada com a adição do símbolo "#" no inicio da palavra-chave.

[5] https://twitter.com/

Quando um termo é muito comentado, esse termo ou expressão aparece numa lista chamada de *trending topics* para os usuários do Twitter. Os *trending topics* podem ser sobre termos globais (em todo o Twitter) ou regionais (dependendo da escolha do usuário). Uma situação muito comum é que muitos usuários, tanto perfil de pessoas físicas quando perfis empresariais têm como objetivo entrar na lista dos *trending topics* com seus termos, tanto palavras quanto *hashtags*, objetivando garantir uma visibilidade maior. Essa visibilidade não depende apenas do *trending topics*, mas do compartilhamento de informações entre os usuários.

A escolha do Twitter como domínio foi devido às seguintes características: a limitação de caracteres, um *tweet* geralmente trata apenas de um assunto, que no contexto da Análise de Sentimentos pode ser considerado como a entidade da opinião; a forma rápida de compartilhamento, o que possibilita que diversas opiniões sejam produzidas, e com os valores sentimentais variados, tanto sobre acontecimentos de grande repercussão quanto para situações do cotidiano, tais como produtos comprados, atendimento em serviços, dentre outros.

Para poder obter os *tweets* do domínio foi necessário estudar a API de acesso a dados que é disponibilizada pelo Twitter. Dessa API dois meios foram utilizados para este trabalho: A REST API e a *Streaming* API. Ambas acessam e retornam o mesmo tipo de dado, os *tweets*, diferenciando-se no modo de como esse acesso é realizado. Na seção 4.1 será apresentada mais detalhadamente sobre a API, seu funcionamento e o uso neste trabalho.

3.2.3. A aplicação da técnica de Análise de Sentimentos

Com o domínio escolhido, foi iniciado o processo de Análise de Sentimentos a partir da definição da SVM como técnica de classificação de sentimento. Com isso, foi realizada a aplicação dos métodos de extração de Informação, apresentados na seção 2.1, responsáveis por preparar os dados para as etapas seguintes.

Para o treinamento da SVM foi elaborada uma base de dados de 1600 *tweets*, a partir de 4000 *tweets*, que foram divididos nos seguintes valores sentimentais:

- Positivo: 444
- Negativo: 372
- Neutro: 784

Essa base foi elaborada a partir da análise de elementos presente nos tweets, isto é, adjetivos, verbos, expressões e também *emoticons*. Adjetivos como “bom”, “legal” são indicadores de sentimento positivo, assim como o verbo conjugado “gostei”, o *emoticon* “:)” e

algumas *hashtags* que indicavam valores sentimentais. *Tweets* irônicos, que usavam os indicadores de forma diferente foram classificados conforme o sentido apresentado, ao invés de considerar apenas a palavra principal do texto, com o objetivo de treinar a ferramenta para não utilizar apenas uma palavra no texto, e sim todas a palavras relevantes do texto.

Com essa base de dados, foi realizado o treinamento das SVM. Conforme será apresentado na seção 4.3, esses dados foram usados de duas maneiras diferentes, uma usando apenas os *tweets* positivos e negativos, e a outra usando todos os *tweets* da base de dados. Ao término dessa etapa de treinamento, o classificador de sentimentos foi criado.

A etapa seguinte consistiu na aplicação desse classificador acessando diretamente os dados do Twitter. Esse procedimento foi realizado no RapidMiner com o uso dos operadores que foram criados durante este trabalho para acessar dados do Twitter. Nessa etapa do trabalho o uso das APIs foi necessário, pois somente através delas foi possível coletar os dados do Twitter para então poder aplicar a Técnica de Análise de Sentimentos, e cada uma delas foram utilizadas de modo distintito, devido às características e limitações próprias.

Os resultados obtidos precisaram ser avaliados para verificar se a hipótese lançada por esse trabalho teve resultados satisfatórios ou não. Foram realizadas duas análises diferentes: uma para o treinamento das SVM e outra para Análise de Sentimentos.

Para avaliar o treinamento foram utilizadas informações sobre precisão e cobertura. A precisão é a porcentagem de objetos classificados para uma classe que realmente pertencem a essa classe, e a cobertura é a porcentagem de objetos de uma classe que foram classificados corretamente para essa classe (POWERS, 2007, p. 2). Esses valores são disponibilizados pelo RapidMiner logo após o treinamento da SVM.

Para validar o classificador em um teste real de Análise de Sentimento, foram utilizados os mesmos parâmetros usados na validação do SVM (taxa de precisão e cobertura). A validação dos resultados foi realizada através de uma avaliação manual dos resultados obtidos.

3.2.4. Desenvolvimento da Aplicação

Após a utilização da Técnica de Análise de Sentimentos com o RapidMiner, foi projetada e implementada uma aplicação para possibilitar a pesquisa pelo sentimento associado a termos usados no Twitter. O desenvolvimento da aplicação é descrito na seção 4.4. Devido ao fato da biblioteca do RapidMiner ser escrita em Java, foi necessário escolher uma ferramenta que fosse apta para o RapidMiner, sendo escolhido o Grails.

Na implementação da Aplicação houve duas situações de complexidade relativamente maior. A primeira delas foi utilizar o RapidMiner sem usar a interface que é disponibilizada para o mesmo, que foi contornada com a adição dos arquivos JAR (*Java ARchive*, formato da biblioteca de código Java) no projeto da Grails da aplicação. A segunda situação está relacionada a necessidade da aplicação acessar os *tweets* através da API do Twitter, que também foi contornada devido ao aproveitamento e adaptação do código utilizado para criar os operadores para o RapidMiner.

4 RESULTADOS E DISCUSSÃO

Nesta seção serão apresentados os resultados obtidos durante o desenvolvimento deste trabalho, bem como as análises desses resultados. As ferramentas e procedimentos apresentados na seção 3 foram utilizados neste trabalho conforme apresentado na Figura 13.

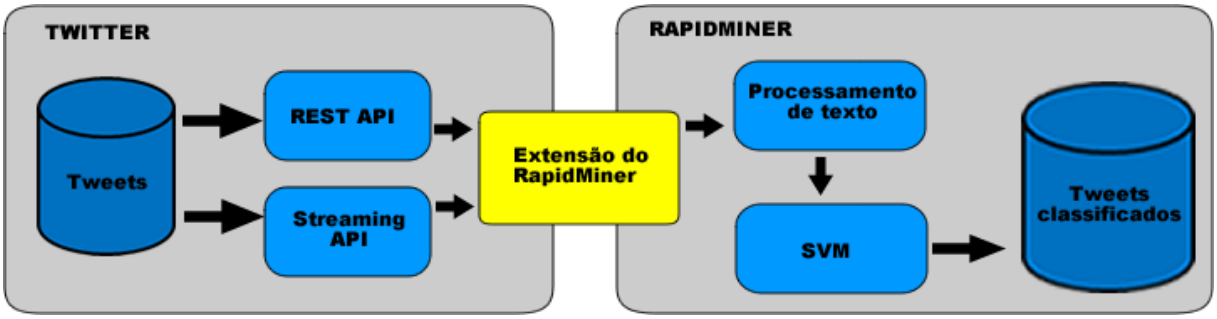

Figura 13: Relação entre os componentes utilizados neste trabalho.

Como apresentado na Figura 13, há dois ambientes distintos: o Twitter e o RapidMiner. Como mencionado na seção 3.2.2, o Twitter foi escolhido como domínio da aplicação, disponibilizando duas APIs para acesso a dados, a REST e a *Streaming* API, que terão apresentadas as características de ambas, bem como a forma de utilização, testes desenvolvidos até a diferença entre ambas. Como forma de interligar esse dois ambientes foram criados duas extensões para o RapidMiner (em amarelo), que será apresentado na seção 4.2. No RapidMiner foi desenvolvido o processo de Análise de Sentimentos, que resultou nos *tweets* classificados de acordo com os respectivos valores sentimentais. Esse processo será apresentado na seção 4.3.

Foi desenvolvida uma aplicação web para realizar a Análise de Sentimento a partir da utilização do RapidMiner, que mudou a relação entre os componentes utilizados nesse trabalho, conforma a Figura 14.

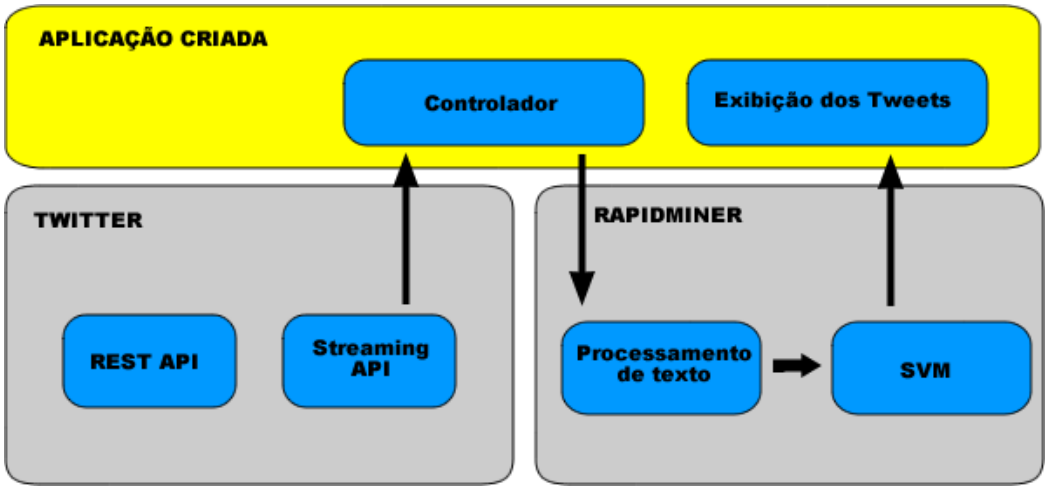

Figura 14: Relação entre componentes utilizados no desenvolvimento da aplicação.

A principal diferença entre a arquitetura apresentada na Figura 14 e a da Figura 13 é que a integração entre os dados do Twitter e o RapidMiner é realizada pela aplicação desenvolvida, que será apresentada na seção 4.4.

4.1. API do Twitter

Uma API (*Application Program Interface*) é o meio de acessar um recurso de uma aplicação sem a necessidade de utilizar a aplicação em si. Em aplicações web, a API é um serviço disponibilizado para acessar os recursos do sistema usando apenas requisições HTTP, sem a necessidade de utilizar a interface HTML para o mesmo.

O Twitter disponibiliza uma API que permite utilizar todos os recursos que estão disponíveis no site, isto é, acesso aos *tweets* postados, postagem de novos *tweets*, acesso aos contatos, busca por *tweets*, entre outros. A API do Twitter utilizada é divida em dois grupos: REST API e *Streaming* API. A REST API oferece os mesmos serviços que disponibilizado aos usuários do Twitter através do site "twitter.com", enquanto o *Streaming* API oferece acesso a todos os *tweets* através de uma conexão de baixa latência (menor atraso de recepção) em relação à REST API. Isso interfere também na interação entre o usuário, o site que usa a API do Twitter e o Twitter em si. Na Figura 15 é apresentada essa interação no caso da REST API.

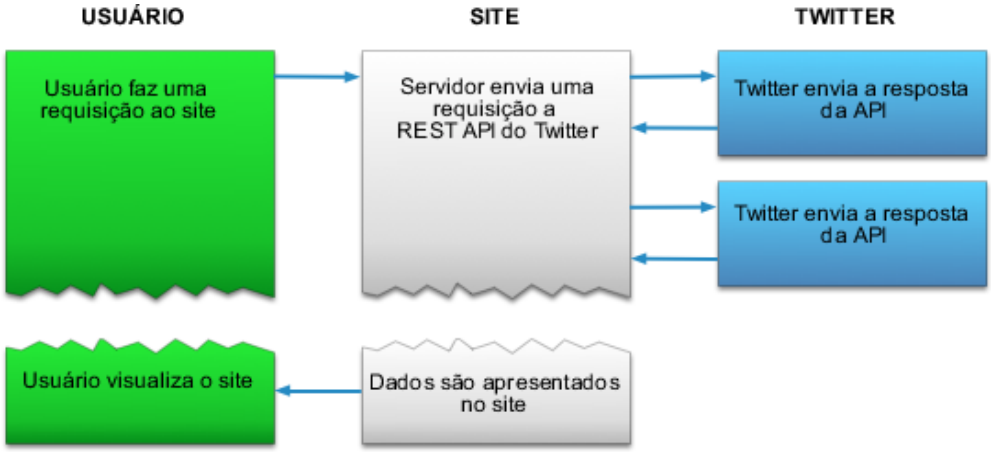

Figura 15: Interação usuário-site-Twitter com a REST API (TWITTER, online, adaptado).

Na situação apresentada na Figura 15, o acesso e processamento dos dados obtidos do Twitter são realizados de forma sincronizada com a exibição dos resultados ao usuário. Para a maioria das aplicações, a REST API é suficiente para atender as necessidades, pois disponibiliza todas as funcionalidades encontradas no site, enquanto a *streaming* API é mais adequada para um caso específico, como a recuperação de *tweets*. A *Streaming* API tem seu funcionamento apresentado na Figura 16.

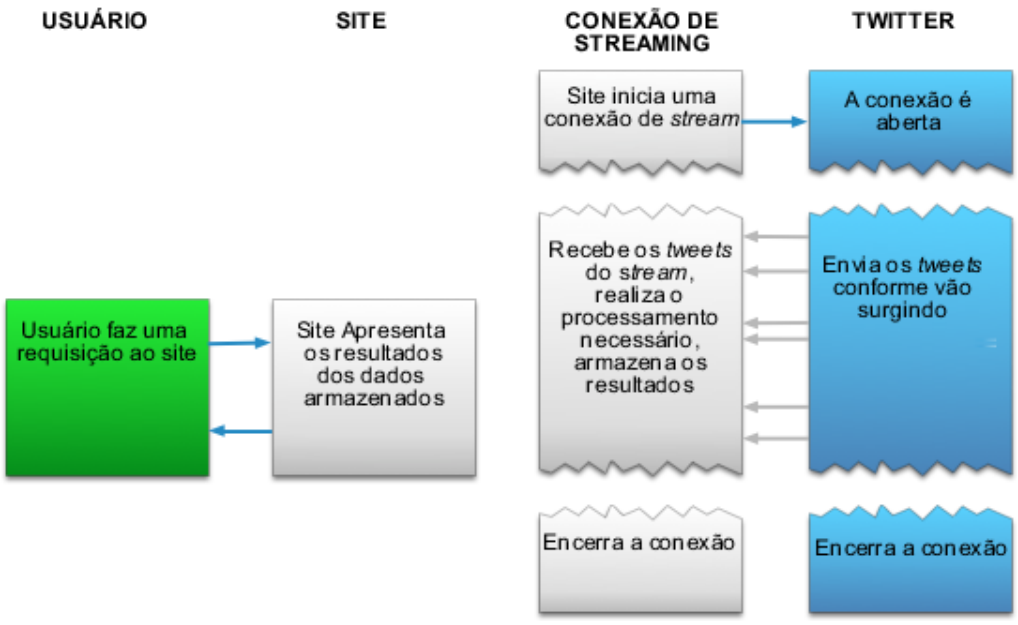

Figura 16: Interação usuário-site-Twitter com o Streaming API (TWITTER, online, adaptado).

Na situação apresentada na Figura 16 é possível verificar a realização de acesso aos *tweets* de forma assíncrona em relação à exibição dos resultados ao usuário, já que a conexão está sempre aberta e os dados são entregues assim que são criados. Essa abordagem é

importante em situações em que há a necessidade de ler uma grande quantidade de dados e oferecer um *feedback* contínuo ao usuário.

O acesso ao *tweets* públicos através da REST API tem uma desvantagem em relação ao *Streaming* API: um limite de, no máximo, 100 *tweets* por requisição, além de, no máximo, 450 requisições em um espaço de 15 minutos. Apesar dessa desvantagem, a REST API, por receber todos os resultados em uma única requisição, conforme a Figura 15, possibilita que a aplicação que esteja consumindo a API crie resultados mais rápidos do que se usasse a Streaming API, que conforme a Figura 16, recebe os resultados com o decorrer do tempo, logo depende da quantidade de *tweets* que estão sendo produzidos no momento. Com isso percebe-se que o a escolha de qual API utilizar depende da utilização dos tweets retornados, por exemplo, o uso da REST API seria indicado melhor em casos onde há necessidade de um retorno rápido de resultados sem a necessidade uma grande quantidade *tweets*. E no caso da *Streaming* API, um exemplo de uso seria onde uma grande quantidade de *tweets* é necessário, e/ou para evitar que o limite de requisições da REST API seja atingido.

O acesso aos recursos é feito através dos métodos HTTP GET (para consultas) e POST (para envio de dados), com o objetivo de simplificar ao máximo o acesso aos recursos. A resposta da requisição está no formato JSON[6] (*JavaScript Object Notation*), um formato simples de troca de dados que pode ser manipulado por um grande número de linguagens de programação. Nas seções 4.1.1 e 4.1.2 serão apresentados mais detalhadamente os serviços disponibilizados pela API. Para simplificar o uso da API do Twitter foi utilizada a biblioteca Java Twitter4J.

A REST API foi utilizada nesse trabalho, primeiramente, por oferecer um meio mais simples de trabalhar com os dados do Twitter, em relação a *Streaming* API, pois é possível obter os *tweets* com apenas uma requisição HTTP. Além disso, é possível ter um tempo de resposta mais rápida, apesar do limite de valores retornados. Já a *Streaming* API foi utilizada por possibilitar acompanhar e analisar assuntos por um período grande de tempo, além de uma grande quantidade de *tweets* e, devido a essa característica possibilitar, por exemplo, acompanhar o resultado da análise de sentimento durante um intervalo definido. A *Streaming* API, por ser em tempo real, só retorna os tweets que estão sendo criados durante a conexão, o que impossibilita de ter acesso aos tweets anteriormente criados. Já a REST API não tem essa

[6] http://www.json.org/

limitação, podendo até retornar *tweets* criados horas antes da requisição. Sendo assim cada API possui situações em que eu uso é o melhor aplicado.

O acesso a API do Twitter é publico, mas não é realizado de forma anônima, sendo necessária uma forma de identificação que permita que o Twitter controle o acesso a API, bem como a autorização para cada recurso. A forma como foi trabalhada essa autenticação no Twitter é apresentada na seção 4.1.3.

4.1.1. REST API

Como mencionado anteriormente, a REST API permite acessar as mesmas funcionalidades que o site do Twitter, esses recursos, segundo Twitter (online) são agrupados em:

- **Timeline:** acesso aos *tweets* da página inicial do usuário.
- **Tweets:** postagem de *tweets* e de *retweets*.
- **Search:** pesquisa de *tweets*.
- **Direct Message:** leitura e envio de mensagens privadas.
- **Friend & Followers:** listagem de contatos e gerenciamento dos mesmos.
- **Users:** acesso as configurações, perfil do usuário, entre outras informações relacionadas à informação do usuário.
- **Suggested Users:** visualização dos usuários que o Twitter sugere para seguir.
- **Favorites:** acesso e criação de *tweets* marcados como favoritos.
- **Lists:** gerenciamento das listas criadas, bem como criação e exclusão das mesmas.
- **Saved Search:** criação e acesso às consultas criadas, o que permite o reuso das mesmas.
- **Places & Geo:** informações geográficas (latitude e longitude) sobre locais, bem como possibilidade de realizar consulta com bases em dados geográficos.
- **Trends:** visualização dos *trends topics*, isto é, os termos mais comentados no Twitter.
- **Spam Reporting:** permite denunciar e bloquear usuários indesejados.
- **OAuth:** gerenciar a autenticação e autorização da aplicação consumidora no Twitter.
- **Help:** disponibiliza funcionalidades para quem estiver desenvolvendo com a API do Twitter.

Dos recursos apresentados acima, os métodos de *search* foram os utilizados nesse trabalho, pois a Análise de Sentimento foi aplicada nos *tweets* recuperados. Esse recurso de busca é feito através da URL: `https://api.twitter.com/1.1/search/tweets.json`. O acesso é feito através de uma requisição do tipo GET e aceita como parâmetros o termo da

consulta, área de busca, linguagem, período de busca, limite de resultados, entre outros. O retorno é um JSON com uma lista de objetos que contêm dados sobre a mensagem postada, dados sobre localização e sobre o usuário que o postou.

Para exemplificar o uso da API foi realizada uma consulta sobre o termo "#doctorwho". Além disso, a consulta foi definida para *tweets* em português, limitado a no máximo 7 resultados. A URL utilizada para efetuar essa busca é definida por: `https://api.twitter.com/1.1/search/tweets.json?q=#doctorwho&lang=pt&count=7`. O código em Java para realizar essa requisição é apresentado no Código-fonte 1.

```
26      public static void main(String[] args) {
27          try{
28
29              Twitter twitter= new TwitterFactory(
30                      getConfiguration()).getInstance();
31              Query query=new Query("#doctorwho");
32              query.setLang("pt");
33              query.setCount(7);
34              QueryResult result= twitter.search(query);
35              for (Status tweet: result.getTweets()){
36                  System.out.println("@"+tweet.getUser().getScreenName()
37                          +" says:\n\t"+tweet.getText());
38              }
39          }
40          catch (Exception e) {
41              System.out.println("erro: "+ e.toString());
42          }
43      }
```

Código-fonte 1: Busca de um termo no Twitter pela API.

A biblioteca Twitter4J se encarrega de montar a URL e realizar a requisição e converter a resposta da API em um objeto. No Código-fonte 1, a instância da classe "`Twitter`" contem como métodos os recursos da API, que no caso apresentado é o método de busca, que é chamado na linha 30. A instância de "`Query`" encapsula os parâmetros de entrada. Na Figura 17 é apresentado o resultado da consulta.

```
@lomyne says:
        Vício, vício, o NetFlix tinha razão, eu amo. #DoctorWho #GetGlue http://t.co/h8zpOy6pK5
@BowtiesAndWebs says:
        A quinta temporada de Doctor Who é uma obra de arte sem precedentes. #DoctorWho #TeamMoffat #TeamMatt
@Newtondossantos says:
        é isso define a era moffat  - Dr Who: Bla Bla Bla Daleks: http://t.co/dILFmDBrJA via #Doctorwho
@llajkjdsghfd says:
        RT @universowho: FALTAM 6 DIAS! Doctor Who retorna em 30 de Março com 'The Bells of St John'. #DoctorWho
@tacamara says:
        os efeitos kkkkkkkkkkk #doctorwho
@MariBookBlog says:
        Quando eu fico obcecada com alguma coisa eu começo a escrever fanfics LSAKDNÇASNDKAJS #DoctorWho
```

Figura 17: Resultado formatado da consulta a API do Twitter.

Nos resultados apresentados na Figura 17, são exibidos alguns dos elementos retornados na consulta, isto é, o nome e o texto dos *tweets*. Para este trabalho foi utilizado apenas o segundo item, pois a Análise de Sentimento, nesse contexto, não considerou o emissor da opinião. Entre as diversas informações que também são retornadas pela API, têm-se: a data de criação do *tweet*, as coordenadas de geolocalização, um indicador que informa se o *tweet* foi marcado como favorito, um indicador que informa se o *tweet* é um *retweet*, o idioma e os dados do usuário.

4.1.2. Streaming API

O *Streaming* API é disponibilizado a partir de três modos (TWITTER, online):

- **Public Stream:** Disponibiliza os *tweets* públicos existentes no Twitter.
- **User Stream:** Disponibiliza todos os dados de um usuário.
- **Site Stream:** Semelhante ao anterior, mas não se limitando a um único usuário.

Dos modos apresentados anteriormente, o *Public Stream*, que é utilizado nesse trabalho, apresenta três recursos distintos:

- **Filter:** fornece *tweets* que podem ser filtrados tanto por termos, localização e/ou usuário. URL: `https://stream.twitter.com/1.1/statuses/filter.json`
- **Sample:** disponibiliza uma amostra aleatória dos *tweets* públicos do Twitter. URL: `https://stream.twitter.com/1.1/statuses/filter.json`
- **Firehouse:** disponibiliza todos os *tweets* públicos do Twitter. URL: `https://stream.twitter.com/1.1/statuses/firehouse.json`.

Um exemplo de uso do *Streaming* API é apresentado no Código-fonte 2.

```
46      public static void main(String[] args){
47          try {
48              TweetListener listener=new TweetListener();
49
50              TwitterStream twitterStream=new TwitterStreamFactory(
51                      getConfiguration()).getInstance();
52              twitterStream.addListener(listener);
53              twitterStream.sample();
54
55          }
56          catch (Exception e) {
57
58          }
59      }
```

```
109     @Override
110     public void onStatus(Status tweet) {
111         System.out.println("@"+tweet.getUser().getScreenName()
112                 +"("+tweet.getUser().getLang()+")"
113                 +" says===>"+tweet.getText());
114     }
```

Código-fonte 2: Amostra de *tweets* pelo Streaming API.

Na primeira parte do Código-fonte 2, na linha 50, é feita a configuração da conexão, onde o método `getConfiguration` contém a configuração de autenticação (que será tratada na próxima seção). A instância da classe `TweetListener` (linha 48) é a responsável por processar os *tweets* assim que eles forem recuperados. O recurso acessado é o de amostragem (linha 53). Na segunda parte da imagem há um trecho da classe `TweetListener`. Esse método é executado assim que o evento de um novo *tweet* é recuperado, exibindo o nome do usuário, a linguagem e o conteúdo postado. Na Figura 18 é apresentado o resultado desse código.

```
@BibeenyS499(en) says===>Spring Breakers Merely f'n amazing ""
@Pphilly13(en) says===>RT @Lil_Mexiico: Sometimes to see the Light you have to Risk the Dark.
@P_I_L_L(ja) says===>@kyousokun ??
@kyra_word(en) says===>RT @imlolabashang: "I have the best of friends ever."
@kn21luv(ja) says===>@94riho__1214 ????????????
```

Figura 18: Resultado da requisição de amostra do *Streaming* API.

O exemplo apresentado na Figura 18 contém apenas alguns das informações retornadas na consulta, assim como o exemplo da Figura 17, pois ambas as respostas possuem o mesmo formato. Os sinais de interrogação em dois dos *tweets* são caracteres em japonês que não foram exibidos devido à codificação de caracteres.

4.1.3. Autenticação

Ao utilizar da API do Twitter, é necessário que o acesso seja feito de forma autenticada, tanto para aplicações que acessam e manipulam os dados do usuário quanto para o acesso de

informações públicas. No primeiro caso, além da autorização dada a quem utiliza a API, é necessária também a autorização do usuário do Twitter que terá seus dados acessados, o que é o caso de diversas aplicações existentes. O método de autenticação é definido através da tecnologia OAuth[7].

Como neste trabalho é utilizado apenas acesso aos dados públicos, o *token* de acesso foi obtido através do registro de uma aplicação no site para desenvolvedores (dev.twitter.com). Entre as configurações existentes é possível limitar o acesso aos *tweets*: somente leitura; leitura e escrita; e leitura, escrita e acesso as mensagens privadas. A utilização dessa autorização é mostrada no Código-fonte 3.

```
14    private static Configuration getConfiguration(){
15        ConfigurationBuilder cb=new ConfigurationBuilder();
16        cb.setDebugEnabled(true)
17          .setOAuthConsumerKey("secret_token")
18          .setOAuthConsumerSecret("secret_token")
19          .setOAuthAccessToken("secret_token")
20          .setOAuthAccessTokenSecret("secret_token");
21        return cb.build();
22    }
```

Código-fonte 3: Dados da autenticação.

O método `getConfiguration` do Código-fonte 3 é o mesmo que é utilizado nos códigos-fonte Código-fonte 1 e Código-fonte 2 e contém apenas a configuração dos dados de acesso, isto é, os *tokens*. Os dois primeiros parâmetros (linhas 17 e 18) são a identificação da aplicação que acessa a API do Twitter, que é usada para verificar se a aplicação tem autorização para utilizar o recurso. O parâmetro da linha 17 é o utilizado nas requisições, e o da linha 18 é utilizado pela aplicação para confirmar a autenticidade da requisição, que é feito automaticamente pelo *framework* utilizado. E os parâmetros das linhas 19 e 20 são utilizados para identificar o usuário (do Twitter) autenticado para acessar a API. Semelhante a identificação da aplicação, há dois valores: o utilizado na requisição (linha 19) e o utilizado para confirmação de identidade (linha 20).

4.2. Extensão para o *RapidMiner*

[7] http://oauth.net/

A partir de um conjunto de dados o RapidMiner, é possível treinar um classificador e, caso sejam dados textuais, transformar essas informações em dados que possam ser manipulados. Mas há situações em que o RapidMiner não tem atuação, um exemplo disso é o acesso aos *tweets*, e a transformação dos mesmos em um formato que possa ser utilizado pelo *RapidMiner*.

Para contornar essa situação, o próprio *RapidMiner* disponibiliza uma solução: o uso de extensões (ou *plugins*). As extensões servem para adicionar novas funcionalidades ao RapidMiner através de novos operadores, que são integrados às demais funcionalidades. Para este trabalho foi criado o operador **Twitter Reader** e **Process Tweets from Stream**.

O uso de cada operador tem relação com a API do Twitter utilizada. Como o operador Twitter Reader utiliza REST API, é o melhor indicado na construção de processo em que se deseja utilizar um número pequeno de *tweets*, com um tempo de resposta mais rápido possível. E o operador Process Tweets from Stream, por usar a *Streaming* API, é mais indicado, por exemplo, onde é necessário trabalhar com um grande numero de *tweets*.

Operador Twitter Reader

O operador criado tem como objetivo ler os *tweets* utilizando a REST API do Twitter, tendo como base os estudos apresentados na seção 4.1.1. O Código-fonte 4 mostra a classe e os atributos do operador.

```
21 public class TwitterReaderOperator extends Operator {
22     private OutputPort exampleSetOut=getOutputPorts().createPort("example set");
23 
24     private final String PARAMETER_QUERY="query";
25     private final String PARAMETER_QUANTITY="quantity";
26 
27 
28     private Attribute[] attribute_list;
29 
30     public TwitterReaderOperator(OperatorDescription description) {
31         super(description);
32 
33         getTransformer().addGenerationRule(exampleSetOut, ExampleSet.class);
34 
35         attribute_list=new Attribute[]{
36                 AttributeFactory.createAttribute("user",Ontology.NOMINAL),
37                 AttributeFactory.createAttribute("user-lang",Ontology.NOMINAL),
38                 AttributeFactory.createAttribute("tweet",Ontology.STRING),
39                 AttributeFactory.createAttribute("isRT",Ontology.BINOMINAL)
40         };
41     }
```

Código-fonte 4: Classe do operador Twitter Reader.

Como visto no Código-fonte 4, para criar um operador, este deve obrigatoriamente herdar a classe `Operator`, que também disponibiliza diversas funcionalidades para a criação de um operador. O primeiro atributo da classe (linha 22), que é do tipo `OutputPort`, é um objeto responsável por enviar a saída para outros operadores, que nesse caso são os *tweets* obtidos; esses objetos são responsáveis pelas conexões entre os operadores. Os objetos `OutputPort` são referenciados por um nome ("example set") e são utilizados pelo RapidMiner para indicar ao usuário qual é o tipo de dado trabalhado. Além do objeto `OutputPort`, há também objetos do tipo `InputPort`, que são responsáveis por receber os dados do operador. Sobre o tipo de dados gerados (e também os tipos recebidos), é criada uma validação (linha 33), onde é definido que a saída deverá ser uma instância de um `ExampleSet`, isto é, um objeto que disponibiliza acesso a uma coleção de dados. O atributo da linha 28 contém a configuração da tabela de dados que o operador irá criar, e sua definição é feita da linha 35 a 40. Com isso é definido que o operador irá gerar uma tabela de dados com quatro colunas:

- **user:** nome de usuário do Twitter.
- **user-lang:** idioma do usuário.
- **tweet:** o texto que compõe o *tweet*, com o parâmetro `Ontology.STRING` definindo que a coluna será do tipo texto.
- **isRT:** campo de dois valores apenas (`Ontology.BINOMINAL`), indicando se o *tweet* foi criado pelo usuário ou foi um *retweet*.

Ainda em relação ao Código-fonte 4, os atributos das linhas 24 e 25 são duas variáveis que armazenam os identificadores dos parâmetros do operador ("query" e "quantity"), a configuração dos parâmetros é apresentada no Código-fonte 5.

```
72  @Override
73  public List<ParameterType> getParameterTypes() {
74      List<ParameterType> parameters=super.getParameterTypes();
75
76      parameters.add(
77          new ParameterTypeString(PARAMETER_QUERY, //identificador do parâmetro
78              "query for search", // texto de descrição
79              false, // campo opcional
80              false // tipo de parâmetro (avançado ou básico)
81          )
82      );
83      parameters.add(
84          new ParameterTypeInt(
85              PARAMETER_QUANTITY, //identificador do parâmetro
86              "number maximum of returned tweets", // texto de descrição
87              1, // valor minimo
88              100, // valor máximo
89              25, // valor padrão
90              false // tipo de parâmetro (avançado ou básico)
91          )
92      );
93
94      return parameters;
95  }
```

Código-fonte 5: Configuração dos parâmetros do operador Twitter Reader.

O método `getParameterTypes` do código do Código-fonte 5 deve ser sobrescrito para adicionar os parâmetros. O primeiro parâmetro (linhas 76 a 82) é do tipo de texto e se chamara “query” sendo um campo obrigatório e do tipo básico. Já o segundo parâmetro (linhas 83 a 92) é do tipo inteiro, aceitando valores no intervalo de 1 a 100, tendo como 25 o valor inicial, sendo também do tipo básico. O tipo de parâmetro (básico ou avançado) é utilizado pelo RapidMiner para proporcionar uma interface que seja adaptada ao usuário de acordo com o nível de conhecimento do mesmo sobre o processamento oferecido pelo operador. Parâmetros avançados devem possuir um valor padrão, e sua configuração pode ser ocultada pelo usuário do RapidMiner.

O processamento de um operador é realizado pelo método `doWork`, apresentado no Código-fonte 6.

```
44 public void doWork() throws OperatorException {
45     try {
46         Twitter twitter=TwitterConfiguration.getTwitterInstance();
47         Query query=new Query(getParameterAsString(PARAMETER_QUERY));
48         query.setLang("pt");
49         int numberMaxTweet= getParameterAsInt(PARAMETER_QUANTITY);
50         query.setCount(numberMaxTweet);
51         QueryResult result= twitter.search(query);
52 
53         MemoryExampleTable table=new MemoryExampleTable(this.attribute_list);
54         DataRowFactory rowFactory=new DataRowFactory(
55                 DataRowFactory.FIRST_TYPE_INDEX,'.');
56         for (Status tweet: result.getTweets()){
57             DataRow dataRow=rowFactory.create(
58                     new Object[]{
59                             tweet.getUser().getScreenName(),
60                             tweet.getUser().getLang(),
61                             tweet.getText(),
62                             tweet.isRetweet()?"yes":"no"},
63                     this.attribute_list);
64             table.addDataRow(dataRow);
65         }
66 
67         exampleSetOut.deliver(table.createExampleSet());
```

Código-fonte 6: Código responsável pela funcionalidade do operador Twitter Reader.

O processamento feito no Código-fonte 6 pode ser sistematizado em três partes. A primeira parte (das linhas 46 a 51) é a que faz a consulta no Twitter, conforme foi explicado na seção 4.1.1, que consiste em criar objeto que acessará o Twitter, depois definir os parâmetros da consulta (termo consultado, idioma dos resultados, e quantidade máxima) e, por fim, executando a consulta. A diferença existente no código apresentado no Código-fonte 6 é que o termo da busca e a quantidade máxima de resultado são definidos pelos parâmetros do operador, através dos métodos `getParameterAsString` (linha 47) e `getParameterAsInt` (linha 49) que retornam os valores do parâmetro, de acordo com o identificador passado. Na segunda parte (linhas 53 a 65), os *tweets* são formatados para os tipos do RapidMiner, e ficam armazenados em memória, através da classe `MemoryExampleTable`. E, por fim, (linha 66), é gerado um objeto `ExampleSet`, que fornecerá a visualização dos dados. Na Figura 19 é apresentado o operador sendo executado no RapidMiner.

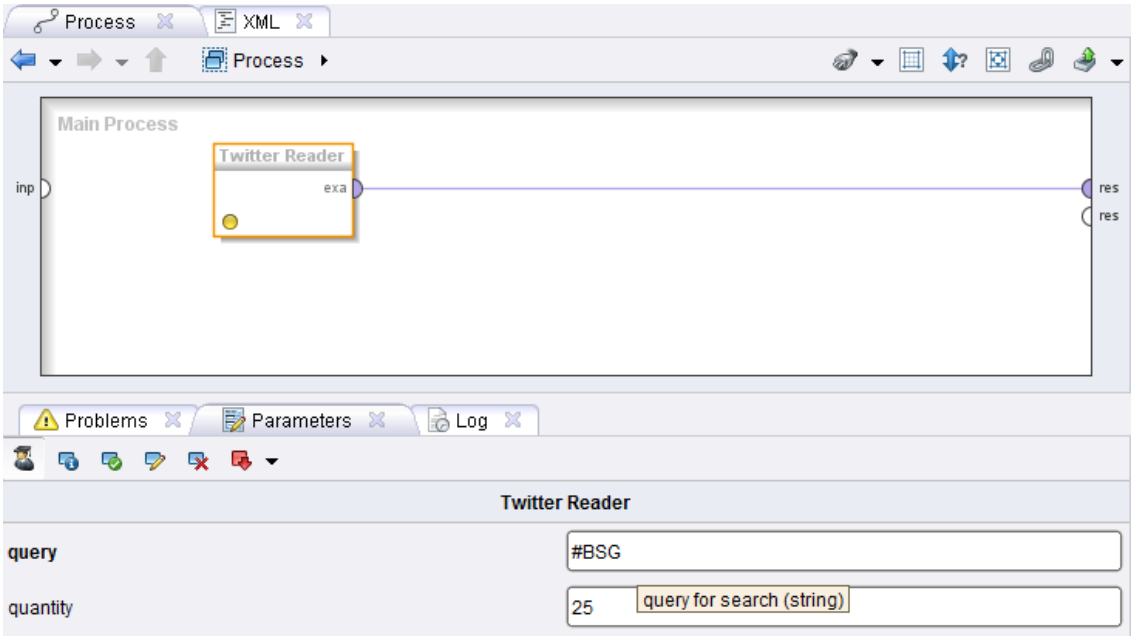

Figura 19: Operador Twitter Reader no RapidMiner.

Conforme a Figura 19, e de acordo com as definições apresentadas nos Código-fonte 5 e Código-fonte 6, é visualizada a saída do operador, bem como seus parâmetros de consulta e quantidade máxima de resultados. Na Figura 20 é apresentado o resultado do uso do operador.

Ro...	user	user-lang	tweet	isRT
1	Raphvcl	pt	Fazendo trabalho ao som da trilha de #BSG. Inspirador	no
2	cepheids_	en	DO I LOVE GAIUS BALTAR OR DO I HATE HIM???? #bsg #s1	no
3	tecnauta	en	E a boa notícia do dia. Nova expansão de #bsg #boardgame saindo!! @F	no
4	pedrobeck	en	Alguém por um acaso tem essa edição da EW? Pago bom dinheiros por	no
5	pedrobeck	en	#Defiance estreou cheia de buzz, mas é #Helix, que estréia no meio do a	no
6	cassiano	en	ADEUS VIDA SOCAIAL: http://t.co/dr7CADse2v #BSG	no
7	xBeiro	en	Q igual é algo precipitado, pero, Ron Moore (creador de #BSG), take my n	no

Figura 20: Resultado do Operador.

No resultado apresentado na Figura 20 os *tweets* já estão aptos para serem utilizados em um processo do RapidMiner como qualquer outro tipo de dado importado. Como pode ser notado, alguns dos resultados retornados são *tweets* em inglês (en), tanto pelo atributo "user-lang" quanto pelo conteúdo em si. O atributo "user-lang", que apesar de indicar qual foi o idioma que o usuário do Twitter definiu, não influencia nos resultados da consulta, pois a identificação do idioma dos resultados é feita através da análise do conteúdo dos *tweets*.

Operador "Process Tweets from Stream"

O objetivo desse operador é recuperar os *tweets* através do *Streaming* API e aplicar esses *tweets* a um subprocesso, armazenando os resultados e exibindo resultados gerados. Esse subprocesso consistirá na aplicação da técnica de "Análise de Sentimento" nos *tweets* retornados. No Código-fonte 7 é apresentada a classe que define o operador.

```
30 public class ProcessTweetsOperator extends OperatorChain {
31     public static final String PARAMETER_QUERY="query";
32     public static final String PARAMETER_TOUCH_FILE="stop touch file";
33     public static final String PARAMETER_TEMP_FILE="temporary file";
34 
35     private OutputPort subprocessInput=getSubprocess(0).getInnerSources().createPort("tweets");
36     private InputPort subprocessOut=getSubprocess(0).getInnerSinks().createPort("result");
37     private OutputPort operatorOutput=getOutputPorts().createPort("example set");
38 
39     private Attribute[] attribute_list;
40 
41     public ProcessTweetsOperator(OperatorDescription description) {
42         super(description,"Reading tweets");
43         getTransformer().addGenerationRule(operatorOutput, ExampleSet.class);
44         attribute_list=new Attribute[]{
45                 AttributeFactory.createAttribute("user",Ontology.NOMINAL),
46                 AttributeFactory.createAttribute("user-lang",Ontology.NOMINAL),
47                 AttributeFactory.createAttribute("isRT",Ontology.BINOMINAL),
48                 AttributeFactory.createAttribute("tweet",Ontology.STRING)
49         };
50     }
```

Código-fonte 7: Classe ProcessTweetsOperator.

O operador "Process Tweets from Stream", que é apresentado no Código-fonte 7, é do tipo `OperatorChain`, o que possibilita a existência de um subprocesso a um operador. Os três primeiro atributos são os identificadores dos parâmetros do operador. Assim como no operador "TwitterReader" há o parâmetro "query", no qual é informado qual o termo que será consultado no Twitter. Os dois parâmetros seguintes têm como objetivo controlar o processamento. O primeiro desses parâmetros, PARAMETER_TOUCH_FILE, receberá o endereço de um arquivo, e quando esse arquivo for criado o acesso ao *Streaming* API será encerrado. E o outro parâmetro, PARAMETER_TEMP_FILE, armazenará o nome do arquivo que servirá de comunicação entre a *thread* que está acessando o Streaming API e a *thread* do RapidMiner.

Nas linhas 35 a 37 do Código-fonte 7 são definidos as portas de entrada e saída do operador. A primeira porta (linha 35) irá passar os *tweets* obtidos do *Streaming* API para o processo interno. A segunda porta (linha 36) é a conexão que transmite os resultados do processo interno para o operador. Esses dados são, então, passados para a porta de saída do operador (definido na linha 37). Na Figura 21 é apresentado o operador no RapidMiner.

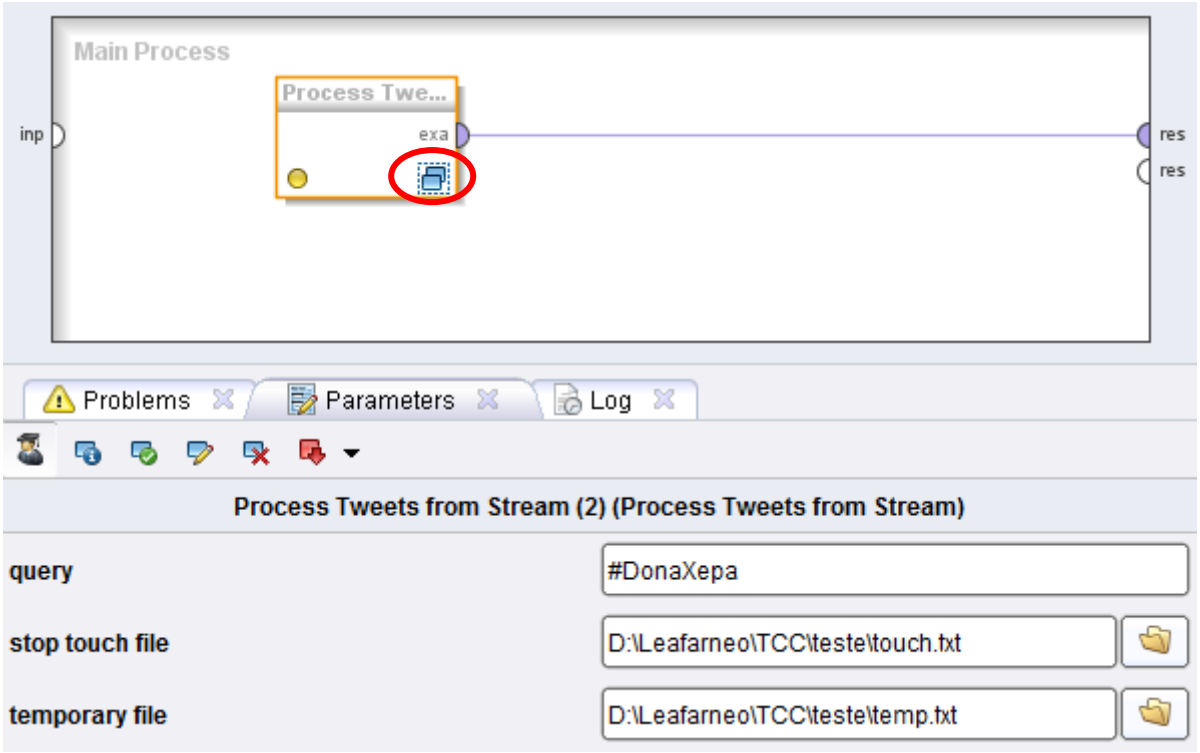

Figura 21: Operador "Process Tweets from Stream" no RapidMiner.

O operador apresentado na Figura 21 possui um símbolo (destacado na figura) indicando que se trata de um operador que possui um processo interno, sendo este processo uma das grandes diferenças em relação ao operador apresentado na Figura 19. O operador pode ser utilizado com ou sem o subprocesso, que será utilizado na seção 4.3.

Na Figura 22 é apresentado o uso do operador sem o subprocesso.

ExampleSet (6 examples, 0 special attributes, 4 regular attributes)

Row No.	user	user-lang	isRT	tweet
1	helenballet	pt	no	Pirando hj ao ver#Donaxepa ja é sucesso,parabes a t
2	Manuela_Duarte	pt	no	FELIZ DESSE JEITO!!! Parabéns a TODOS que fizeran
3	AmoLuAr_Willy	pt	no	Hoje Foi Meu Thur - Xepinho O Diego Montes A Pérola
4	NosTrendsBrasil	pt	no	Audiência da estreia de #DonaXepa, nova novela da R
5	AmoLuAr_Willy	pt	no	Hoje Foi Meu Thur - Xepinho O Diego Montes A Pérola
6	NosTrendsBrasil	pt	no	Audiência da estreia de #DonaXepa, nova novela da R

Figura 22: Parte do resultado do operador "Process Tweets from Stream".

O resultado apresentado na Figura 22 possui a mesma estrutura de dados do operador que utiliza a REST API, isto é, nome do usuário, idioma, indicador se é um *retweet* e o texto do *tweet*. Como a saída do operador é do tipo `ExampleSet`, o formato de saída dos dados varia conforme o resultado do processo interno, que nesse caso foram os tweets da *Streaming* API.

4.3. Processo de Análise de Sentimentos

Nessa seção serão apresentados os processos e testes criados para a etapa de Análise de Sentimentos, tendo como base teórica o conteúdo sobre classificação de sentimento e SVM apresentados na seção 2.3.1, e como base de treinamento os tweets pré-classificados que foram apresentados na Seção 3.2.3. Para tanto, foram realizados testes para a classificação de sentimentos em um conjunto de *tweets* a partir da sistematização de dois casos:

1. Classificar *tweets* com valor sentimental positivo (“pos”) ou negativo (“neg”)
2. Classificar *tweets* com o valor sentimental positivo, negativo ou neutro (para *tweets* sem valor sentimental, ou seja, que representam apenas fatos e não emoção/sentimento).

1º caso

Para realizar o treinamento do SVM foram utilizados apenas os 816 *tweets* positivos e negativos da base de *tweets*, conforme foi apresentado na seção 3.2.4 sobre a aplicação da Análise de Sentimentos. Os dados de treinamento foram armazenados em arquivos de texto em formato **CSV**. O processo de treinamento é apresentado na Figura 23.

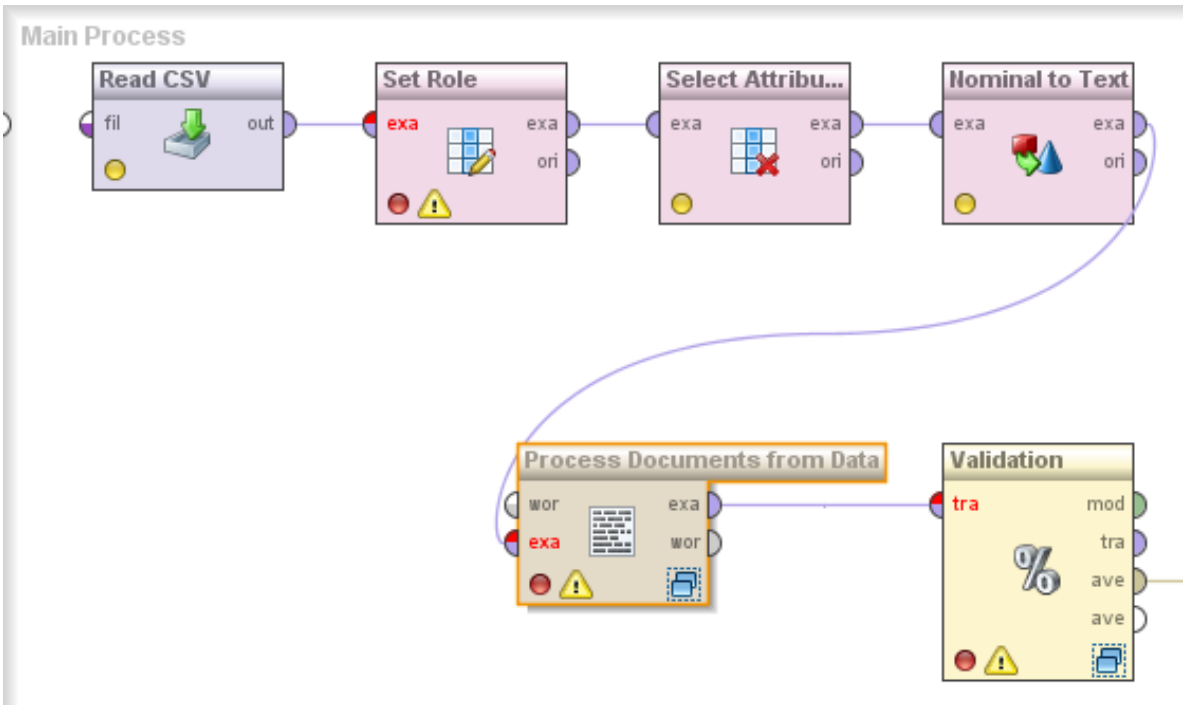

Figura 23: Treinamento do SVM para classificação de sentimentos.

O operador **Read CSV** da Figura 23 acessa um arquivo CSV e gera uma tabela de dados, contendo o nome de usuário, linguagem do usuário, o texto do *tweet*, um indicador se o *tweet* é um *retweet* e a classificação do *tweet*. O operador **Set Role** altera o papel, o objetivo

de um atributo (coluna), que nesse caso é modificar o tipo de regular (atributo normal de informação) para *label*, para indicar que o mesmo é um rótulo para os operadores seguintes. Os operadores seguintes (**Select Attribute** e **Nominal to Text**) selecionam apenas o campo atributo *tweet* e o convertem para o tipo *text* (texto), pois o próximo operador só trabalha com dados do tipo *text*. O operador **Process Documents from Data** processa um conjunto de documentos para gerar uma representação vetorial dos mesmos, no qual esses vetores serão utilizados para treinar o SVM, tarefa essa executada pelo operador **Validation**. Os operadores de processamento de documento e de validação possuem um processamento interno. Na Figura 24 é apresentado como é efetuado o processamento de documentos.

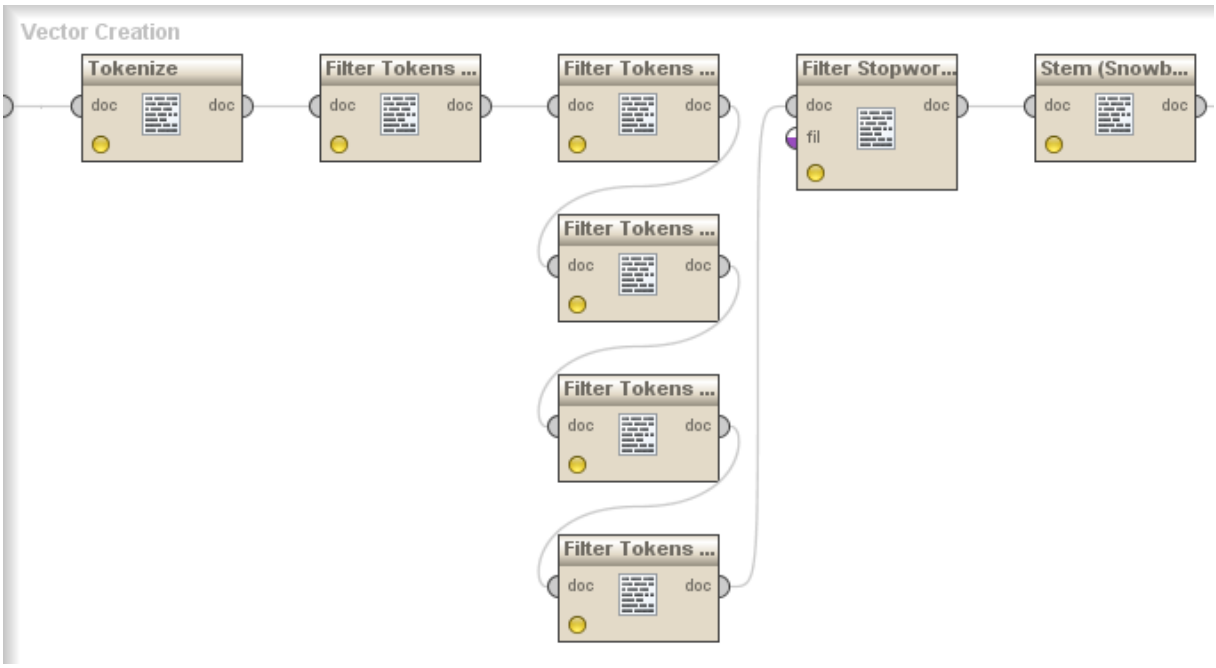

Figura 24: Processamentos do texto dos *tweets*.

O processo da Figura 24 começa com a transformação do texto em *token* através do operador **Tokenize**. Um *token* é a unidade básica de um texto, que no caso desse trabalho, consiste em uma palavra apenas. O segundo operador remove os *tokens* pelo tamanho, que foi configurado no mínimo 2 letras e no máximo 25 letras. Na terceira coluna há quatro operadores que removem os seguintes *tokens*:

- Contenha "@": Nomes dos usuários começam com "@", e usuário geralmente é a entidade no qual a opinião é emitida, e não uma forma de expressar um valor sentimental.
- Contenha "http": Links e Urls não são considerados porque, também, não expressam opinião.

- *Token* “RT”: O RT indica que a informação é replicada de outro usuário e nem sempre indica concordância com o *tweet* compartilhado, sendo comum um usuário emitir uma opinião sobre esse *tweet*, podendo ser tanto uma opinião positiva quanto uma opinião negativa.
- *Tokens* numéricos: números geralmente fazem parte do fato comentado e não denota nenhum tipo de valor sentimental. É importante ressaltar que os *tokens* que contêm números não são removidos, como, por exemplo, o *token* “<3”, muito utilizado para expressar sentimentos de valor positivo.

Ainda em relação ao processo da Figura 24, o operador **Filter Stopwords** realiza a remoção de *stop words* a partir de uma lista definida de palavras, e para finalizar é realizado o *stemming* dos *tokens* resultantes. *Stop word* e *stemming* são discutidos na seção 2.1, mas resumidamente uma *stop word* é uma palavra que não possui um valor para o processamento de texto, como artigos e preposições, e o *stemming* é um processo de reduzir uma palavra ao seu radical.

Após esse processamento, o vetor de representação é criado pelo algoritmo TF-IDF (*term frequency – inverse document frequency*) que é então enviado ao processo de treinamento (operador **Validation**, Figura 23). O processo de treinamento e validação é apresentado na Figura 25

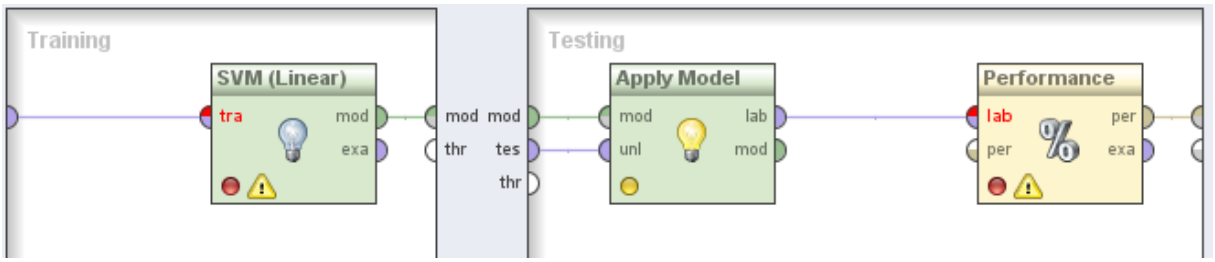

Figura 25: Treinamento do SVM.

De acordo com a Figura 25, o operador **Validation** é composto de dois processos: treinamento e teste. No primeiro processo é feito o treinamento de fato do SVM, recebendo como dados os vetores criados no processamento de *tweets*. Na etapa de teste é realizada uma amostragem dos dados de treinamento e, então, essa amostragem é aplicada ao modelo gerado, isto é, o SVM, tendo os resultados capturados pelo operador **Performance**, e o resultado é exibido na Figura 26.

accuracy: 72.55% +/- 5.51% (mikro: 72.55%)

	true pos	true neg	class precision
pred. pos	400	180	68.97%
pred. neg	44	192	81.36%
class recall	90.09%	51.61%	

Figura 26: Precisão, Cobertura e Acurácia do treinamento.

Conforme observado na Figura 26, dos 444 exemplos positivos (“true pos”), 400 foram classificados corretamente como positivos (“pred. pos”), com taxa de cobertura (*“class recall”*) de 90,09%. Dos 372 exemplos negativos (“true neg”), 192 exemplos foram classificados corretamente como negativo (“pred neg), o que resultou em uma taxa de cobertura de 51,61%. Como a taxa de cobertura consiste em *tweets* de uma classe que foram classificados corretamente, e a precisão consiste no acerto do classificado produzido, infere-se que o classificador gerado para realizar a Análise de Sentimento possuiu maior exatidão para identificar os *tweets* de valor sentimental positivo.

2º Caso

Para o segundo teste, que teve como o objetivo criar um classificador que retornasse 3 classes (positivo, negativo e neutro), foi utilizada a base dos *tweets* do teste anterior, mais 784 *tweets* de valor sentimental neutro. Como o SVM trabalha apenas com duas classes, foi necessária criar uma estratégia de classificação que utilize dois SVM para realizar a classificação. A Figura 27 contém uma representação da estratégia utilizada.

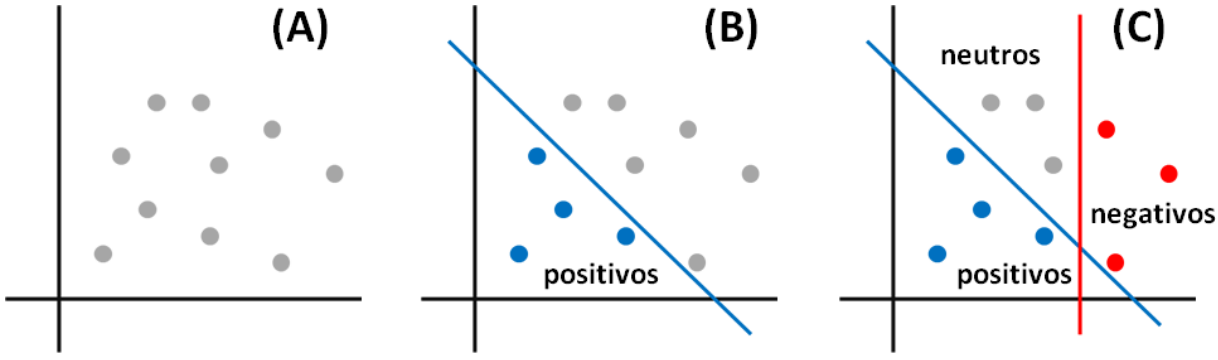

Figura 27: Estratégia para SVM de 3 classes.

Considerando um espaço de soluções, os exemplos apresentados no item (A) da Figura 27 precisam ser classificados em uma das três classes. O primeiro classificado SVM deve separar o conjunto entre os valores **positivos** e **não-positivos**, conforme apresentado no item (B). Em seguida é aplicado um novo classificador que separa o conjunto entre **negativo** e

não-negativo, apresentado em (C), e se um elemento não for classificado nem como positivo e nem como negativo, então ele é classificado como neutro.

A etapa de treinamento é bastante semelhante ao 1º caso, mas ao invés de definir diretamente a classe, foi criado um atributo para indicar se o *tweet* pertence à categoria positiva e um atributo para indicar se o mesmo pertence ou não a categoria negativa. Essa etapa do processo é apresentada na Figura 28.

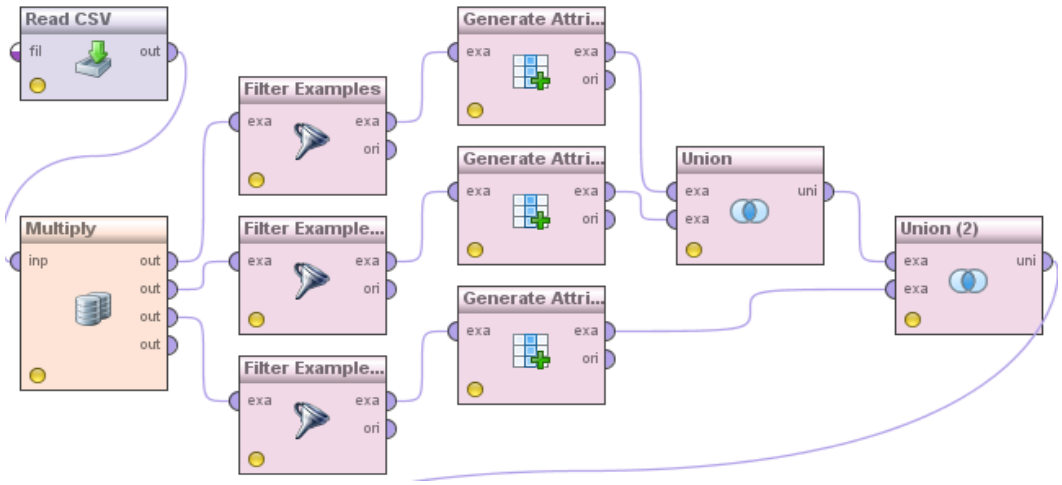

Figura 28: Etapa de geração dos atributos das classes.

Na parte do processo de treinamento apresentado na Figura 28, os operadores **Filter Examples** têm como objetivos separar os *tweets* de rótulo neutro, positivo ou negativo para que seja possível criar os atributos "isNeg" e "isPos" através do operador **Generate Attribute**. A etapa de treinamento é apresentada na Figura 29.

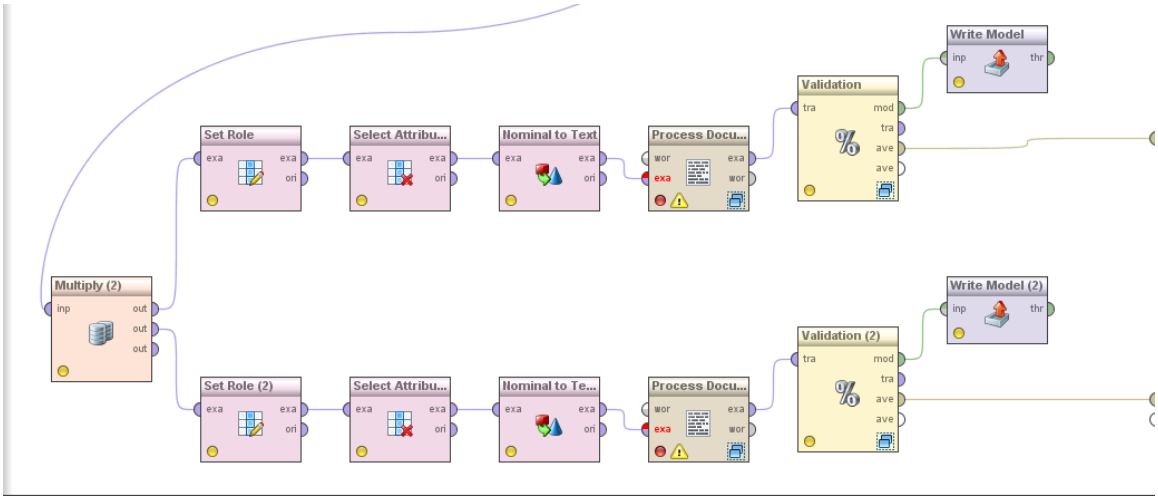

Figura 29: Treinamento dos dois classificadores.

Na Figura 29 há dois treinamentos em paralelo, o primeiro é o classificador de *tweets* positivos e o segundo é o classificador de *tweets* negativos. O processamento dos *tweets* é o mesmo apresentado na Figura 24, onde ocorre a geração de *token*, a filtragem de *tokens* e *stop words* e o processo de *stemming*. Ao fim do treinamento, ambos os modelos são armazenados.

Para testar os modelos, foi criado um processo que acessa o Twitter (através do operador **Twitter Reader**). Assim, foi possível classificar os *tweets* coletados a partir da aplicação baseada nos resultados dos dois modelos. Na Figura 30 é apresentada uma representação genérica do processo de aplicação dos dois SVM, que devido ao grande número de operadores utilizado dificultaria a visualização.

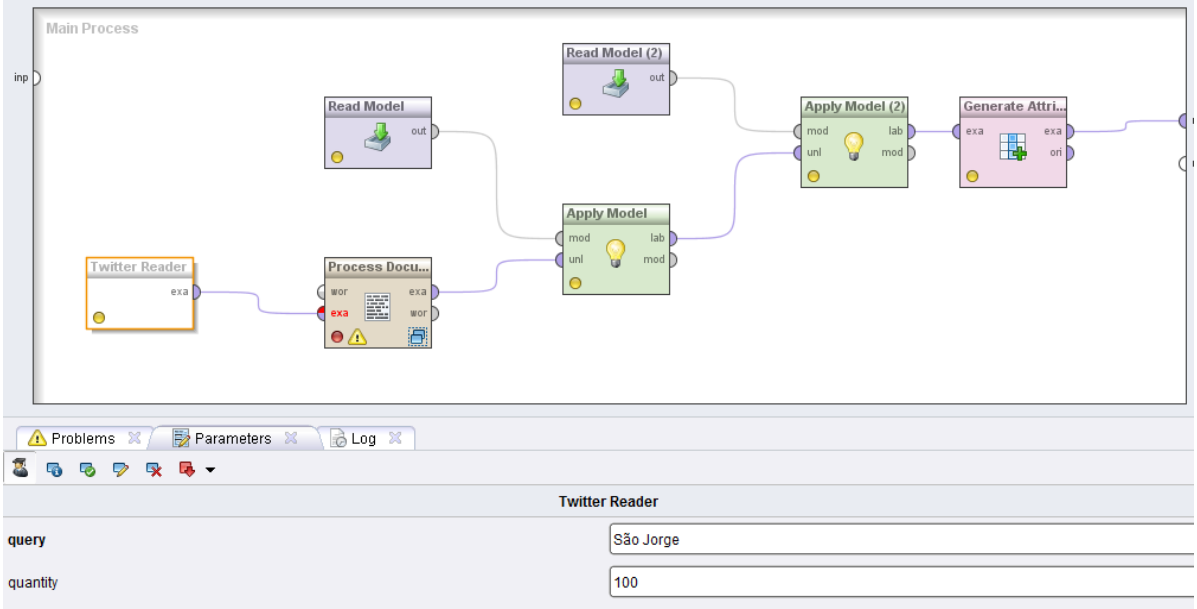

Figura 30: Aplicação dos dois SVMs (representação).

O teste da estratégia foi com a consulta do termo "São Jorge" limitando a 100 resultados (lembrando que é o máximo disponibilizado pelo API REST). Como não é possível trabalhar diretamente nos *tweets*, os mesmos são transformados para um formato que seja possível aplicar um modelo, um vetor de palavras, que é criado através do operador **Process Documents from Data**, que foi desenvolvido para este trabalho e apresentado na seção 4.2. O primeiro modelo (de valor sentimental positivo) é aplicado ao conjunto de *tweets* gerando o atributo "isPos" tendo os valores "true" ou "false" conforme o resultado da SVM. Em seguida é aplicado nesses *tweets* o segundo modelo, que irá gerar o atributo "isNeg", também com os valores "true" e "false". Para gerar o atributo "classe", que conterá uma das três classes possíveis, foi utilizado o operador **Generate Attribute**, tendo como base os valores dos atributos "isPos" e "isNeg", conforme definido na Tabela 4

Tabela 4: valor dos atributos para cada classe.

	isPos	isNeg
Positivo	True	False
Negativo	False	True
Neutro	False	False
Neutro	True	True

A Tabela 4 apresenta dois casos em que um *tweet* é classificado como neutro: quando os dois atributos possuem o valor "true" e quando os dois possuem o valor "false". A situação de um *tweet* ser classificado para as duas classes ocorre devido ao fato de que os dois modelos são aplicados no conjunto de testes, sem haver uma filtragem dos *tweets* que já foram classificados para uma classe. Mas se houvesse essa filtragem surgiria o seguinte questionamento: Qual SVM aplicar primeiro? E com essa resposta surge outro questionamento: Por que priorizar uma classificação em detrimento de outra?

Partindo da premissa que um *tweet* não pode possuir dois valores sentimentais, pois seria uma contradição, e que definir uma classificação padrão seria uma decisão aleatória, ficou definido que um *tweet* nessa situação (classificado como positivo e negativo) terá a classificação **neutra**. Isso porque se o elemento é classificado em uma classe, automaticamente exclui a outra, e como é possível não estar em nenhuma das duas categorias (positivo e negativo), é classificado conforme a regra de nenhuma das duas categorias, isto é, neutro. E na Figura 31 é apresentado parte do resultado do classificador criado, utilizando o operador "Twitter Reader" para acessar os *tweets*.

Ro...	classe	tweet
1	neutro	'São Jorge' mostra-se neste momento como Trending Topic em Manaus http://t.co/nNGecbq9gM
2	positivo	RT @eduardacortez: Vou acender vela pra São Jorge, a ele eu quero agradecer, vou plantar comigo-ninguém-pode, para que o mal não possa e
3	neutro	Hj fui p festinha da sao jorge...
4	neutro	Soltaram mts fogos aki perto de casa KkKk :) por causa diiiii São Jorge.... :D KkKk....
5	negativo	Ontem quase nenhuma guria daqui da rua saiu de casa, porque era dia de são jorge, imagina se ele vê elas e mata como se fossem dragões!
6	neutro	RT @rebeca182: tinha uma imagem de são jorge enorme na frente da Empório. Super trash :s
7	neutro	São Jorge: Comentários sobre o Dia de São Jorge que foi ontem, terça-feira. #TrendingsTop
8	neutro	Levantei e me deparei com a Lua de São Jorge maravilhosa. Povo abre a janela a noite esta linda! Vontade de nem dormir...no mundo da lua...
9	positivo	Vou acender vela pra São Jorge, a ele eu quero agradecer, vou plantar comigo-ninguém-pode, para que o mal não possa então vencer #SalveJ

Figura 31: Resultado do classificador de tweets para 3 classes.

Dos resultados apresentados na Figura 31, dois *tweets* foram classificados como positivos (os números 2 e 9) e um *tweet* foi classificado como negativo (numero 5). Da coleção de 96 *tweets* utilizados para esse teste o resultado da classificação foi:

- **positivo:** 10 *tweets*

- **negativo:** 1 *tweet*
- **neutro:** 85 *tweets*

O mesmo processo apresentado na Figura 30 foi aplicado para outro termo, mas utilizando o operador "Process Tweets from Strem", com as devidas alterações. O termo pesquisado foi o *trend topic* "Dona Xepa", tendo os resultados apresentados na Figura 32.

To gostando de Dona Xepa :)	positivo
dona xepa c cara de rebelde rsx	positivo
Arthur, brilhando em dona xepa .	positivo
Dona xepa Acabando cmgo :c	positivo
To adorando o papel da Rayana Carvalho em Dona Xepa	positivo
Dona xepa é chatinha, só vejo pelo arthur!	positivo
Ta o elenco de Rebelde quase td em Dona Xepa, plmdds.	neutro

Figura 32: Tweets obtidos pelo operador do Streaming API.

Ressalta-se que forma de acessar os *tweets* não interfere no resultado do classificador, e as considerações por qual API usar foram apresentadas nas seções 4.1 e 4.2, isto é, o limite de resultados e a quantidade retornada por requisição.

4.4. Aplicativo

Com o objetivo de demonstrar a viabilidade de aplicação comercial da Análise de Sentimentos desenvolvido por este trabalho, e bem como verificar a flexibilidade no uso da ferramenta, foi projetada e desenvolvida uma aplicação que se utiliza da Técnica de Análise de Sentimentos, mas sem a dependência de usar, diretamente, o RapidMiner, isto é, utilizando o RapidMiner como uma biblioteca de código. Além disso, o RapidMiner também foi empregado para elaborar o processo que foi utilizado pela aplicação.

A aplicação foi desenvolvida com base no diagrama apresentado na Figura 33.

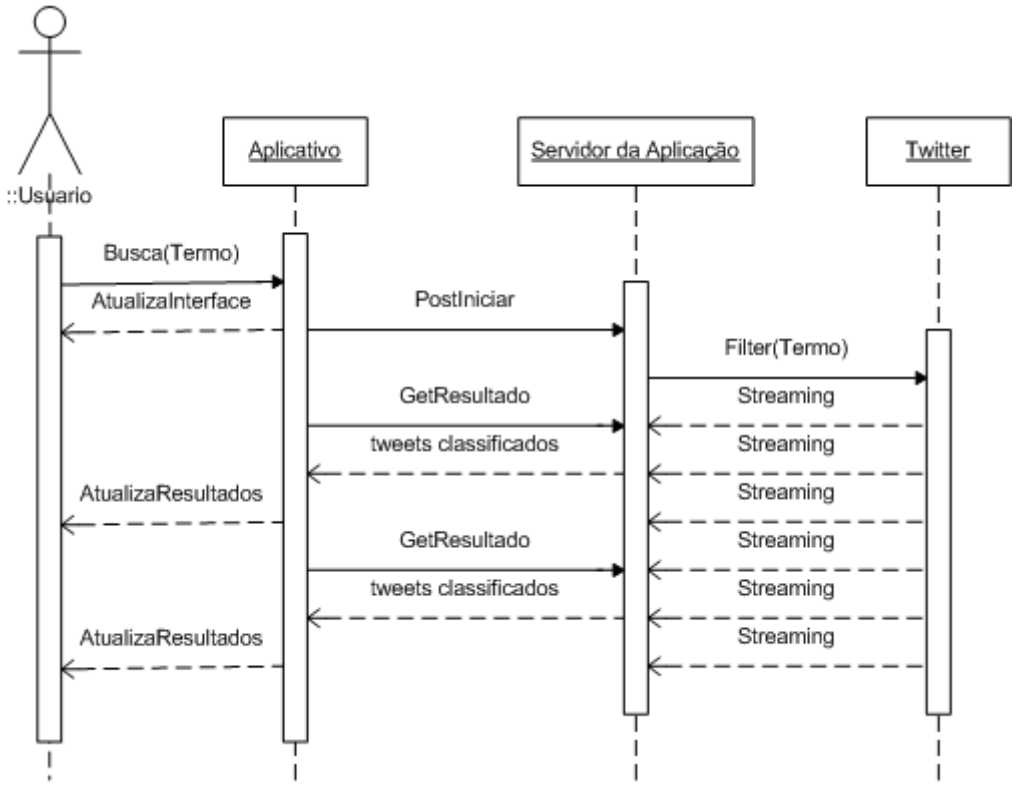

Figura 33: Diagrama de Sequência da Aplicação.

De acordo com a Figura 33, o processamento começa com o usuário informando o termo que ele deseja pesquisar no Twitter. Com essa ação o aplicativo informa ao servidor que deve ser iniciado o processo de Análise de Sentimento (PostIniciar), que se dá através de uma conexão com a *Streaming* API, que irá retornar os *tweets* que contenham o termo requisitado pelo usuário. Periodicamente, a aplicação irá verificar se há resultados no servidor (GetResultado), isto é, se há *tweets* classificados entre os valores sentimentais propostos (positivo, negativo e neutro). E conforme os resultados forem retornados, os mesmos serão exibidos ao usuário da aplicação.

A classe responsável por controlar as requisições do usuário (como iniciar o processo de Análise de Sentimentos) e enviar os resultados do servidor a aplicação (os *tweets* e seus respectivos valores sentimentais) é apresentada no Código-fonte 8.

```
21 class SentimentoController {
22
23     SentimentoService sentimentoService
24
25     def index() {
26         return
27     }
28
29     def iniciar(){
30         if(request.method=='POST'){
31             String termo=params.termo
32             if(termo==null || termo.size()==0){
33                 render "nothing"
34             }
35             sentimentoService.IniciarStreaming(termo)
36             render "OK"
37         }
38         else{
39             render "nothing"
40         }
41     }
42
43     def resultado(){
44         def tweets=Tweet.getAll();
45         def retorno=sentimentoService.AnalisarTweets(tweets)
46         tweets.each{
47             it.delete()
48         }
49         render retorno as JSON
50     }
```

Código-fonte 8: Controller responsável pela requisição da Aplicação.

No código apresentado no Código-fonte 8, a classe `SentimentoController` é responsável por gerenciar as requisições do lado do servidor (o objeto "Servidor da Aplicação" do diagrama apresentado na Figura 33), sendo que essas requisições (chamadas de *actions*) correspondem a métodos da classe. O objeto `sentimentoService` é o responsável pela lógica de negócio da aplicação. O método `index` (linha 25-27) apenas apresenta a interface através da qual o usuário irá informar o termo desejado. O método `iniciar` (linhas 29-41) corresponde a chamada **PostIniciar** do diagrama, que tem como responsabilidade verificar se foi informado um valor e iniciar o processo de *Streaming* de dados (linha 35), usando os conceitos apresentados na seção 4.1.2, que também foi utilizado para elaborar o operador "Process Tweets from Stream" apresentado na seção 4.2. E o método `resultado` (linhas 43-50), que corresponde a chamada **GetResultado** do diagrama, é o responsável pela etapa de classificar os *tweets* coletados até o momento. Primeiramente, são selecionados os *tweets* capturados em um dado tempo (linha 44), sendo aplicados ao método que irá executar o processo de Análise de Sentimento (linha 45). Após o fim do processo de Análise de Sentimento, os *tweets* classificados são enviados para a aplicação (linha 49). O método

`analisarTweets` (linha 45) realiza a classificação do *tweets* através do processo do RapidMiner apresentado na Figura 34.

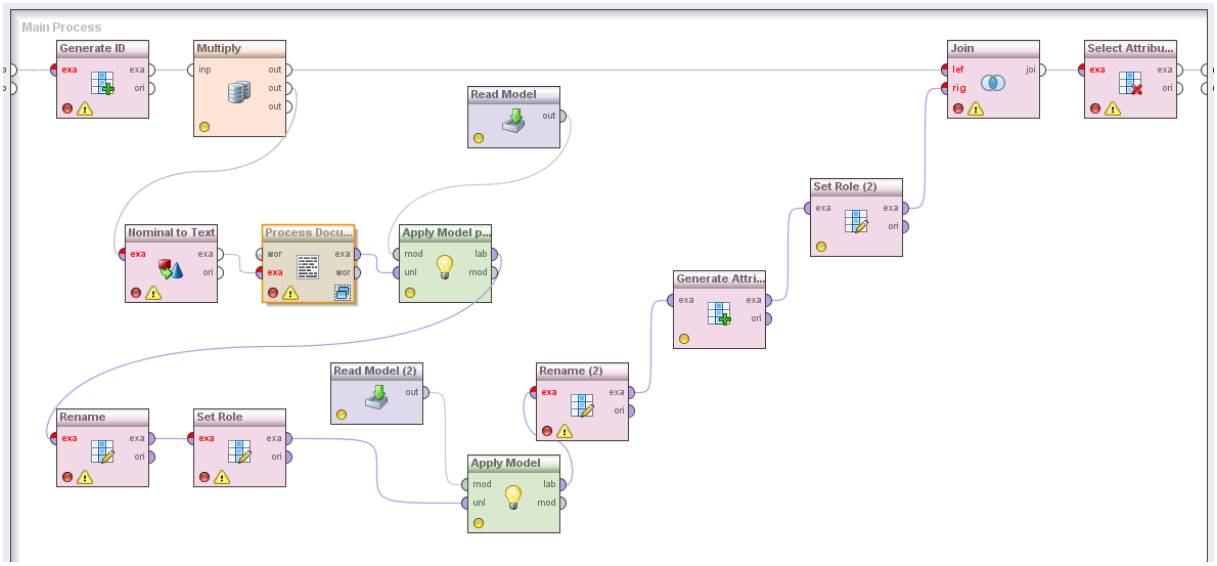

Figura 34: Processo de Analise de Sentimento (executado externamente).

Os três operadores principais do processo apresentado na Figura 34 são: “Process Documents from Data”, que realiza o pré-processamento dos *tweets* (comentado na seção 4.3, Figura 24), e os dois operadores “Apply Model” (coloração verde), que realizam as duas classificações na sequência que são utilizadas para criar a classificação final. Os dados de entrada desse processo são fornecidos durante a execução do processo pela aplicação, utilizando a API do RapidMiner, que carrega todos os operadores com base em um documento XML que descreve o processo como um todo.

A página em que o usuário acessa o sistema, nomeada de Klaĉo (se pronuncia “klatchô”, que significa “fofoca” em Esperanto), é apresentada na Figura 35.

Figura 35: Klaĉo, a aplicação cliente.

A interface apresentada na Figura 35 consiste em dois elementos: um campo de entrada, no qual o usuário especificará o termo desejado, e um botão de ação. Na Figura 36 é apresentado o resultado da classificação de *tweets* a partir da utilização do *Streaming* API.

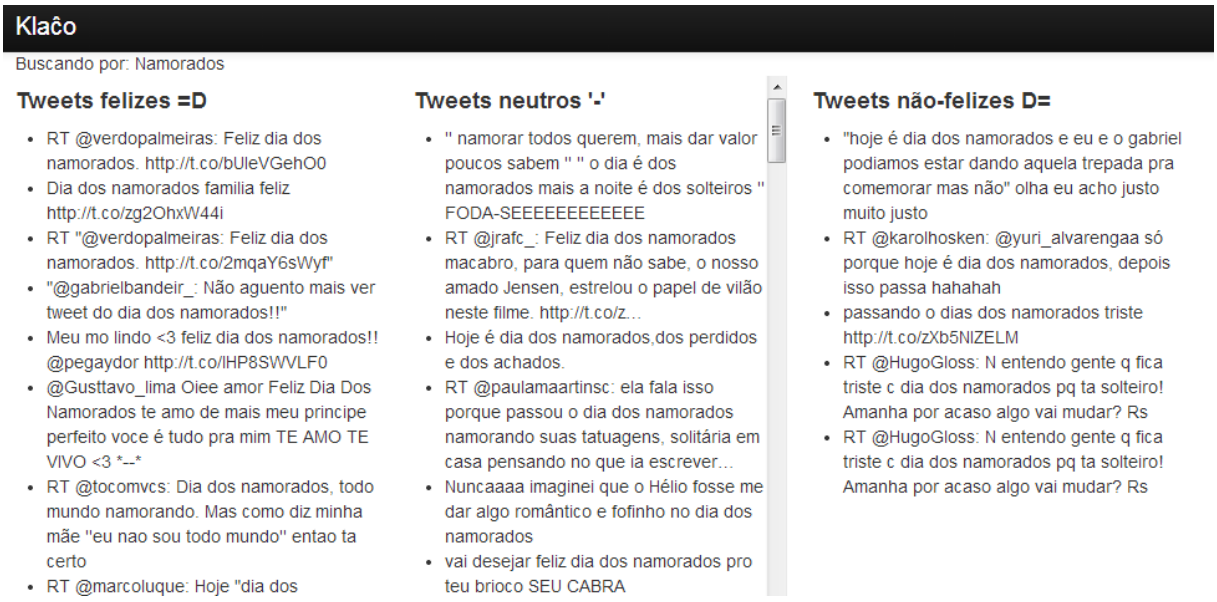

Figura 36: Resultado da Aplicação para o termo "Namorados".

Nos resultados apresentados na Figura 36, há três colunas, uma para cada valor sentimental: positivo, neutro e negativo. Essa classificação utiliza o processo realizado no segundo caso apresentado na seção 4.3, que rotula os *tweets* com um dos três valores sentimentais. É possível ver que muitas das classificações foram corretas, tais como o *tweet*

“Meu mo lindo <3 feliz dia dos namorados!!! [...]” que foi classificado como positivo e o *tweet* “passando o dias dos namorados triste [...]” classificado como negativo.

Na Figura 37 há outro resultado obtido da ferramenta, sendo a busca feito pelo termo “Feliz dia dos amigos”.

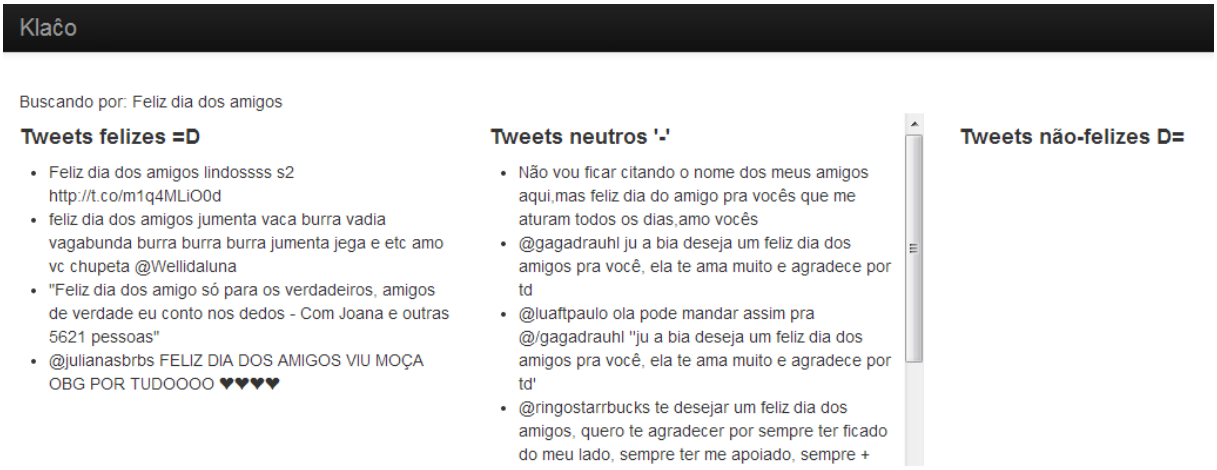

Figura 37: Resultado para o termo "Feliz dia dos amigos".

Nos resultado apresentado na Figura 37 há algumas situações peculiares, como por exemplo, o segundo exemplo de *tweets* positivo possui uma variedade de palavras que indicariam um sentido negativo, e de positivo há apenas a palavra “amo”, e do ponto de vista de um observador humano esse tweet também seria classificado como positivo. Há também classificações erradas, tais como o segundo exemplo de *tweets* neutros, que, do ponto de vista de um observador humano, seria classificado como positivo.

Na Figura 38 há o resultado da aplicação para o termo “#7DVDCalypso”.

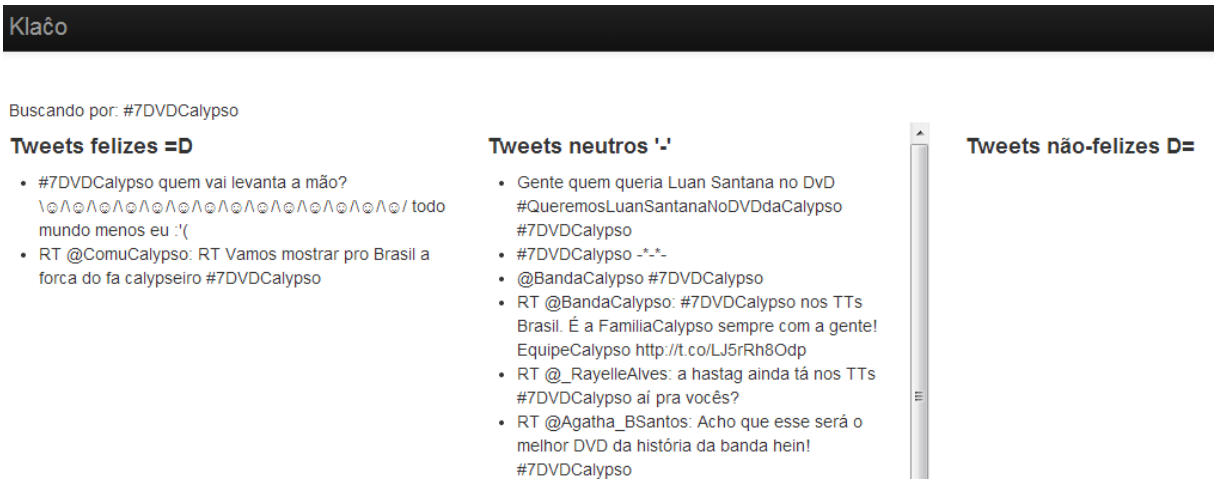

Figura 38: Resultado para o termo "#7DVDCalypso".

No primeiro exemplo apresentado na Figura 38, que foi classificado como positivo, é um caso um pouco complexo, pois indica que o usuário gosta da banda (pelos *emoticons* de braço levantados), mas não esta feliz com a situação (pela frase “todo mundo menos eu :′(”). Esse *tweet* foi classificado como positivo devido ao grande numero de *emoticons* positivos que encontra-se presente na frase. Em relação aos *tweets* neutros, o primeiro deles há menção do usuário ter interesse por outro cantor não mencionando nada em relação ao termo pesquisado. Já os 3 seguintes exemplos foram classificados corretamente como neutros pois um apresenta apenas um emoticon desconhecido da ferramenta, o outro apenas cita a *hashtag* utilizada (que é o termo pesquisado) e o ultimo desses apenas cita o fato de que a *hashtag* está nos *trends topics* e não apresenta nenhum sentimento relacionado.

O estudo da API do Twitter possibilitou o acesso aos dados do domínio, permitindo que a Análise de Sentimentos fosse efetuada pelo RapidMiner, diretamente ou através da aplicação criada. Para utilizar diretamente o RapidMiner, foi necessário criar dois novos operadores para realizar o processo de Análise de Sentimentos no Twitter. E ao adaptar esse processo, foi criada a aplicação que foi apresentada nesta seção.

5 CONSIDERAÇÕES FINAIS

O primeiro objetivo proposto pelo trabalho, que foi o estudo e a compreensão do que é a Análise de Sentimentos, bem como os conceitos relacionados, foi realizado durante as etapas iniciais deste trabalho, que juntamente com outros assuntos que foram necessários estudar, foi apresentado na seção 2.3. Durante esta etapa de estudo, percebeu-se o relacionamento do conceito de Análise de Sentimento, que até então era desconhecido, com conceitos mais conhecidos, como a tarefa de Classificação, apresentado na seção 2.2, que é utilização para realizar a Classificação de Sentimentos.

Inicialmente este trabalho não tinha o Twitter como foco, sendo utilizado um conceito genérico de domínio de aplicação. Independentemente do domínio que seria escolhido, alguns novos conceitos deveriam ser estudados para serem aplicados futuramente quando o domínio fosse escolhido. Foi estudado sobre Extração de Informação, que teve como objetivo permitir que os dados de um determinado domínio fossem aproveitados de forma mais eficiente em um processo seguinte, que neste caso trata-se da Análise de Sentimentos. E conforme foi descrito na seção 4.3, no processo de Análise de Sentimentos, os conceitos estudados permitiram que dados que não possuem valor para a Análise de Sentimentos fossem descartados, o que contribuiu para que o processo fosse otimizado.

Com os dados formatados, iniciou-se o processo de Análise de Sentimento a partir da utilização de um algoritmo de Aprendizagem de Máquina presente no RapidMiner. Para tanto, utilizou-se mais especificamente a técnica SVM na Classificação de Sentimentos. Com a utilização do RapidMiner, foi possível treinar e avaliar modelos até chegar ao classificador que foi apresentado neste trabalho.

Na etapa de aplicação da Análise de Sentimento no domínio, a escolha da rede social Twitter mostrou-se adequada ao contexto. Isso porque a limitação de caracteres dos *tweets* possibilitou que a opinião expressada, na maioria das vezes, fosse relacionada a uma única entidade, garantindo que os termos utilizados no *tweet* fossem opiniões ou fatos sobre um único objeto.

O tema Análise de Sentimento e os demais apresentados neste trabalho tiveram seus conceitos compreendidos tanto de um modo geral quando no contexto do trabalho em si. O ambiente de desenvolvimento utilizado, o RapidMiner, disponibilizou recursos para a continuidade do trabalho, que juntamente com o que foi apreendido sobre os documentos do

domínio, isto é, os *tweets*, permitiu que fossem produzidos os resultados apresentados, principalmente, nas seções 4.3 e 4.4. Com isso, é possível dizer que o objetivo deste trabalho, que é a aplicação da técnica de Análise de Sentimentos em um domínio através do RapidMiner, foi alcançado.

5.1. Trabalhos futuros

É possível avançar o trabalho nas seguintes situações: identificar a intensidade do valor sentimental classificado e utilizar mais dos elementos do domínio criado.

O possível valor sentimental de uma opinião pode ser compreendido como um intervalo de valor negativo até positivo, com o valor neutro entre eles. Como este trabalho utilizou esses valores como absolutos, é possível aprimorar o mesmo para lidar com situações entre determinados intervalos, possibilitando, por exemplo, a inferência de opinião "fortemente positiva" ou opinião "neutra tendendo ao positivo". Como uma hipótese para esse problema, pode-se desenvolver um classificador SVM de modo hierárquico, isto é, que separe, por exemplo, documentos positivos de documentos negativos, e para cada grupo desses, classifique a intensidade do valor sentimental dos documentos do conjunto.

Outra possibilidade de trabalho futuro é utilizar melhor os valores do domínio, como no caso do Twitter, os *tweets* que são compartilhados por outros usuários (o *retweet* ou RT). Como o trabalho utilizou apenas dois elementos da definição de opinião apresentada por Liu na seção 2.3, é possível trabalhar o elemento "tempo", isto é, verificar o momento no qual a opinião foi expressa ou *tuitada*. Assim, usando o contexto desse trabalho, poderia ser analisada a variação da opinião de uma entidade no decorrer do tempo.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ÁLVAREZ, Alberto Cáceres. **Extração de Informação de artigos científicos**: uma abordagem baseada em indução de regras de etiquetagem. 2007. 131 p. Dissertação (Mestrado em Ciência da Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC-USP), São Carlos, SP.

FELDMAN, R., SANGER, J. **The text mining handbook**: Advanced Approaches in Analyzing Unstructured Data. Cambridge: Cambridge University Press. 2007.

GRISHMAN, R. Information extraction: techniques and challenges. In: SCIE ´97: INTERNATIONAL SUMMER SCHOOL ON INFORMATION EXTRACTION. 1997. New York. **Anais...** New York: Spring-Verlag. 1997. P. 10-27.

GAIZAUSKAS, Robert, WILKS, Yorick. Information Extraction: Beyond Document Retrieval. **Journal of Documentation**. 1998, Vol. 54 Iss: 1, pp.70 – 105.

HOUAISS. Dicionário Houaiss da língua portuguesa. On-line. Disponível em: <http://houaiss.uol.com.br/>.

JOACHIMS, Thorsten. Text Categorization with Support Vector Machine: Learning with Many Relevante Feauture. ECML '98 Proceedings of the 10th European Conference on Machine Learning, **Anais…** p. 137-142. 2007.

KINTO, Eduardo Akira. **Otimização e análise das máquinas de vetores de suporte aplicadas à classificação de documentos**. 2011. 145 p. Tese (Doutorado em Engenharia Elétrica) – Escola Politécnica de São Paulo, São Paulo.

KIVINEN, J., WARMUTH, M., AUER, P. The perceptron algorthm vs. winnow**:** Linear vs. Logarithmic mistake bounds when few input variables are relevant. In: Conference on Computational Learning Theory. **Anais…** 1995

KUSHMERICK, Nicholas, THOMAS, Bernd. **Adaptive information extraction: Core technologies for information agents**. 2003. 33 p. Universität Koblenz-Landau, Koblenz, Alemanha.

LIU, Bing (2010). Sentiment Analysis and Subjectivity. In: INDURKHYA, N.; DAMERAU, F. J. (Eds.). **Handbook of Natural Language Processing**. Connecticut: Chapman and Hall/CRC. 2010.

LIU, Bing (2011a). Opinion Mining and Sentiment Analysis. IN: ___. **Web Data Mining**: Exploring Hyperlinks, Contents, and Usage Data. 2ª Ed. Berlim: Springer-Verlag, 2011, cap. 11, p 459-526.

LIU, B (2011b). **Sentiment analysis tutorial**. San Francisco, CA, USA. 08/08/2011. Palestra proferida no 25º Conferencia em Inteligência Artificial (AAAI-2011), material disponível em: <http://www.cs.uic.edu/~liub/FBS/NLP-handbook-sentiment-analysis.pdf>

LORENA, Ana Carolina, CARVALHO, André C. P. L. F. de. Uma Introdução às Support Vector Machines. **Revista de Informática Teórica e Aplicada**. On-line. V. 14, n. 2, p. 43-67. Disponível em: <http://seer.ufrgs.br/rita/article/view/rita_v14_n2_p43-67>

MITCHELL, T. **Machine Learning**. McGraw Hill. 1997. New York, USA.

NILSSON, Nils J. **Introduction to machine learning**. 1998. Stanford, CA, USA. Disponível em: <http://robotics.stanford.edu/people/nilsson/mlbook.html>

PAN, Lin. **Sentiment Analysis in chinese**. 2012. 46 p. Dissertação (mestrado em ciência da computação) – Brandeis university, Waltham, MA, USA.

PANG, Bo. **Automatic Analysis of Document Sentiment**. 2006. 126 p. Tese (Doctor of Philosophy em Ciencia da Computação) – Cornell University, Ithaca, NY, USA.

PANG, Bo, LEE, Lillian, VAITHYANATHAN, Shivakumar. Thumbs up? Sentiment Classification using Machine Learning Techiniques. In: Proceeding of EMNLP. **Anais…** p. 79-86. 2002.

POWERS, David M. N. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. **Technical Report SIE-07-001**. School of Informatics and Engineering, Flinders University Adelaide, South Australia. 2007

RUSSEL, Stuart, NORVIG, Peter. **Artificial intelligence**: a modern approach. Prentince-Hall, New Jersey. 1995

SEBASTIANI, Fabrizio. Machine learning in automated text categorization. **ACM Computing Surveys**, v. 34, n. 1, p. 1-47, 2002.

SILVA, Eduardo Fraga do Amaral e. **Um sistema para extração de informação em referências bibliográficas baseado em aprendizagem de máquina**. 2004. 94 p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Pernambuco, Recife, Pernambuco.

TWITTER. **Twitter Developers**. Disponível em: <https://dev.twitter.com/>. Acesso em: 24 mar. 2013.

ZAMBENEDETTI, Christian. **Extração de Informação sobre bases de dados textuais**. 2002. 142 p. Dissertação (Mestrado em Ciências da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre.