



LEANDRO MATIOLI SANTOS

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

**LAVRAS - MG
2010**

LEANDRO MATIOLI SANTOS

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

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Orientador:

D.Sc. Ahmed Ali Abdalla Esmin

Co-orientador:

D.Sc. André Luiz Zambalde

**LAVRAS - MG
2010**

LEANDRO MATIOLI SANTOS

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

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

APROVADA em ____ de _____ de _____

M.Sc. André Grützmann UFLA

M.Sc. Cristiano Leite de Castro UFLA

D.Sc. Ahmed Ali Abdalla Esmín

Orientador

Co-orientador:

D.Sc. André Luiz Zambalde

**LAVRAS - MG
2010**

RESUMO

Este trabalho teve como objetivo o desenvolvimento de um protótipo capaz de realizar mineração de opinião em textos de redes sociais, tendo como unidade-caso, o *Twitter*. Uma descrição detalhada dos conceitos associados ao campo de *web mining* e ao processo de mineração de textos foi apresentada, mostrando-se as técnicas atuais para a sua realização. Para a construção do protótipo, utilizou-se a linguagem Java, tendo o cuidado de realizar uma pequena modelagem *UML* do mesmo antes. O método de aprendizagem de máquina conhecido como *SVM* foi escolhido para realizar a classificação binária entre sentimentos positivos e negativos, que representa uma ação de mineração de opinião ou análise de sentimento. Para se representar os documentos de texto de maneira estruturada, optou-se pelo modelo conhecido como vetor de características, que é um abordagem estatística para análise de textos. Nos testes realizados, foi possível observar que o classificador treinado alcançou, em média, uma taxa de acerto de 80% nas classificações desejadas. Conclui-se que o sucesso foi obtido no protótipo criado, pois o mesmo foi capaz de realizar as tarefas requisitadas de maneira razoável.

Palavras-chave: Mineração de opinião. Análise de sentimento. Web Mining. Aprendizagem de máquina. Classificação.

ABSTRACT

This work aimed the development of a prototype that is able to perform opinion mining in texts from social networks, using Twitter as a study case. A detailed description of the concepts associated with the field of web mining and the process of text mining was presented, showing the current techniques for its realization. The programming language Java was used to build the prototype, but previously, a small UML modeling was done. The machine learning method known as SVM was chosen to perform the binary classification between positive and negative sentiments, which represents an action of opinion mining or sentiment analysis. To represent the text documents in a structured manner, it was opted to use the feature vector, which is a statistical approach to text analysis. In the tests executed, it was possible to observe that the trained classifier achieved an average accuracy of 80% on the desired classifications. It was concluded that success was obtained with the prototype created, because it was able to perform the tasks solicited in a reasonable way.

Keywords: Opinion mining; Sentiment analysis. Web mining. Machine learning. Classification.

LISTA DE ILUSTRAÇÕES

Figura 1 Estrutura da Mineração na Web. Fonte Carrilho Junior e Passos (2007)	14
Figura 2 Etapas do processo de Mineração de Textos. Adaptado de Aranha e Vellasco (2007)	16
Figura 3 Relação de co-citação entre páginas e a técnica de tunelamento. Fonte: Soares, Passos e Vellasco (2008).....	21
Figura 4 Representação atributo/valor. Fonte: Carrilho Junior e Passos (2007).	22
Figura 5 Representação <i>Bag-of-Words</i> . Fonte: Carrilho Junior e Passos (2007).	30
Figura 6 Demonstração das Listas Invertidas. Fonte: Carrilho Junior e Passos (2007).	31
Figura 7 Pontos representando duas classes no plano 2d.	39
Figura 8 Infinitas retas separam as duas classes. Qual é a melhor?	40
Figura 9 Reta que melhor separa as classes.	42
Figura 10 Estrutura de um objeto <i>JSON</i> . Fonte Crockford.	46
Figura 11 Estrutura de um vetor <i>JSON</i> . Fonte Crockford.	47
Figura 12 Estrutura de um valor <i>JSON</i> . Fonte Crockford.....	47
Figura 13 Estrutura de uma string <i>JSON</i> . Fonte Crockford.	48
Figura 14 Estrutura de um número <i>JSON</i> . Fonte Crockford.	49
Figura 15 Tela principal mostrando opções do menu Arquivo.....	55
Figura 16 Tela mostrando o seletor de arquivos após seleção da opção Abrir base de texto.....	57
Figura 17 Tela Idioma.....	58
Figura 18 Tela Idioma mostrando a barra de progresso.	59
Figura 19 Tela Remoção.....	60
Figura 20 Tela Quantidade desejada para treino e teste.	61
Figura 21 Tela Define Classificação.	62
Figura 22 Tela Treinamento.	63
Figura 23 Tela Saida Treinamento.....	64

Figura 24 Tela Saída Teste.....	65
Figura 25 Tela Saída Classificação.	66
Figura 26 Tela Resultado.....	67
Figura 27 Tela Amostra.	68
Figura 28 Tela Utiliza Modelo Treinado.....	69
Figura 29 Tela Converte <i>Json</i>	72
Figura 30 Tela Identifica Idiomas.	73
Figura 31 Tela Remove Mensagens Indesejadas.	75
Figura 32 Tela Define Coleta.	76
Figura 33 Tela Resultados Anteriores com combobox clicada.	78
Figura 34 Tela define banco de dados.	79
Figura 35 Resultado da classificações entre Neutro x Opinativo no caso do <i>Windows 7</i>	82
Figura 36 Tela contendo o resultado da classificação Neutro x Opinativo sobre a <i>Apple</i>	83
Figura 37 Classificação final entre Positivo x Negativo sobre a <i>Apple</i>	85
Figura 38 Distribuição obtida após classificação entre neutro e opinativo sobre o filme <i>Jackass 3D</i>	88
Figura 39 Distribuição entre mensagens com conteúdo positivo e negativo sobre o filme <i>Jackass 3D</i>	89

SUMÁRIO

1	INTRODUÇÃO	9
1.1	Contextualização e Motivação	9
1.2	Objetivos do Trabalho.....	10
1.3	Estrutura do Trabalho.....	11
2	WEB MINING E TEXT MINING	13
2.1	Web Mining	13
2.2	Text Mining	15
3	ETAPAS DA MINERAÇÃO DE TEXTOS	16
3.1	Etapa de Coleta	17
3.1.1	Crawler.....	18
3.1.2	Crawler Focado.....	19
3.2	Etapa de Pré-processamento	22
3.2.1	Tokenização	23
3.2.2	Redução do Léxico	24
3.3	Etapa de Indexação	27
3.3.1	Indexação Textual.....	28
3.3.2	Indexação Temática	31
3.4	Etapa de Mineração.....	31
3.5	Etapa de Análise	33
4	ANÁLISE DE SENTIMENTO / MINERAÇÃO DE OPINIÃO	34
4.1	Análise de Sentimento ou Mineração de Opinião.....	34
4.2	Dificuldades para a sua realização	35
4.3	Aplicações.....	36
5	APRENDIZADO DE MÁQUINA / SVM.....	37
5.1	Aprendizado de Máquina.....	37
5.2	SVM.....	38
6	MÉTODOS UTILIZADOS.....	44
6.1	Tf-Idf.....	44
6.2	JSON.....	46
7	METODOLOGIA	50
7.1	Tipo de pesquisa	50

7.2	Procedimentos metodológicos	50
7.2.1	Visão geral	51
7.2.2	Visão detalhada.....	52
8	INTERFACE E IMPLEMENTAÇÃO DO PROTÓTIPO.....	55
8.1	Abrir base de texto	56
8.2	Utilizar modelo previamente treinado.....	68
8.3	Converter base de texto JSON	70
8.4	Identificar idiomas de um arquivo	72
8.5	Remover mensagens não relacionadas.....	74
8.6	Realizar coleta no Twitter	75
8.7	Ver resultados anteriores.....	77
8.8	Definir banco de dados	78
8.9	Organização de arquivos e pastas	79
9	RESULTADOS E DISCUSSÃO	81
10	CONCLUSÃO	93
	REFERÊNCIAS BIBLIOGRÁFICAS.....	95

1 INTRODUÇÃO

Neste capítulo inicial, introduz-se de maneira concisa os principais conceitos e elementos que proverão a base para o desenvolvimento deste trabalho. Os objetivos e o modo como o trabalho está dividido também se encontram nessa seção.

1.1 Contextualização e Motivação

A *web* constitui atualmente o maior repositório de informações existente no mundo. Pessoas interagem todos os dias com uma enorme quantidade de dados e se perdem entre conteúdos diversos, sempre buscando encontrar o que realmente querem. A dificuldade está exatamente nesse ponto: como filtrar essas informações que correm em um fluxo constante? Como recuperar apenas o conteúdo que se deseja? Ou melhor, como resumir de maneira clara e representativa a imensa quantidade de dados encontrada? O desafio que todos os usuários enfrentam reside basicamente nesses pontos.

Têm-se várias formas dessas informações serem dispostas na *web*: blogs, redes sociais, fóruns, vídeos, imagens, *websites* (sendo estes dos mais variados assuntos), etc, todas contendo suas especificidades e maneirismos, necessitando de abordagens distintas para a análise de seus conteúdos. Este trabalho lida com textos presentes em redes sociais (mais especificamente, o *Twitter*), realizando uma análise da polaridade do sentimento contido na mensagem.

Mineração de opinião ou análise de sentimento é um ramo da mineração de textos preocupado em classificar textos não por tópicos, e sim pelo sentimento ou opinião contida em determinado documento. Geralmente associado à classificação binária entre sentimentos positivos e negativos, o

termo é usado de uma forma mais abrangente para significar o tratamento computacional de opinião, sentimento e subjetividade em textos, segundo Pang e Lee (2008).

Twitter é um serviço de *micro-blogging* que permite a postagem de mensagens (atualizações) de até 140 caracteres para que outras pessoas as visualizem. Atuando também como uma rede social, permite que um usuário siga outros usuários e receba em sua página inicial ou aparelhos móveis as atualizações dessas pessoas na ordem em que foram postadas. A troca de mensagens privadas entre usuários cadastrados também é possível, assim como respostas públicas direcionadas.

Lançado em 2006, ainda é uma rede recente e em expansão. Operando inicialmente apenas como um serviço de postagens, teve várias funções incorporadas à medida que os usuários criavam alternativas para suas necessidades como, por exemplo, a comunicação entre dois usuários através da citação da pessoa no início da mensagem com o símbolo @ antes do nome, indicando que aquela mensagem era direcionada àquele usuário.

Possuindo conteúdo imensurável e possibilidades ainda não exploradas, a *web* tem se mostrado como um excelente objeto de estudo e foco de pesquisas na área de mineração de textos. Essa quantidade imensa de textos torna impraticável a assimilação por uma pessoa de todas as páginas sobre um assunto específico, tornando imprescindível uma ferramenta simples capaz de realizar a “leitura” dessa gama de textos e informar ao usuário se aquela informação está falando positivamente ou negativamente sobre o assunto e onde este conjunto de dados se situa num medidor de polaridade (para citar apenas um modo de lidar com esses dados textuais).

1.2 Objetivos do Trabalho

O objetivo principal deste trabalho é desenvolver um protótipo para análise de opinião na *web*, com estudo de caso à partir do *Twitter*. Para atingir a meta estipulada, os seguintes objetivos específicos devem ser alcançados:

- Realizar um levantamento bibliográfico dos métodos a serem utilizados;
- Selecionar um termo de estudo e investigação, envolvendo o *Twitter*;
- Adquirir uma coleção de documentos para a realização do processamento desejado;
- Modelar o protótipo, construindo diagramas de classe e casos de uso;
- Implementar o protótipo, utilizando o modelo *MVC* (será explicado adiante nesse trabalho);
- Testar e interpretar os resultados obtidos à partir do protótipo em funcionamento.

A implementação realizada pretende prover uma base para a possível inclusão de novas funcionalidades. O intuito é abranger as diversas fontes onde textos podem ser coletados.

1.3 Estrutura do Trabalho

Para atender aos objetivos apresentados, têm-se, além desta Introdução, a seguinte distribuição de assuntos:

- Os Capítulos 2, 3, 4, 5 e 6 constituem o Referencial Teórico deste trabalho. No Capítulo 2 são definidos os conceitos de

mineração na *web* e mineração de textos. No Capítulo 3, tem-se a apresentação das etapas da mineração de textos (coleta, pré-processamento, indexação, mineração e análise) de forma detalhada. No capítulo 4 é mostrado os conceitos que envolvem a mineração de opinião ou análise de sentimento propriamente dita. No Capítulo 5 é dada uma definição do que é aprendizado de máquina e quais são seus tipos, além de uma explicação do algoritmo utilizado neste trabalho, o *SVM (Support Vector Machine)*. No Capítulo 6 são apresentados os métodos utilizados nesse trabalho, fornecendo uma explicação e contextualização dos mesmos.

- O Capítulo 7 descreve a Metodologia utilizada no trabalho, definindo o tipo de pesquisa e onde esta se situa. Os procedimentos metodológicos e tecnologia que constituem este trabalho também são discutidos neste capítulo.
- O Capítulo 8 mostra a interface do protótipo implementado e comentários sobre sua construção.
- O Capítulo 9 contém os resultados obtidos e discussões acerca do que foi atingido ou não com a pesquisa realizada.
- O Capítulo 10 apresenta as conclusões acerca da validade do trabalho realizado e possibilidades que ainda podem ser exploradas.

2 WEB MINING E TEXT MINING

Neste capítulo são apresentadas definições para os termos *web mining* e *text mining*, explicando de maneira sucinta do que são constituídas essas abordagens.

2.1 Web Mining

Hoje, ninguém questiona o valor da *web* como fonte de informações. Tem-se uma quantidade imensurável de páginas espalhadas por todo o globo, em várias línguas e contendo vários tipos de informações, sendo essas informações dispostas em dados não estruturados, como textos, vídeos, imagens, sons, etc. A necessidade de absorver essas informações é alta e sabendo-se o quão custoso é realizar tal tarefa através de meios não computacionais, pesquisadores desenvolveram o conceito de *web mining*, derivado da expressão *data mining*.

De maneira simplificada, *data mining* (ou mineração de dados), segundo Han e Kamber (2006), se refere à extração ou mineração de conhecimento através de grandes quantidades de dados. Uma definição semelhante seria que a mineração de dados “designa uma área de trabalho e investigação, pertencente à Inteligência Artificial, que tem como principal objectivo a descoberta de conhecimento, de estruturas e relações no seio dos conteúdos das Bases de Dados”, Costa Cordeiro (2003, p. 15). Como se trabalha com a *world wide web* e não com banco de dados, utiliza-se o novo termo *web mining* (ou mineração na *web*) para se referir a trabalhos realizados na *web*.

Basicamente, divide-se a mineração na *web* da seguinte forma:

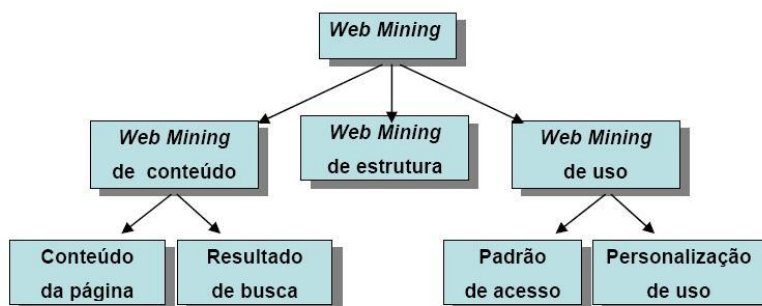


Figura 1 Estrutura da mineração na web. Fonte Carrilho Junior e Passos (2007)

Nos subseqüentes parágrafos tem-se uma explicação básica sobre cada subtópico da *web mining*, sem se estender até os nós folha da árvore demonstrada na Figura 1.

Web content mining (ou mineração do conteúdo da *web*) se preocupa com o conteúdo das páginas em si, desejando analisar textos, imagens, etc para extrair informações destes meios. Nesta área, também se localizam os motores de busca, utilizados para se encontrar páginas que o usuário deseja através de critérios passados pelo utilizador.

Web usage mining (ou mineração do uso da *web*) estuda a maneira como os usuários navegam, analisando o comportamento dessas pessoas na *web*. É verificada a ordem de navegação de uma pessoa, por exemplo, para analisar quais partes de um site precisam ser repensadas do ponto de vista da disposição de seus links, uma vez que se verifique que determinada informação não é muito acessada. Diversas aplicações podem ser pensadas utilizando a abordagem de *web usage mining*, como a sugestão de produtos que você possa gostar em um site de vendas, de acordo com a análise de sua navegação pelo site da empresa.

Por fim, tem-se a *web structure mining* (ou mineração da estrutura da *web*), que observa a estrutura dos hiperlinks presentes nas páginas *web*, verificando o relacionamento das mesmas. Verificam-se características como:

qual página é a que contém mais ligações com outras, qual é a mais referenciada em determinado contexto, etc.

Este trabalho se situa no nó à esquerda da árvore demonstrada na Figura 1, na subárea da *web mining* conhecida como *web content mining*, visto que será analisado o conteúdo presente na *web*, mais especificamente, textos presentes na rede social *Twitter*.

2.2 Text Mining

Text mining ou mineração de textos é uma área que tem como principal objetivo extrair conhecimento implícito de grandes quantidades de textos escritos em linguagem natural. Sua definição é bastante semelhante àquela dada à mineração de dados, sendo a grande e importante diferença o meio utilizado para minerar: na *data mining*, a mineração é realizada em uma base de dados.

A coleção de documentos em uma tarefa de mineração de textos é conhecida como *corpus*. Em geral, o *corpus* analisado é composto por uma grande quantidade de documentos, que são previamente coletados utilizando algum método definido pelo usuário.

3 ETAPAS DA MINERAÇÃO DE TEXTOS

A mineração de textos pode ser dividida em várias etapas, gerando um processo estruturado onde passos devem ser completados antes de se prosseguir para as próximas fases. Tal separação ainda não possui um padrão definido na literatura e a que será demonstrada nesse trabalho foi proposta por Aranha e Vellasco (2007).

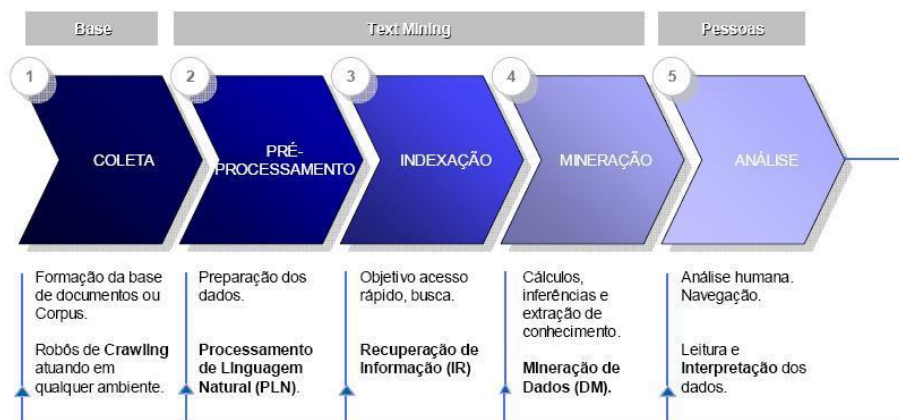


Figura 2 Etapas do processo de mineração de textos. Adaptado de Aranha e Vellasco (2007)

Primeiramente, tem-se a etapa de coleta de dados, onde textos são obtidos de diversas fontes para constituir o material a ser analisado no estudo em questão.

Em segundo vem a etapa de pré-processamento, na qual os textos são tratados para a futura representação computacional dos mesmos e análise. É uma etapa custosa devido à natureza dos dados sendo analisados, pois um trecho de texto possui muitas palavras que deverão ser processadas.

A etapa de indexação tem como finalidade representar os documentos de uma maneira inteligível para o computador e facilitar o seu acesso futuro, atuando como um índice.

Com os documentos devidamente representados, ocorre a etapa de mineração, que é a aplicação de algoritmos e técnicas para tentar extrair conhecimentos e padrões relevantes das massas textuais analisadas.

Por fim, após a mineração terminada, deve-se avaliar e interpretar os resultados obtidos. A fase de análise cuida exatamente desse ponto, podendo ser feita utilizando medidas quantitativas ou ser realizada por especialistas no assunto em questão. Finalizando a análise (ou em qualquer outra fase), pode-se retornar em algum ponto no processo de mineração de textos onde considerou-se que ali reside um problema que afetou os resultados finais e deve ser repensado.

Será explicado agora de maneira detalhada cada uma dessas fases apresentadas.

3.1 Etapa de Coleta

Como dito anteriormente, a mineração de textos é feita em cima de uma base textual coletada de algum meio, ficando disponível para posterior análise. De acordo com Carrilho Junior e Passos (2007), o meio onde se coleta pode ser dividido em 3 tipos:

- Arquivos encontrados no disco rígido
- Tabelas presentes em bancos de dados
- *Web*

Uma explicação mais detalhada será dada para cada um desses meios, demonstrando suas diferenças, os cuidados com a coleta, como é realizada, etc.

A primeira fonte se refere aos arquivos texto encontrados no disco rígido de um usuário de computador. Todos estão acostumados a lidar com esse tipo de massa textual no seu dia-a-dia, escrevendo e salvando documentos de trabalhos, por exemplo. Os cuidados na hora de realizar a coleta se resumem a não pegar arquivos de sistema ou similares, que em geral se apresentam na forma binária.

As tabelas de bancos de dados podem fornecer alguns campos contendo *strings* (tais como VARCHAR, MEMO) para a mineração de textos. Tais tabelas estipulam apenas a quantidade máxima de caracteres que se pode utilizar e podem prover uma boa base, dependendo do assunto em questão e da quantidade de informações armazenadas no banco.

Por fim, tem-se a *web*, que como demonstrado anteriormente, é onde existe a maior coleção de textos existente atualmente. Os problemas enfrentados residem em sua grande dimensão e nas diferenças entre os vários tipos de páginas, ou seja, “a heterogeneidade é o desafio predominante”, segundo Carrilho Junior e Passos (2007, p. 28). Intuitivamente, pode-se imaginar como é custoso percorrer a *web* em busca de páginas sobre um determinado tema e realizar o seu arquivamento de maneira manual. Para isso então, foram criadas ferramentas que executam todo o trabalho de navegar na internet e buscar o que se deseja. Na próxima seção, explica-se em detalhes como se dá esse tipo de coleta.

3.1.1 Crawler

Um método de coleta é a utilização de *crawlers*, que são robôs que percorrem a *web* de uma maneira específica, salvando as páginas visitadas de

acordo com as necessidades do usuário. Os *crawlers* possuem vários nomes, sendo alguns destes *web spider* ou *web robot*. Como demonstrado em Soares, Passos e Vellasco (2008), alguns autores conceituam de maneira diferente cada tipo de *crawler*.

Bots são programas que podem recuperar informações de locais específicos na Internet. *Spiders* são *bots* específicos que vão até a *Web* e identificam múltiplos sites com informações sobre um tópico escolhido e recuperam a informação (SOARES; PASSOS; VELLASCO, 2008, p. 77).

Apesar das sutis diferenças apontadas, é comum se falar apenas o termo *crawler*.

Sua execução é bem simples e consiste no fornecimento de um site raiz (ou vários) para que ele inicie a navegação, sendo que este site recebe o nome de semente (ou *seed*) na terminologia de recuperação de informação. A partir desse site inicial dado, o *crawler* realiza uma “leitura” da página em busca dos links contidos na mesma, e à medida que vai encontrando essas novas *urls*, vai armazenando cada um em uma fila que é responsável por gerenciar os sites que o *crawler* deve visitar. É importante perceber que a estrutura de dados do tipo fila foi escolhida para facilitar as operações de retirada de sites que devem ser visitados e a adição de sites a serem visitados, além de manter uma ordem de visitação coerente.

3.1.2 Crawler Focado

O *web crawler* realiza uma coleta automática de uma coleção de páginas na *web*. Existem *crawlers*, os chamados *crawlers* focados, que executam uma busca mais focada no assunto que o utilizador deseja.

Para manter o escopo de varredura dentro do domínio desejado um *crawler* focado pode contar com dois tipos de algoritmos ou até mesmo com alguma forma de combinação das duas idéias. São elas: *Web Analysis*, que julga relevância e qualidade das páginas apontadas por uma *URL* alvo; e *Web Search*, que determina a melhor ordem em que as *URL* alvo serão visitadas (SOARES; PASSOS; VELLASCO, 2008, pp. 85-86).

Uma breve explicação dos dois tópicos será apresentada a seguir, fornecendo maior detalhamento acerca dos possíveis métodos a serem utilizados para realizar a coleta do assunto desejado e tentando evitar a aquisição de muito ruído junto aos dados.

A *web analysis* pode ser dividida em dois tipos: análise baseada no conteúdo da página (chamada também de *content-based*, em inglês) e análise baseada na estrutura dos sites, nas ligações entre os mesmos (*link-based*, em inglês).

Análise de conteúdo utiliza algoritmos que verificam se dada página condiz com o que usuário deseja coletar. Diversos são os métodos utilizados com esse fim, como, por exemplo, identificar se determinadas palavras chaves estão contidas no texto da página sendo visitada ou análise de rótulos *html* que possam fornecer algum indicativo que o site em questão é de interesse do usuário.

Análise baseada nas ligações entre as páginas (*link-based*) é fundamentada no fato de que as páginas que estão ligadas possuem conteúdo relacionado. Conforme dito em Soares, Passos e Vellasco (2008), pode-se explicar que a pessoa que fez a página provavelmente colocará links que ele

julga ser relevante para o conteúdo da mesma ou que esteja relacionado aos temas tratados no site.

A *web search* deseja que a ordem de visitação das páginas de uma *crawler* seja a melhor possível. Para isso, deseja-se que a busca não se estenda para muito além da semente, pois quanto mais longe dela, mais ruído é adicionado ao *corpus* do assunto em questão sendo formado. Técnicas como *best-first* em oposição à *breadth-first* são utilizadas. Para maiores detalhes, consulte Soares, Passos e Vellasco (2008).

Existem também abordagens que desejam extrapolar a área de uma comunidade na *web* visando a obtenção de mais páginas relacionadas para a coleção de textos. Possuindo o nome de tunelamento, essa técnica permite que o *crawler* navegue por páginas não muito relacionadas (não sendo necessário armazená-las) a fim de se atingir um novo grupo de páginas que contenham textos relevantes para o assunto sendo coletado. Tem-se também a relação de co-citação, onde uma página e outra possuem essa relação caso sejam citadas por uma terceira página. Em geral, páginas ligadas dessa maneira demonstram tratar de assuntos semelhantes entre si.

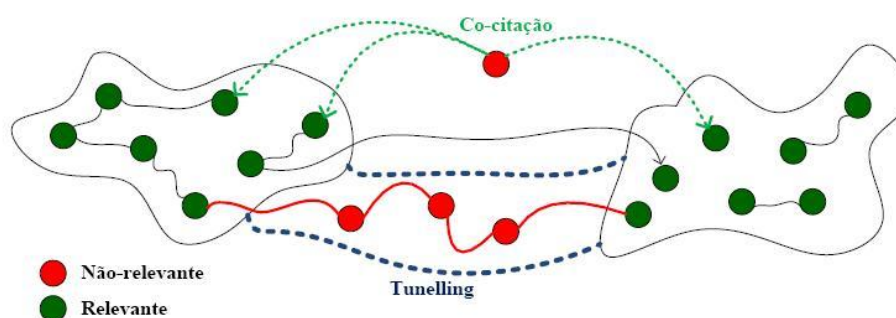


Figura 3 Relação de co-citação entre páginas e a técnica de tunelamento. Fonte: Soares, Passos e Vellasco (2008).

3.2 Etapa de Pré-processamento

Terminada a coleta, faz-se necessário pré-processar esses textos armazenados. A etapa de pré-processamento realiza toda a limpeza e inicia a adaptação do texto para uma futura representação mais estruturada, possibilitando seu tratamento pelo computador. De acordo com Carrilho Junior e Passos (2007), essa etapa é a mais onerosa devido a quantidade de técnicas diferentes que podem ser aplicadas, sendo que não existe uma técnica considerada superior que as outras, pois cada uma se adequa melhor em determinado contexto.

Deve-se então realizar vários tipos de filtros para “aumentar a qualidade inicial dos dados, aonde diversas técnicas podem ser aplicadas e até mesmo combinadas”, Carrilho Junior e Passos (2007, p. 30). Essa fase é umas das principais etapas em todo o processo de Mineração de Textos, pois uma abstração ruim do documento pode impedir que este seja analisado corretamente.

Como resultado final do pré-processamento, em geral, tem-se uma representação de atributos e valores associados a eles, formando um par atributo/valor. A figura 4 demonstra uma possível representação desse tipo, utilizando uma tabela.

	Atrib1	...	AtribN
Doc1	V11	...	V1N
...
Docm	Vm1	...	VmN

Figura 4 Representação atributo/valor. Fonte: Carrilho Junior e Passos (2007).

Nas próximas sub-seções, uma breve explicação de algumas técnicas para a realização do pré-processamento será dada.

3.2.1 Tokenização

Essa etapa consiste na separação de um texto em suas unidades mínimas, ou seja, deve-se separar cada palavra presente no texto, a pontuação, etc. Tais unidades mínimas são conhecidas como *tokens*. Existem abordagens que agrupam dois *tokens* para formar um *token* com significado agregado pois, por exemplo, o nome composto de uma entidade tem seu valor no seu conjunto de palavras, e não em cada palavra separadamente.

Um exemplo de um processo de tokenização básico:

Tem-se a seguinte frase: “Luiz Tatit é um grande compositor brasileiro!”. Separando em *tokens*, resultaria na seguinte distribuição: [Luiz] [Tatit] [é] [um] [grande] [compositor] [brasileiro] [!], onde cada termo entre colchetes representa um *token* presente na frase.

É importante perceber que o processo de tokenização pode parecer simples, mas sua dificuldade provém da separação de termos que não poderiam ser separados. Por exemplo, ao se separar o ponto [.] dos demais *tokens* no texto, pode-se perder o sentido de abreviação de palavras, números decimais ficariam sem sentido, etc..

Existem várias técnicas presentes no processo de tokenização que visam a melhor representação do documento de texto. Uma breve explicação será dada sobre algumas possibilidades de tratamento, mas esse trabalho irá lidar com a tokenização simplificada.

Conforme demonstrado em Carrilho Junior e Passos (2007), pode-se realizar a identificação de palavras combinadas, símbolos da internet, identificação de abreviações, entre outros.

Palavras combinadas são palavras que, como dito anteriormente, agregam um valor juntas mas não possuem o mesmo significado separadamente. Por exemplo, as palavras Coca Cola separadas não possuem a mesma relevância do que quando lidas em seguida. Deve-se, portanto, agrupá-las em um mesmo token.

Símbolos da Internet são referências a *emails*, *urls* de *websites*, endereços ips, etc. Como exemplo, tomemos um site: <http://www.tomze.com.br>. É fácil de perceber que a separação do ponto nessa *url* geraria um sequencia de tokens que não seriam uma referência ao site em questão.

Identificação de abreviações, como o próprio nome diz, visa identificar as palavras abreviadas presentes no texto, tomando o devido cuidado para que não se perca o seu sentido na transição para o formato de *tokens*.

3.2.2 Redução do Léxico

Não é difícil imaginar que a mineração de textos sofre com um problema relacionado ao meio que ela analisa: os documentos possuem um grande número de palavras diferentes e considerando-se que todas elas serão representadas como dimensões, tem-se um espaço com n dimensões onde n é um valor imenso.

Partindo desse problema, várias abordagens foram propostas com o objetivo de diminuir o léxico sendo tratado. Falando de outra maneira, pretende-se diminuir o número de palavras em um texto, mas sempre tomando o cuidado para que este não perca o seu sentido inicial ou não deturpe o que está sendo

falado. Serão citadas apenas três abordagens para a redução, sendo que existem várias na literatura.

3.2.2.1 Remoção de Stopwords

Existem certas palavras que não são consideradas relevantes para a informação geral de um texto. Tais palavras são conhecidas como *stopwords* na literatura de mineração de textos e quando agrupadas formando a lista de palavras sem importância para a análise, esse grupo recebe o nome de *stoplist*.

Em geral, essas palavras são as que aparecem com mais frequência no texto. Preposições, artigos, conjunções, pronomes e pontuação são os tipos de palavras que na maioria das vezes estão presentes em um sistema que remove *stopwords*.

Sua identificação pode ser feita de maneira manual ou automática. A definição manual de *stopwords* necessita que se conheça bem a massa textual caso deseje-se adicionar outras palavras à *stoplist* além dos artigos, pronomes, preposições, etc, da língua sendo analisada. A definição automática de *stopwords* ocorre através de um algoritmo que identifique as palavras que aparecem em grande quantidade nos textos.

3.2.2.2 Feature Selection

O processo de *feature selection* (ou seleção de características) utiliza várias métricas para tentar definir palavras que podem ser retiradas do texto sendo analisado, tomando o cuidado para que a informação contida neste não

seja perdida ou que a perda seja mínima, não afetando o resultado final da tarefa de mineração de textos.

As possíveis técnicas que podem ser utilizadas para identificar as palavras que representam o conteúdo do texto são: frequência de documentos, ganho de informação, informação mútua, estatística χ^2 . Para maiores detalhes em cada uma dessas métricas, consulte Carrilho Junior e Passos (2007).

Esse processo é de extrema importância devido ao problema relatado anteriormente da enorme dimensão que os textos podem atingir. Atenção especial deve ser dada a esta etapa quando se estiver lidando com textos grandes (o que não é o caso desta monografia).

3.2.2.3 Normalização

Normalização é a etapa que realiza a identificação de palavras que possuem algum tipo de relação entre elas e as agrupa. Pode-se verificar a constituição da palavra ou a presença de sinônimos, por exemplo.

Em geral, a aplicação de técnicas de Normalização introduz uma melhora significativa nos sistemas de Mineração de Texto. Esta melhora varia de acordo com o escopo, o tamanho da massa textual e o que se pretende obter como saída do sistema (CARRILHO JUNIOR; PASSOS, 2007, p. 40).

Dentre as possíveis técnicas para a realização da normalização das palavras no texto, destacam-se os processos de *stemming*, *lemmatization* e

identificação de sinônimos, hierarquias e relacionamentos associativos. A seguir, uma breve descrição de cada um.

O processo de *stemming* realiza a conversão das palavras presentes no textos para sua forma original. Termos no plural, verbos em diferentes tempos verbais, por exemplo, são transformados para representar apenas a sua palavra de origem. Existem diversos métodos para se fazer tal procedimento, cada um com suas particularidades, e abaixo está comentado brevemente apenas o método de Porter. Outros métodos presentes na literatura são o método de Lovins e o método do *Stemmer S*.

O método de Porter analisa as palavras presentes nos textos e as convertem para os seus respectivos radicais, fazendo com que palavras semelhantes se tornem uma palavra só. O radical é o elemento comum de palavras da mesma família. Cantar, cantor e cantoria, por exemplo, possuem o mesmo radical *cant*.

O processo de *lemmatization* consiste em transformar as palavras derivadas de outras na respectiva palavra de origem delas. Por exemplo, as palavras música, músicas e musical compartilham a mesma palavra de origem, música.

O processo de identificação de sinônimos, hierarquias e relacionamentos associativos nada mais é do que atribuir a termos diferentes (mas relacionados) uma mesma palavra que os represente. Utiliza-se em geral um *thesaurus*, que é um dicionário que contém palavras para representar termos que são sinônimos, abreviações, enfim, que possuam alguma relação.

3.3 Etapa de Indexação

Indexação é o processo responsável pela criação de estruturas auxiliares denominadas índices e que garantem rapidez e agilidade na recuperação dos documentos e seus termos (SOARES; PASSOS; VELLASCO, 2008, p. 42)

Isto se torna necessário pois, antes da etapa de indexação, não se dispõe de nenhum mecanismo que facilite a busca dentro da coleção de documentos sendo analisada.

Existem dois tipos de indexação, como visto em Soares, Passos e Vellasco (2008), sendo eles: Indexação Textual e Indexação Temática.

3.3.1 Indexação Textual

A indexação textual realiza a indexação da massa textual termo a termo de maneira automática, não necessitando de fontes externas. Abaixo, estão descritas duas formas desse tipo de indexação.

3.3.1.1 Representação do documento de texto

A mineração de textos necessita que os dados escritos sejam convertidos para uma representação que possibilite ao computador processá-los de alguma maneira, uma vez que este não será capaz de “ler e compreender” o texto como um ser humano o faz.

Segundo Carrilho Junior e Passos (2007), diversos são os modelos para se representar um documento. O mais comum presente na literatura é o Modelo

de Espaço Vetorial (em inglês, *Vector Space Model*), “que representa um documento utilizando uma abstração geométrica”, Carrilho Junior e Passos (2007, p. 48). Descrevendo de uma maneira mais formal, “documentos são representados como pontos em um espaço Euclidiano t-dimensional em que cada dimensão corresponde a um token do léxico”, Carrilho Junior e Passos (2007, p. 48). Tal representação também é chamada por alguns de vetor de características.

Neste trabalho, o vetor de características, conhecido como *feature vector* na língua inglesa, será o modelo utilizado para representar os documentos de textos em dados que o computador possa trabalhar.

Essa representação possui também o nome de saco de palavras (em uma tradução literal de *bag-of-words*) devido à sua natureza na qual “um documento é visto como um container de *tokens*, aonde a ordem e a ligação entre os *tokens* não tem nenhum valor para o sistema”, Carrilho Junior e Passos (2007, p. 48). Embora conhecidamente falha para identificar certas inflexões da língua escrita, “a codificação *bag of words* vem apresentando bons resultados na literatura, justificando a sua abordagem puramente estatística”, Carrilho Junior e Passos (2007, 48).

Como exemplo, tem-se abaixo a Figura 5 que contém um pequeno texto e a ideia de como é essa representação.

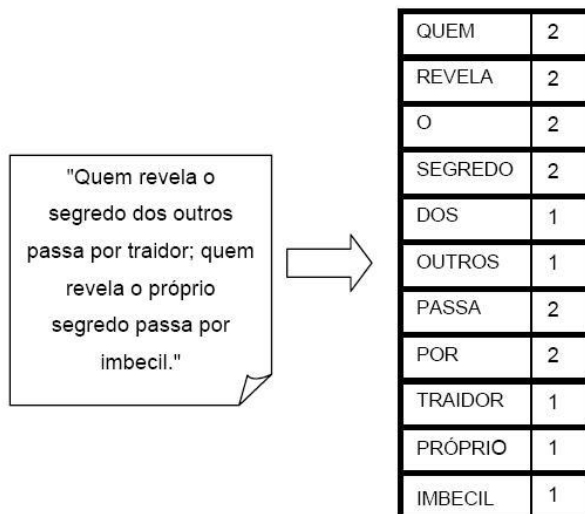


Figura 5 Representação *bag-of-words*. Fonte: Carrilho Junior e Passos (2007).

3.3.1.2 Lista Invertidas

Esse tipo de representação inverte o conceito padrão de documentos mostrando quais palavras estão presentes nele. Neste caso, as palavras é que são indicadores, tendo descrito em quais documentos elas estão contidas.

As listas invertidas possibilitam uma pesquisa mais rápida em um dado corpus pois se acessa o termo em questão diretamente pelo índice, ao contrário da representação do documento e palavras contidas nele, onde uma pesquisa de um termo precisa passar por todos os documentos da coleção dada.

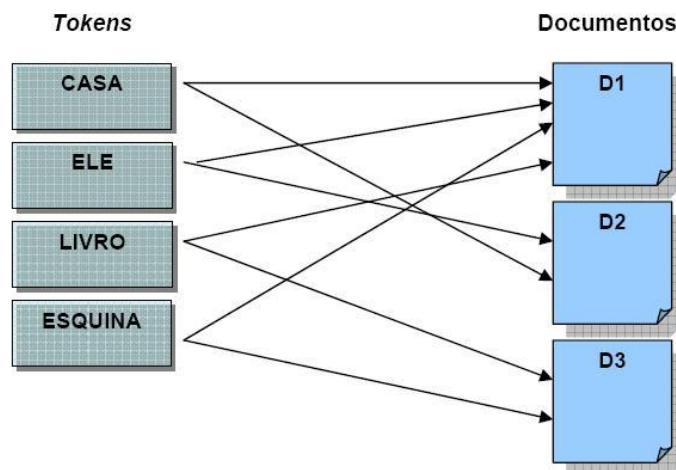


Figura 6 Demonstração das listas invertidas. Fonte: Carrilho Junior e Passos (2007).

3.3.2 Indexação Temática

A indexação temática, ao percorrer o documento, não indexa todos os termos como eles estão escritos. O que ela faz é utilizar um termo relacionado para indexar duas palavras diferentes, por exemplo, cão e cachorro seriam indexados como sendo apenas cão (ou cachorro, dependendo da preferência onde está sendo consultado).

Esse tipo de indexação necessita de um fator externo conhecido como *thesaurus*, que é um dicionário onde se tem as palavras e ao invés de seu significado, contém uma referência de palavras semelhantes a essa.

3.4 Etapa de Mineração

A quarta etapa é a da mineração propriamente dita. Nela é que são aplicados os algoritmos específicos para extrair o conhecimento desejado da coleção de textos.

Deve-se primeiramente decidir o que se deseja obter exatamente da massa textual. Algoritmos de classificação e clusterização podem ser aplicados, além de processos de sumarização e extração de características. Cada uma dessas possíveis abordagens é descrita resumidamente nos parágrafos abaixo, respectivamente.

Tendo-se uma coleção de textos, deseja-se classificar cada documento como pertencente ou não a uma determinada classe. O processo é semelhante à classificação realizada em bibliotecas e outras formas de catalogar, como, por exemplo, colocando livros em uma seção (classe) de romances. Pode-se classificar em quantas classes julgar-se necessárias, sendo que quanto maior o valor, tende-se a dificultar o trabalho do classificador.

A clusterização diz respeito ao agrupamento de textos em conjuntos de textos similares, chamados *clusters*. A complexidade desse processo reside no fato de não se conhecer, a priori, a quantidade de *clusters* que uma coleção de textos contém, tornando a tarefa não trivial.

A sumarização, como o próprio termo sugere, trata da tentativa de resumir o conteúdo de um texto de forma automática, visando obter um texto menor do que o original sem perder o seu significado principal.

A extração de informação visa construir dados estruturados à partir de dados não estruturados (forma que os textos, em geral, estão representados). Representa-se os dados textuais principais em uma tabela, por exemplo, podendo assim utilizar métodos de mineração de dados já conhecidos.

3.5 Etapa de Análise

Por fim, deve-se verificar se o conhecimento extraído é realmente válido e relevante para o assunto sendo estudado. Métricas quantitativas e análises qualitativas devem ser aplicadas para tal fim, como, por exemplo, a interpretação de um especialista no assunto.

Atenção especial deve ser dada à visualização dos resultados obtidos, pois muitas vezes a informação extraída ainda não está muito clara. Gráficos, tabelas e outras formas visuais devem ser exploradas com o intuito de facilitar a interpretação da saída de todo o processo de mineração de texto.

É importante notar que ao finalizar a etapa de Análise, pode-se optar por retornar em qualquer etapa anterior do processo definido. Através de critérios definidos nessa fase, pode-se verificar onde se encontra o problema e sua possível resolução em alguma etapa anterior.

4 ANÁLISE DE SENTIMENTO / MINERAÇÃO DE OPINIÃO

Neste capítulo, define-se de uma maneira mais formal os conceitos envolvidos na mineração de opinião e quais são as dificuldades encontradas para realizá-la de maneira satisfatória.

4.1 Análise de Sentimento ou Mineração de Opinião

Um ramo da mineração de textos atualmente em alta é o que lida com as opiniões expressas pelas pessoas em textos. Pode-se avaliar o que está sendo dito por alguém sobre determinado filme, livro, ou qualquer outro tópico através de análises computacionais. Tal tratamento é conhecido como mineração de opinião ou análise de sentimento

Desde há muito tempo, as pessoas buscam as opiniões de outras como forma de direcionarem suas compras e suas considerações acerca de determinado assunto. Com o advento da web como fonte de informações, grande parte dos usuários tem buscado nela textos que forneçam esse tipo de informação desejada (opiniões sobre algo). Como visto em Pang e Lee (2008), a maior parte das pessoas comentam que a pesquisa online sobre informações de produtos foram determinantes na decisão pela compra do mesmo.

Sabe-se que a quantidade de textos disponíveis na *web* é incomensurável e que nem sempre acha-se o que se deseja. Tendo em vista a crescente demanda por métodos que facilitem o processamento de grandes quantidades de texto, a necessidade de *softwares* que realizem análise de sentimento ou mineração de opinião tem aumentado.

Porém, a construção de uma aplicação que realize esse trabalho de maneira correta e rápida não é nem um pouco trivial e várias pesquisas na área ainda encontram-se em andamento.

4.2 Dificuldades para a sua realização

Diversas são as dificuldades encontradas na hora de realizar um processo de análise de sentimento. Basta imaginar o que se deseja fazer: tornar o computador capaz de interpretar um determinado documento, a emoção nele contida. Isso não é tarefa trivial nem para seres humanos, visto que diferentes visões e opiniões influenciam a maneira como cada um lê e entende um texto.

Comentando resumidamente os problemas para realizar tal interpretação, o primeiro empecilho encontrado reside na necessidade de filtrar textos que contém carga opinativa daqueles que são objetivos. Dependendo do meio onde se coleta, essa tarefa pode ser minimizada, pois um site de *reviews* conterà apenas opiniões de usuários, por exemplo.

Outra dificuldade está na forma do documento ser disposto. Pode-se ter sites que utilizam medidores como notas ou outras avaliações calculáveis e estas serviriam como indicadores do sentimento global exposto no texto associado. O problema é justamente analisar textos livres, que não contém esse tipo de ajuda.

A representação dos resultados obtidos também constitui um problema, caso se deseje informações adicionais além da quantidade que foi classificada como pertencente a uma classe de opiniões positivas ou negativas (tarefa dentro da mineração de opinião conhecida como *sentiment polarity*). Pode-se desejar, por exemplo, visualizar trechos mostrando onde estão ocorrendo divergências de opinião entre os usuários e/ou trechos mostrando as partes chave do sentimento do usuário acerca do assunto sendo analisado.

4.3 Aplicações

Um sistema para análise de sentimento ou opinião pode ser construído para realizar diversas funções, cada uma necessitando de um enfoque diferente, dependendo do que se deseja realizar.

Um tipo de aplicação que logo vem à mente é a utilização no campo dos negócios. Uma empresa deseja obter feedback sobre seus produtos, saber como está a recepção de um lançamento, investigar opiniões relacionadas à concorrência, entre outros objetivos que podem ser imaginados. Através dos dados obtidos, pode-se elaborar novas estratégias de marketing, propor inovações nos produtos ou processos da empresa, etc.

Sites que agregam análises (*reviews*) de produtos são bastante conhecidos dos usuários da *web* e considerados muito úteis por quem os acessa. Uma possível aplicação para um sistema de mineração de opinião seria a coleta de maneira autônoma de *reviews* pela *web* e posterior sumarização da opinião geral contida na coleção de textos formada.

Outras aplicações também podem ser imaginadas, como detecção de mensagens agressivas (ou contrárias às políticas do meio onde foi postada) e futuro bloqueio desses tipos de mensagens, melhorias em sistemas de recomendação (não recomendar produtos com muitas opiniões negativas, por exemplo), dentre várias outras funcionalidades.

5 APRENDIZADO DE MÁQUINA / SVM

Neste capítulo, apresenta-se a definição e descrição do que é aprendizado de máquina e do algoritmo de treinamento supervisionado utilizado neste trabalho, o *SVM*.

5.1 Aprendizado de Máquina

Aprendizado de máquina ou aprendizagem de máquina é uma área da Inteligência Artificial que estuda algoritmos que possibilitam ao computador perceber e aprender padrões existentes em um determinado conjunto de dados. Uma vez determinado esses padrões, os algoritmos podem então ser usados para realizar previsões acerca de novos dados desconhecidos. Tal aprendizado pode se dar de maneira supervisionada ou não-supervisionada, em geral.

O aprendizado supervisionado requer que, através de dados de treinamento fornecidos como entrada, seja apreendida uma função. Essa função, no caso do problema de classificação de textos (que é o objeto de estudo deste trabalho), é conhecida como função de classificação e é responsável por determinar a qual classe pertence determinado documento desconhecido. Para o algoritmo “aprender” essa função no caso dos dados serem texto, é necessário que seja fornecido uma relação de documentos classificados manualmente por uma pessoa especialista no assunto.

O aprendizado não-supervisionado não necessita de dados de treinamento previamente classificados. Ele analisa os dados e induz algo sobre eles, geralmente realizando agrupamentos. Tal método não será utilizado nessa monografia.

5.2 SVM

Este trabalho utiliza o algoritmo de aprendizado de máquina supervisionado conhecido como *SVM* (sigla para *Support Vector Machine*). Ele foi escolhido por ser considerado um dos melhores e mais utilizados métodos de classificação de textos atualmente, como mostrado em Zaghoul, Lee e Trimi (2009).

Demonstrado por Cortes e Vapnik (1995) e utilizado pela primeira vez para o problema de classificação de textos por Joachims (1998), seu funcionamento, de uma maneira simplificada, se dá da seguinte maneira: o método consiste em representar cada documento por um ponto ou vetor em um espaço t -dimensional e traçar um hiperplano que separe da melhor forma possível as duas classes de documentos em questão, i.e., deve-se maximizar a distância do hiperplano e dos elementos (documentos) de borda das classes, que são os chamados *support-vectors*.

Para melhor entendimento, um exemplo: suponha um problema simples de classificar uma quantidade x de documentos. Deseja-se saber se um determinado documento pertencente à coleção dada é sobre o assunto A ou se é sobre o assunto B (classe a e classe b, respectivamente). Considerando que seria possível representar esses documentos em um plano bidimensional e que esses dados fossem linearmente separáveis, tem-se uma distribuição viável disposta da seguinte maneira (Figura 7).

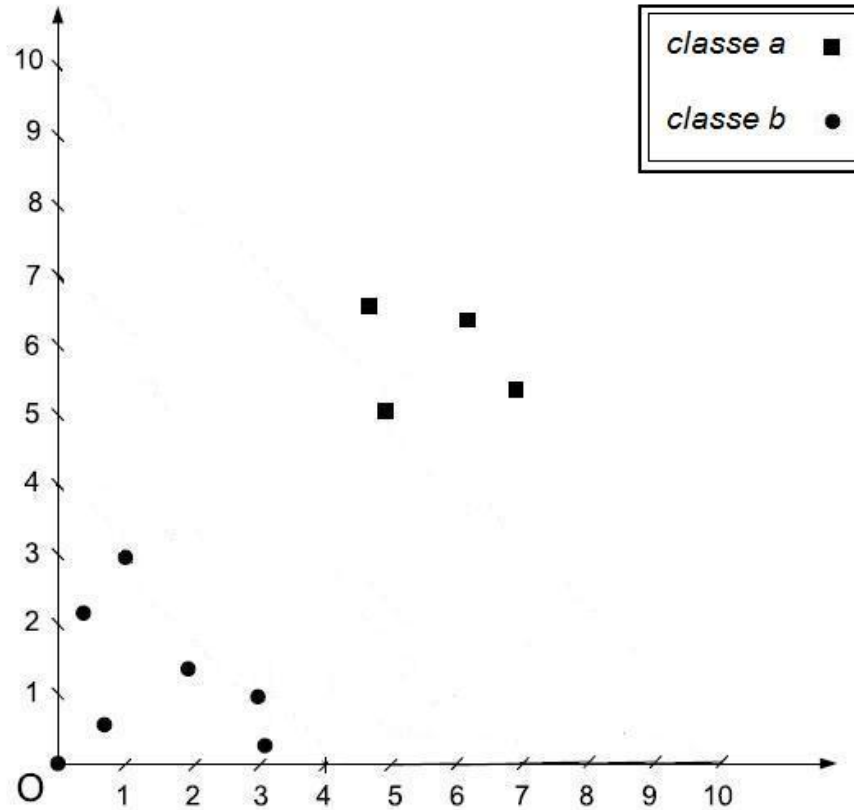


Figura 7 Pontos representando duas classes no plano 2D.

Como dito anteriormente, o método *SVM* vai tentar traçar uma reta que separe os dados nas duas classes mencionadas. Porém, sabe-se que a quantidade de retas que podem ser traçadas separando as duas classes é infinita e todas realizam uma classificação correta para os dados apresentados (situação demonstrada na Figura 8). Porém, deseja-se aquela que tem a maior capacidade

de generalização para dados desconhecidos. Intuitivamente, é a reta que possui a maior margem (distância).

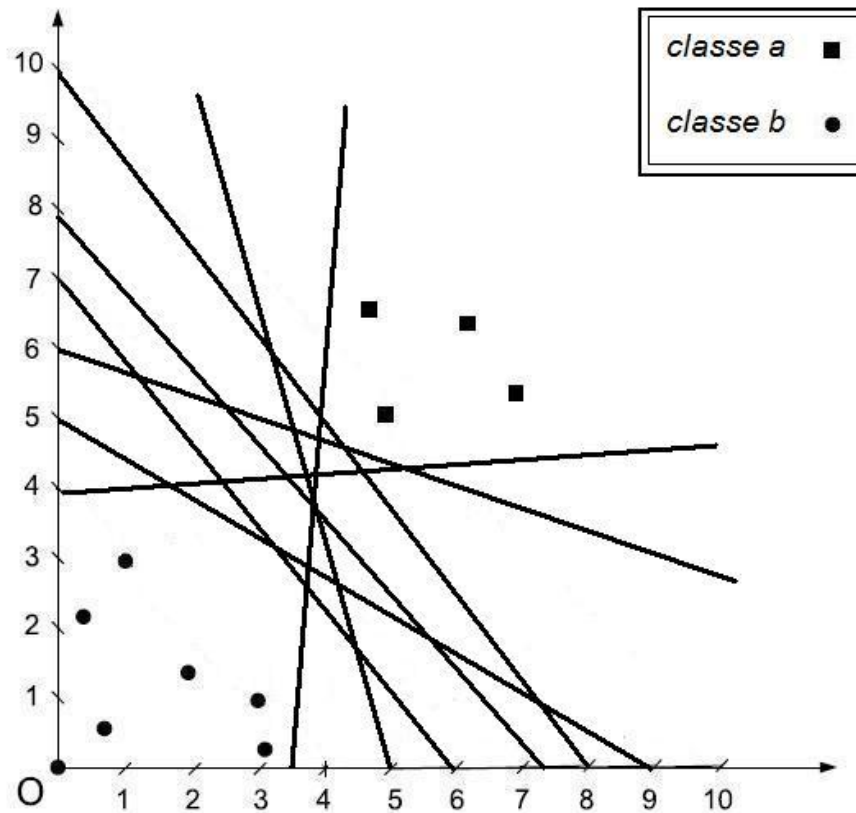


Figura 8 Infinitas retas separam as duas classes. Qual é a melhor?

Portanto, necessita-se identificar qual reta é essa. Como visto em Han e Kamber (2006), o SVM aborda esse problema tentando achar a reta com a maior margem, isto é, o *maximum marginal hyperplane (MMH)*.

Lembrando que o exemplo em questão utiliza apenas duas dimensões e por isso, fala-se em buscar a melhor reta. Se fosse um espaço tridimensional, desejaria-se encontrar o melhor plano. Generalizando para um espaço n -dimensional, o termo correto é hiperplano e seria este que se estaria procurando o melhor. Para o problema de classificação de textos, o problema é de ordem n (sendo n o número de palavras que representa o texto) e portanto, busca-se o hiperplano que melhor separa as classes de documentos.

Sendo assim, o *SVM* irá encontrar a reta que possui a maior margem com relação aos elementos conhecidos como *support-vectors*, que são pontos que estão igualmente distantes do plano que separa as classes. Tal reta pode ser vista na Figura 9.

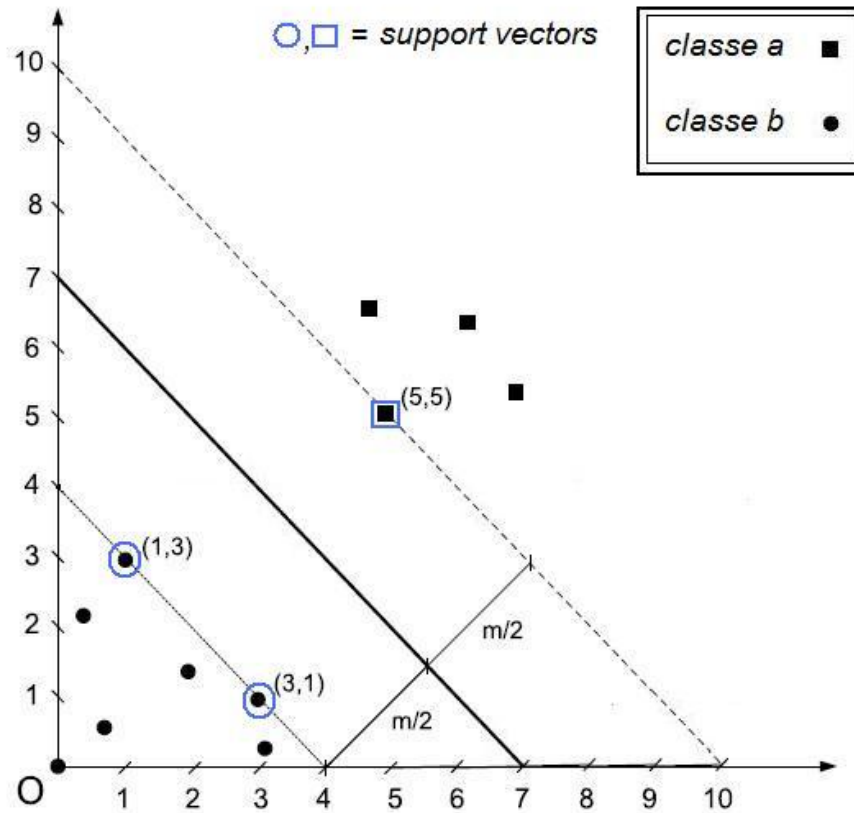


Figura 9 Reta que melhor separa as classes.

Como o SVM realiza os cálculos para encontrar esse hiperplano e os seus detalhes internos fogem do escopo deste trabalho e não serão explicados aqui. Para informações detalhadas, consulte Han e Kamber (2006).

Para o problema de classificação de dados não linearmente separáveis, que é o modo como a maioria das aplicações reais se apresentam, deve-se

extender a ideia do *SVM* para problemas lineares. Segundo Han e Kamber (2006), essa tarefa pode ser dividida em dois passos. No primeiro, transforma-se os dados de entrada originais em um espaço dimensional maior usando um mapeamento não-linear. Tendo os dados transformados, o segundo passo procura por um hiperplano linear que separe os dados nesse espaço dimensional maior da melhor maneira possível (*MMH*). Esse *maximal margin hyperplane* encontrado no novo espaço corresponde a um hiperplano não-linear que separa os dados no espaço dimensional original.

Novamente, os detalhes de como isso é feito não constam nessa monografia pois não é o foco da mesma. Aos interessados, recomenda-se a leitura das referências comentadas anteriormente.

6 MÉTODOS UTILIZADOS

Este capítulo trata especificamente dos métodos utilizados nesse trabalho, fornecendo uma contextualização detalhada sobre cada tópico abordado.

6.1 Tf-Idf

Sendo a representação por vetor de características escolhida, necessita-se de uma forma de colocar pesos em cada *token* presente no vetor. A maneira mais fácil é a mostrada anteriormente, na Figura 5, onde o peso corresponde a quantidade de vezes que a palavra apareceu no texto. Esse modo recebe o nome de *term frequency*. É importante lembrar que nessa representação, a ordem dos termos não é guardada, salvando-se apenas o valor de ocorrência deles.

Como visto em Manning, Raghavan e Schütze (2009), tem-se então que um documento com a frase “Maria gosta de João”, nesta abordagem, é igual a um documento contendo a frase “João gosta de Maria”. Pode-se imaginar, de maneira intuitiva, que os documentos contendo um saco de palavras similar terão conteúdo similar, apesar de possuírem significados diferentes.

O problema de se utilizar essa representação reside no fato de que todos os termos são considerados igualmente importantes. Sabe-se que certos termos não possuem um fator discriminante alto. Por exemplo, uma coleção de textos falando sobre a indústria musical provavelmente conterá a palavra música em praticamente todos os documentos.

Necessita-se então de um meio para atenuar a influência de termos que ocorrem com muita frequência em uma dada coleção de textos. Utiliza-se para

tal fim a frequência inversa do documento (*inverse document frequency*), demonstrada na fórmula abaixo, onde N é o número de documentos do corpus, t é o termo em questão e df_t é a frequência do documento baseada no termo t (ou seja, o df_t é a quantidade de documentos que contém o termo sendo analisado).

$$idf_t = \log \frac{N}{df_t}$$

Por fim, realiza-se então a multiplicação dos dois pesos, criando o peso composto $Tf-idf_{t,d}$ (*term frequency-inverse document frequency*), onde $Tf_{t,d}$ é a frequência de ocorrência de um termo t em um dado documento d e Idf_t é a frequência inversa do documento de um termo t . A fórmula encontra-se representada abaixo.

$$Tf-idf_{t,d} = Tf_{t,d} \times Idf_t$$

Onde o $Tf-idf_{t,d}$ coloca para o termo t presente no documento d , um valor, como visto em Manning, Raghavan e Schütze (2009):

- alto quando um termo ocorre várias vezes numa quantidade pequena de documentos, levando esses documentos a terem alto poder discriminante.
- baixo quando o termo ocorre poucas vezes em um documento, ou ocorre em muitos documentos, sendo assim, um termo com uma relevância reduzida.
- baixíssimo, quando o termo ocorre praticamente em todos os documentos.

Pode-se perceber os efeitos que a multiplicação do peso idf causa na pontuação final de um dado *token* e a influência deste para fornecer uma maneira coerente de representar a importância de cada termo em cada documento.

6.2 JSON

JSON (JavaScript Object Notation) é um formato para intercâmbio de dados considerado simples tanto para a leitura e escrita humana quanto para a realização das mesmas coisas de maneira computacional. É definido em formato texto e está disponível para uso em várias linguagens, como C, C++, Java, Perl, Python, etc.

Sua constituição se dá em duas partes: uma sequência de pares nome/valor e uma sequência ordenada de valores. Essas são estruturas básicas suportadas na maioria das linguagens de programação (objetos, registros, dicionários, etc, correspondem à sequência de pares nome/valor e vetores, listas, etc, correspondem à sequência ordenada de valores).

Demonstrando a estrutura do *JSON*, como visto em Crockford:

- um objeto (Figura 10) é um conjunto não-ordenado de pares nome/valor. Ele começa com o caractere { (chave aberta) e termina com o caractere } (chave fechada). Cada par nome/valor é separado por , (vírgula) e o termo nome é separado do termo valor por : (dois pontos).

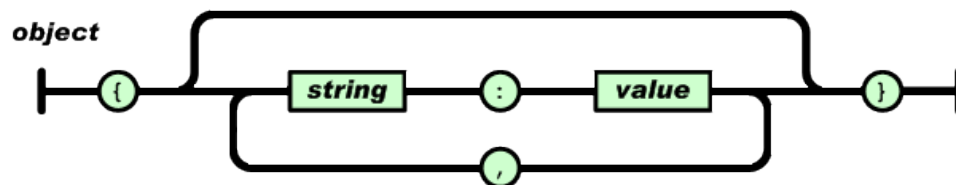


Figura 10 Estrutura de um objeto *JSON*. Fonte Crockford.

- um vetor (Figura 11) é uma sequência ordenada de valores. Ele começa com o caractere [(colchete aberto) e termina com o caractere] (colchete fechado). Cada valor presente no conjunto é separado por , (vírgula).

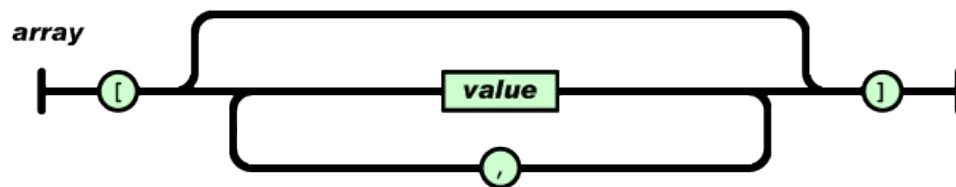


Figura 11 Estrutura de um vetor *JSON*. Fonte Crockford.

- um valor (Figura 12) pode ser um número, um valor booleano (*true* ou *false*), um objeto, um vetor ou uma *string* com aspas duplas.

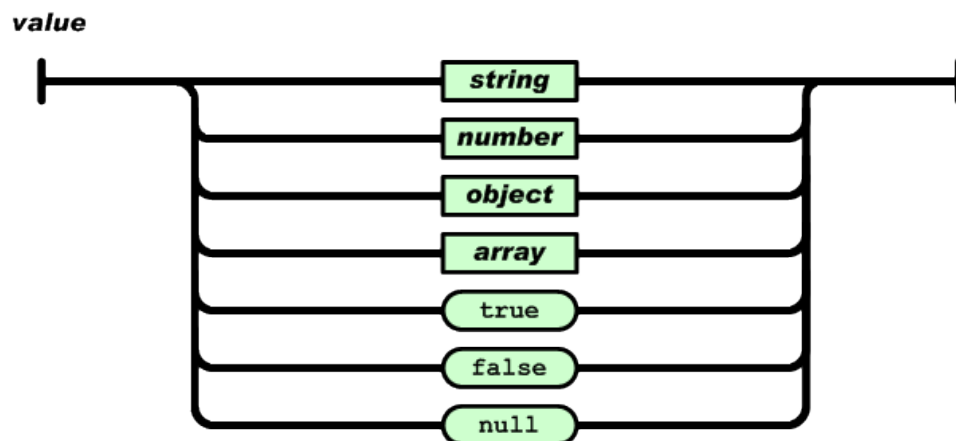


Figura 12 Estrutura de um valor *JSON*. Fonte Crockford.

- uma *string* (Figura 13) é uma sequência de caracteres Unicode iniciada e fechada com aspas duplas.

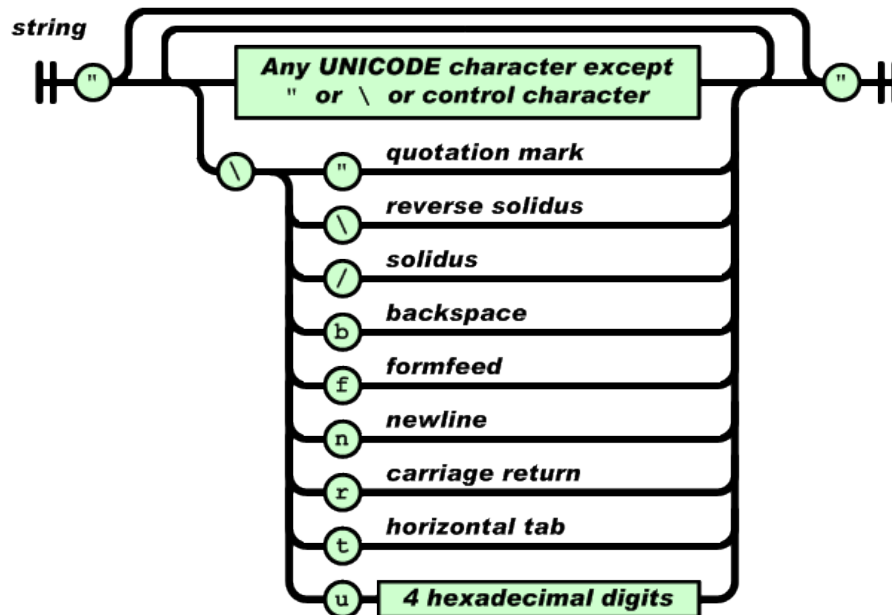


Figura 13 Estrutura de uma string *JSON*. Fonte Crockford.

- um número (Figura 14) é semelhante a um número na linguagem C ou Java, excetuando os formatos hexadecimal e octal, que não são utilizados.

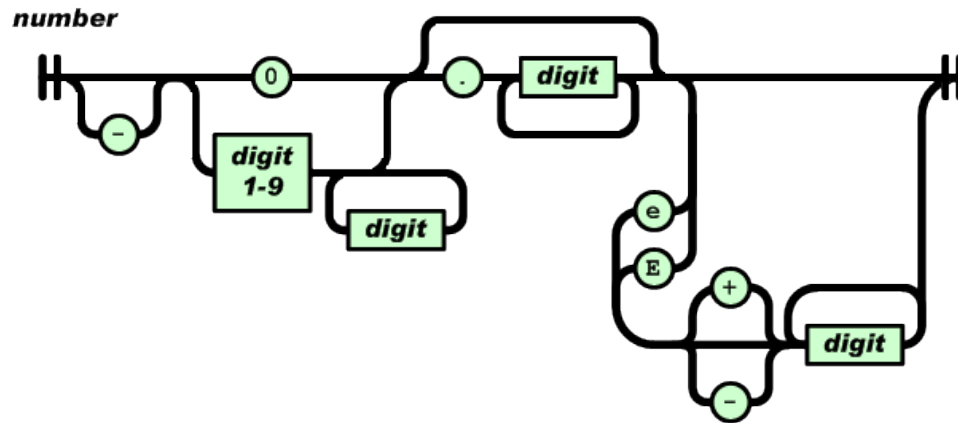


Figura 14 Estrutura de um número *JSON*. Fonte Crockford.

7 METODOLOGIA

Neste capítulo define-se onde se situa esta pesquisa, apresentando tipo e procedimentos metodológicos que foram adotados.

7.1 Tipo de pesquisa

O presente trabalho, quanto ao tipo de pesquisa, classifica-se como tecnológico, tendo objetivos de caráter descritivo, utilizando procedimentos matemáticos envolvendo otimização e mineração de dados, com base laboratorial.

A natureza tecnológica diz respeito ao desenvolvimento de um processo para coletar e analisar o sentimento ou opinião contida em documentos da *web*, trabalhando como estudo de caso com a rede social *Twitter*. Quanto aos objetivos, este trabalho faz uma pesquisa descritiva com a finalidade de observar os métodos de mineração de textos existentes e registrar a maneira como este será desenvolvido. Os procedimentos utilizados na metodologia caracterizam uma pesquisa operacional, gerando um protótipo de *software* com a finalidade de realizar as classificações de polaridade encontradas nos textos da *web*. A última etapa desse processo metodológico é a pesquisa em laboratório, onde os dados serão tratados e submetidos ao protótipo desenvolvido.

Com relação ao modo de aquisição de referências, foi realizado um estudo bibliográfico para adquirir conhecimento na área, dando a devida atenção aos princípios básicos e ao estado da arte no assunto.

O tempo de aplicação do estudo será longitudinal, pois os dados serão coletados ao longo do tempo para futura análise.

7.2 Procedimentos metodológicos

Primeiramente, será dada uma visão geral do que foi feito e posteriormente, a apresentação detalhada da construção e das funções que o protótipo em questão possui.

7.2.1 Visão geral

Implementou-se um protótipo na linguagem orientada a objetos JAVA para coletar textos da rede social *Twitter*, separar as mensagens por idiomas, realizar todo o pré-processamento e conversão para a representação apropriada requisitada pelo *svmlight* (implementação livre e de código aberto desenvolvida por Joachims (1999) do método de aprendizagem de máquina *SVM*) e posterior classificação binária (separação das mensagens em 2 classes distintas).

O *svmlight* possui vários parâmetros que servem para ajustar melhor o método para o problema específico que o usuário estará lidando. No caso deste trabalho, tais parâmetros foram os definidos por *default*.

A entrada do *svmlight* é um arquivo contendo a classe da mensagem em questão (1 ou -1, colocando 0 quando deseja-se saber a qual classe aquele documento pertence) e logo à frente, o índice de cada palavra da mensagem e o peso computado para ela (utilizando o *Tf-Idf*, sigla para *term frequency-inverse document frequency*, como explicado anteriormente). Os índices devem ser dispostos em ordem crescente, de acordo com o dicionário gerado pelo programa nessa classificação. A representação de um texto dessa maneira encontra-se demonstrada abaixo:

1	20:1.7481880270062005	21:1.6232492903979006
	22:1.9294189257142926	23:1.4771212547196624
		24:1.5185139398778875
	25:2.0791812460476247	

Devido à natureza do classificador utilizado, deve-se treiná-lo antes da execução da classificação. A entrada dos dados de treinamento se dá da mesma maneira comentada acima, só que esses dados devem ser classificados de maneira manual previamente, indicando-se a qual classe pertencem no começo da mensagem. Para validar o treinamento, deve-se utilizar também uma base de teste, que é fornecida ao *svmlight* na mesma estrutura.

Por fim, o *software* utiliza os dados classificados pelo *SVM* para mostrar ao usuário a quantidade de mensagens pertencentes a uma classe e a quantidade pertencente à outra.

A aplicação desenvolvida é *desktop* e totalmente modularizada, utilizando o padrão *MVC*, sigla para *Model View Controller*. O *MVC* estabelece uma separação entre as camadas que constituem um *software*: a visão (ou interface gráfica), também conhecida como camada de apresentação, o modelo, também chamado de camada de negócio, e um controlador, que atua como uma camada intermediária entre a visão e o modelo, contendo as operações que serão realizadas entre eles.

Isso permitirá que qualquer modificação ou manutenção que seja necessária no *software* ocorra de maneira facilitada devido a organização utilizada.

7.2.2 Visão detalhada

Para a construção do protótipo proposto, realizou-se primeiramente uma modelagem *UML* básica, visando a construção de um diagrama de casos de uso e um diagrama de classes para estruturação do modelo desejado.

O diagrama de casos de uso encontra-se representado em apêndice deste trabalho.

Através do diagrama, pode-se perceber que quem utiliza o sistema é apenas um usuário, sem a necessidade de autenticação no sistema. Ele é capaz de realizar todas as funções que o sistema oferece, a saber:

- inserir uma base de texto para a realização de todo o processamento e classificação entre as classes desejadas, realizando o processo de mineração de textos (análise de sentimento);
- identificar os idiomas contidos no texto (obtendo em qual língua cada mensagem está);
- selecionar língua desejada para filtrar o escopo da análise das mensagens;
- realizar o pré-processamento do texto (contendo a opção de quais tipos de *tokens* o usuário deseja remover);
- exibir amostra classificada selecionando a quantidade que ele deseja ver e de qual classe;
- exibir resultados tanto anteriores quanto o que resultou de um processo corrente,
- coletar textos do *Twitter* especificando as palavras desejadas;
- classificar textos através do processo de mineração de opinião;
- converter *JSON* para mensagem pois a coleta no *Twitter* é realizada nesse formato;
- utilizar modelo previamente treinado, possibilitando que se classifique novas mensagens com um modelo gerado por um treinamento já realizado;
- remover mensagens indesejadas através de trechos especificados, mostrando o que a mensagem não pode conter;

Construiu-se também um diagrama de classes mostrando a divisão entre os pacotes realizada (modelo *MVC*) e os métodos e atributos que cada classe contém. Sua representação encontra-se também em apêndice.

Como dito anteriormente, pode-se perceber que foi utilizado o padrão *MVC* para prover uma modularização estruturada dos dados e processos que o protótipo deve realizar. Cada classe modelo (do pacote *model*) possui o seu respectivo controlador (do pacote *controller*) e os controladores são responsáveis por realizar os processamentos que o usuário solicita através da interface.

Será demonstrado na seção seguinte a interface do *software*. Explicações de como foi realizada a implementação de alguns métodos e outros aspectos organizacionais do *software*, como meios de persistência e arquivos de configuração utilizados, serão dadas à medida que forem necessárias.

8 INTERFACE E IMPLEMENTAÇÃO DO PROTÓTIPO

Ao iniciar a execução do protótipo, o usuário se encontrará diante da tela principal do *software* (Figura 15). Essa é a tela que oferece as funções especificadas anteriormente através do menu no topo. Ao clicar em Arquivo, tem-se as seguintes opções:

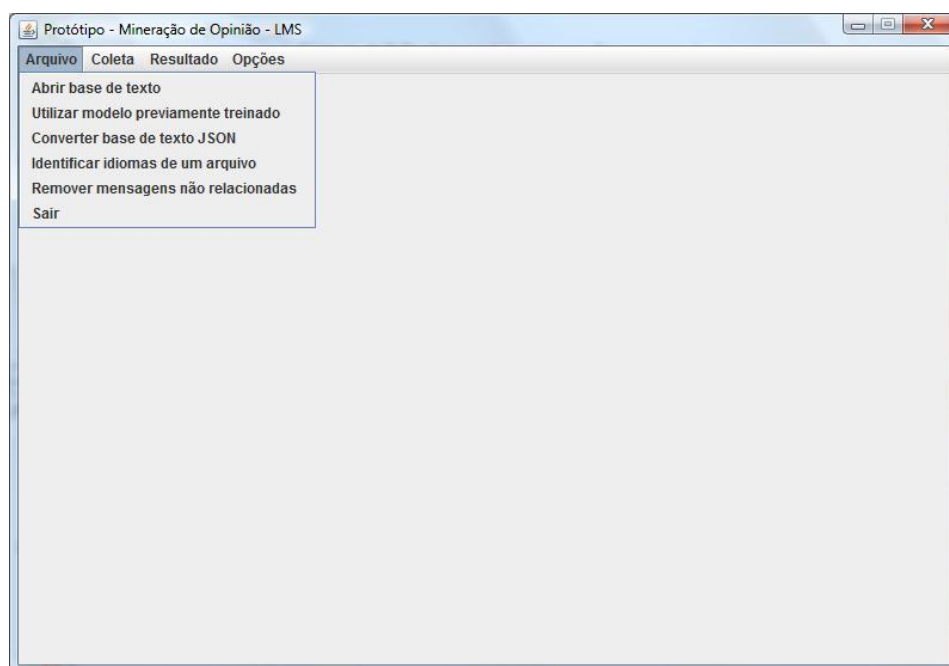


Figura 15 Tela principal mostrando opções do menu Arquivo.

As opções, como é possível observar pela Figura 15, são: **Abrir base de texto**, **Utilizar modelo previamente treinado**, **Converter base de texto JSON**, **Identificar idiomas de um arquivo**, **Remover mensagens não relacionadas** e **Sair** do protótipo.

O menu Coleta possui a opção **Realizar coleta no Twitter**, o menu Resultado possui a opção **Ver resultados anteriores** e o menu Opções contém a opção **Definir banco de dados**.

Cada opção será descrita em detalhes nas próximas subseções.

8.1 Abrir base de texto

A opção **Abrir base de texto** é responsável pelo processo principal do protótipo implementado. Nela realiza-se todas as etapas do processo de mineração de textos (excluindo-se a coleta, que deve ser feita através de outra opção).

Ao selecionar essa opção, o usuário será requisitado a escolher o arquivo que ele deseja abrir (Figura 16):

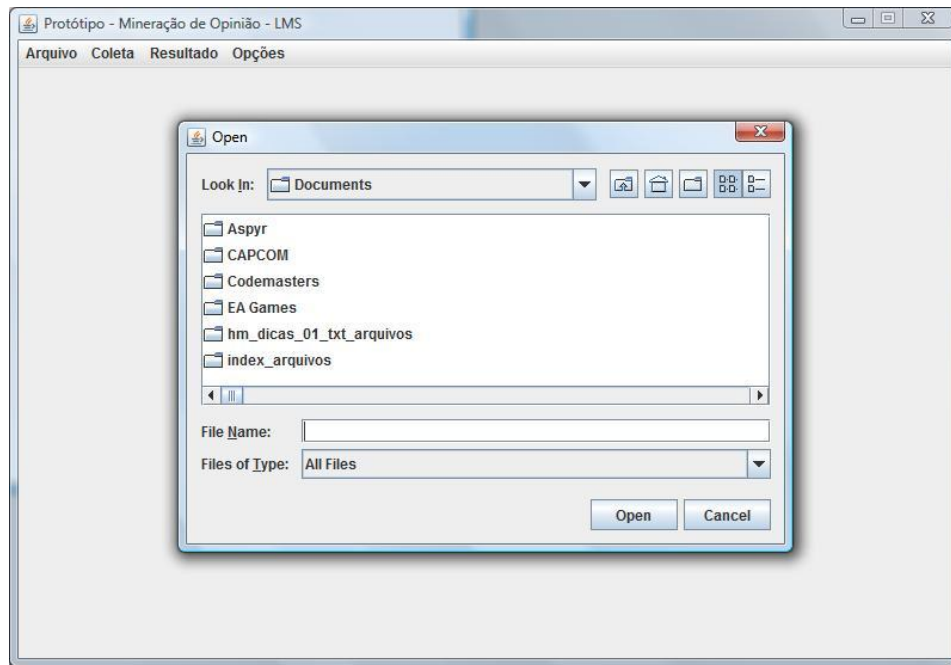


Figura 16 Tela mostrando o seletor de arquivos após seleção da opção Abrir base de texto.

Confirmando-se a seleção do arquivo, cria-se um objeto `BaseDeTextoModel` através de seu controlador contendo os atributos de acordo com o arquivo selecionado. Este objeto será utilizado durante todo o processamento, só sendo novamente instanciado quando o usuário especificar outro arquivo para processar.

O usuário então é levado à próxima tela (Figura 17) na qual deve-se escolher qual idioma deverá ser analisado:

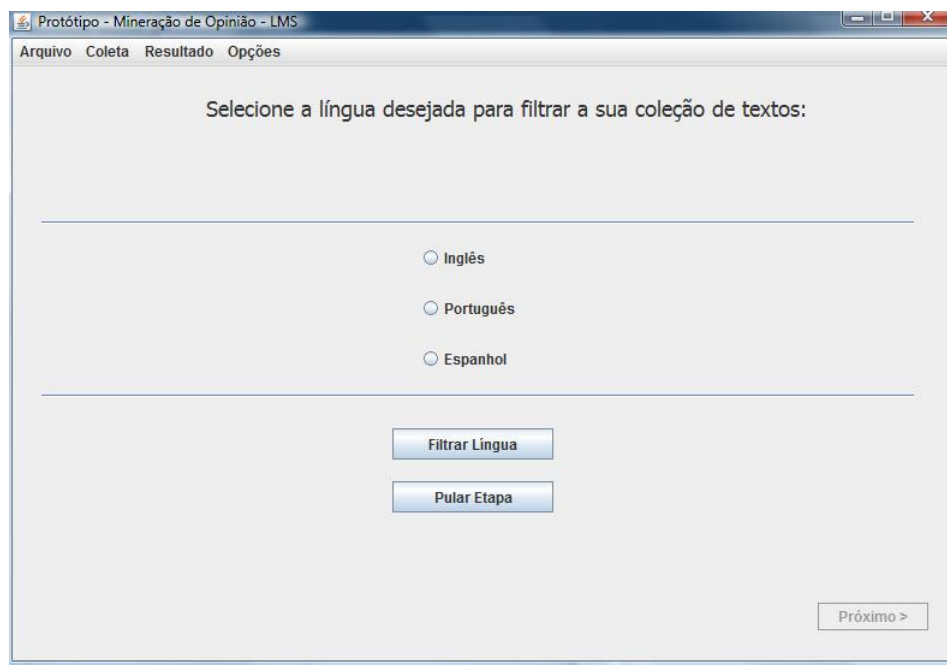


Figura 17 Tela Idioma.

O usuário tem a opção de pular a etapa caso a base de texto selecionada já esteja em um idioma específico. Caso ele deseje filtrar a base obtendo somente mensagens em uma língua, é necessário que ele tenha antes realizado a identificação das línguas através da opção **Identificar idiomas de um arquivo** no menu Arquivo. Caso ele opte por realizar a filtragem, um arquivo novo contendo apenas as mensagens na língua desejada será gerado (caminho do arquivo anterior concatenado com o “-“ e nome da língua). O andamento da filtragem é mostrado em uma barra de progresso, logo abaixo dos botões na tela (Figura 18).

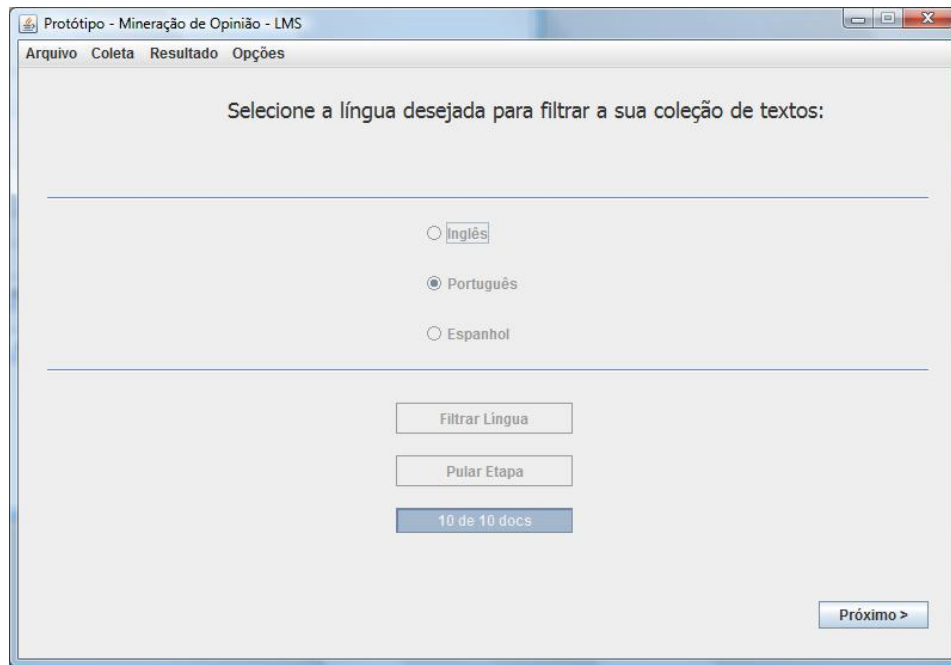


Figura 18 Tela Idioma mostrando a barra de progresso.

Ao selecionar a opção Próximo, o usuário deve especificar quais tipos de palavras ele deseja remover (Figura 19):

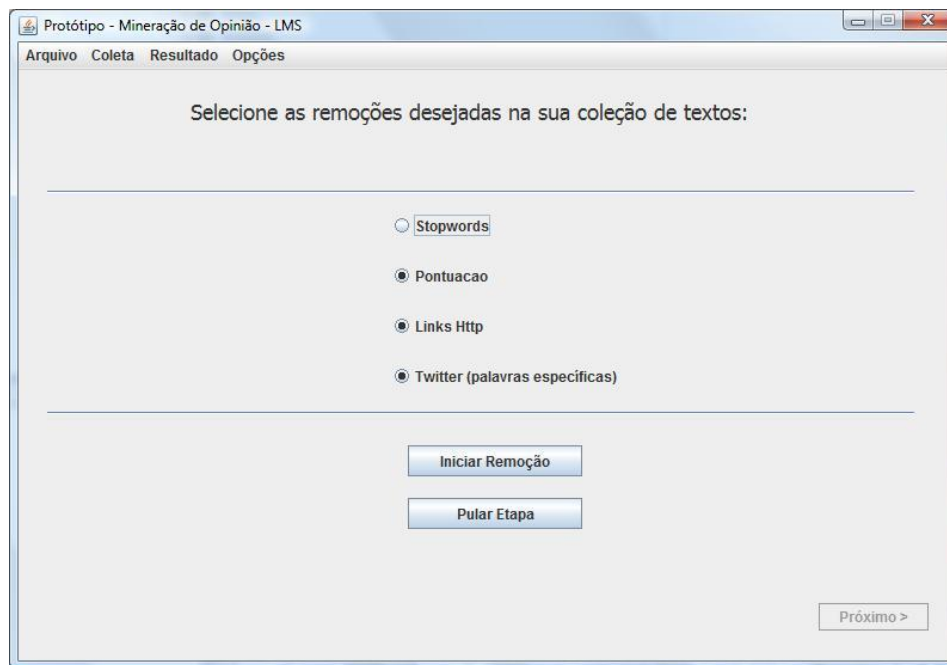


Figura 19 Tela Remoção.

A opção de pular a etapa permite que o usuário não realize nenhuma remoção, mas tal opção não é recomendada. O usuário deve marcar cada remoção que ele deseja realizar. Permite-se a remoção de *stopwords*, mas tal remoção também não é recomendada devido ao método sendo utilizado (o *SVM* lida bem com grandes quantidades de elementos) e a própria natureza das mensagens do *Twitter* (mensagens de no máximo 140 caracteres). Essas opções estão disponíveis para permitir um maior controle do usuário sobre o processo e futuras modificações no protótipo, como a adição de novos métodos e outros lugares de coleta, serem tratadas adequadamente caso recomende-se a remoção de *stopwords*, por exemplo, nesses novos casos.

Ao clicar em **Iniciar Remoção**, o processamento é realizado de acordo com as especificações do usuário e um novo arquivo é gerado contendo o texto com as remoções (seu nome é o nome do arquivo concatenado com “–

TOKENSREMOVIDOS“). O progresso é mostrado em uma barra logo abaixo do botão Pular Etapa, semelhante ao demonstrado na tela de filtro de línguas.

A opção Próximo leva o usuário à seguinte tela (Figura 20):

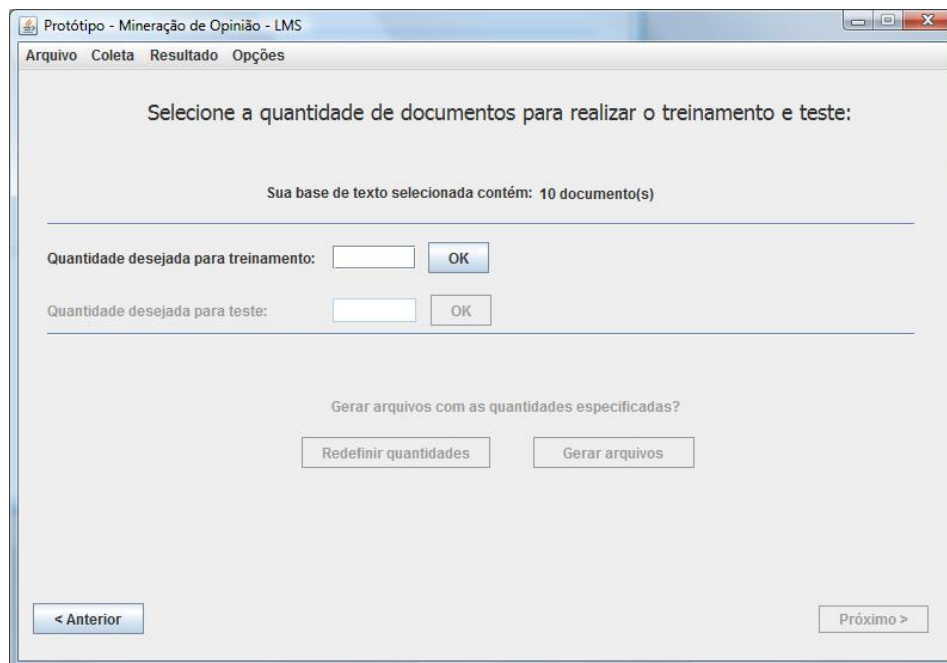


Figura 20 Tela Quantidade desejada para treino e teste.

Aqui, deve-se especificar quantos documentos o usuário deseja utilizar para treinar e testar o *SVM*. É informado quantos documentos ele possui disponível na sua base de texto e à medida que ele vai determinando as quantidades desejadas, mostra-se quantas mensagens restaram. Por fim, tem-se a opção de redefinir as quantidades informadas ou confirmar os dados passados. Ao se confirmar os dados, arquivos separados são gerados, todos com o caminho original concatenados com sua respectiva funcionalidade (um com “-TESTE” no final, outro com “-TREINAMENTO” e outro com “-CLASSIFICACAO”). Todo o andamento do processo de separação é demonstrado através de uma barra de

progresso. O conteúdo desses arquivos são as mensagens na quantidade desejada pelo usuário, sendo que cada mensagem é escolhida aleatoriamente.

A próxima tela (Figura 21) requisita a especificação do tipo de classificação que será realizada, possuindo a opção de estipular qualquer classificação binária que o usuário deseje através da opção Outro. Deve-se informar também o assunto em questão.

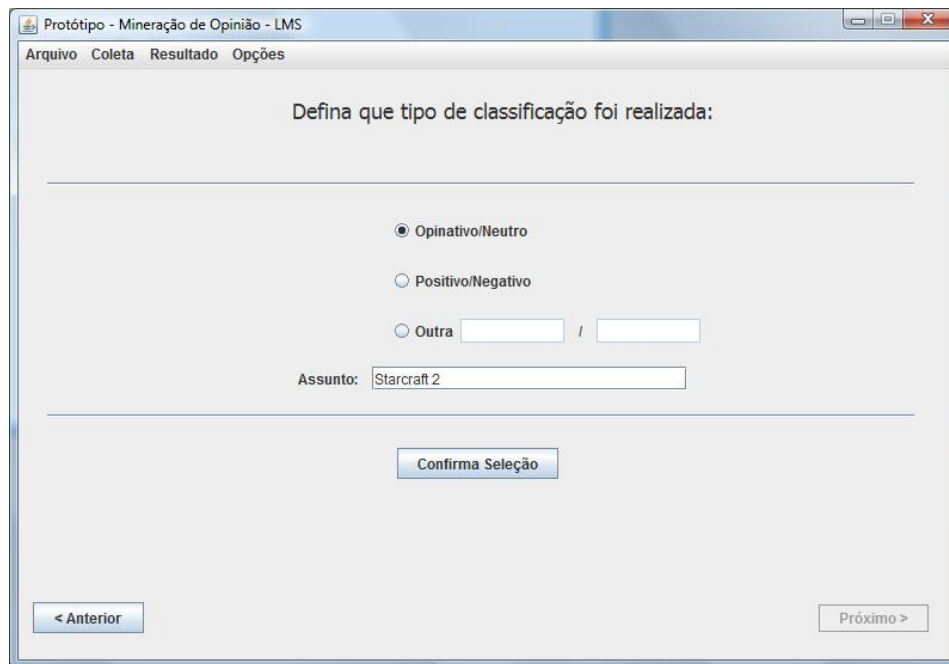


Figura 21 Tela Define Classificação.

Após confirmar a seleção e clicar em Próximo, a seguinte tela aparece diante do usuário (Figura 22):

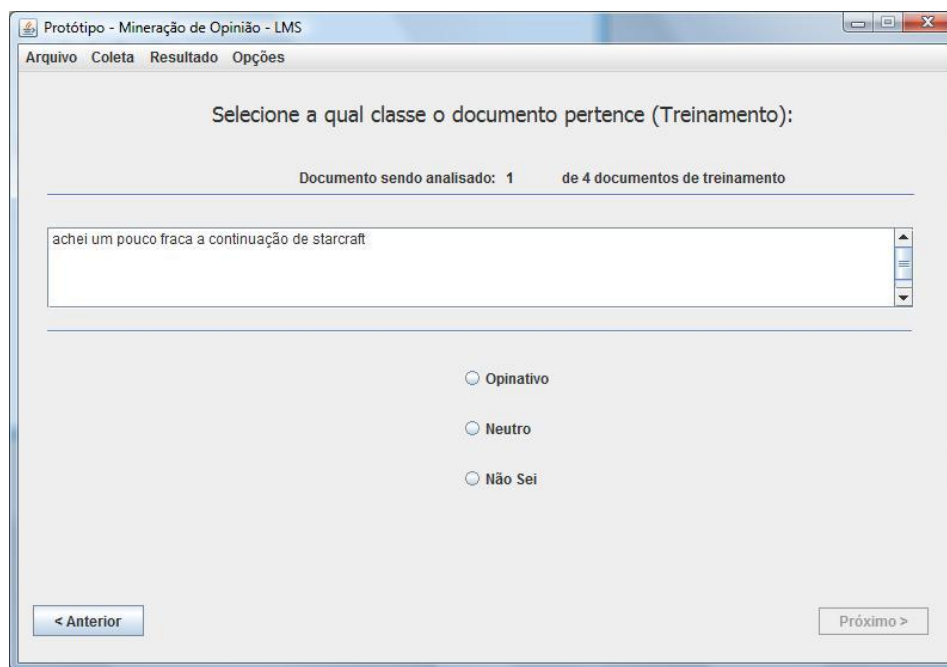


Figura 22 Tela Treinamento.

Note que as opções de classes para classificação serão correspondentes ao que foi selecionado na tela anterior. O usuário deve classificar as mensagens manualmente especificando a qual classe cada texto pertence clicando nas opções disponíveis. Ao selecionar a qual classe determinado documento pertence, realiza-se o pré-processamento dele (cálculo do *Tf-Idf* de cada *token* e escrita do índice respectivo da palavra) e escreve-se essa representação em um novo arquivo, com o nome do arquivo até então concatenado com “-PREPROCESSADO”.

Logo abaixo do título da tela, mostra-se qual mensagem o usuário está analisando no momento e quantas ainda faltam.

A próxima tela (Figura 23) mostra a saída do *svmlight*, que este protótipo executa em *background*.

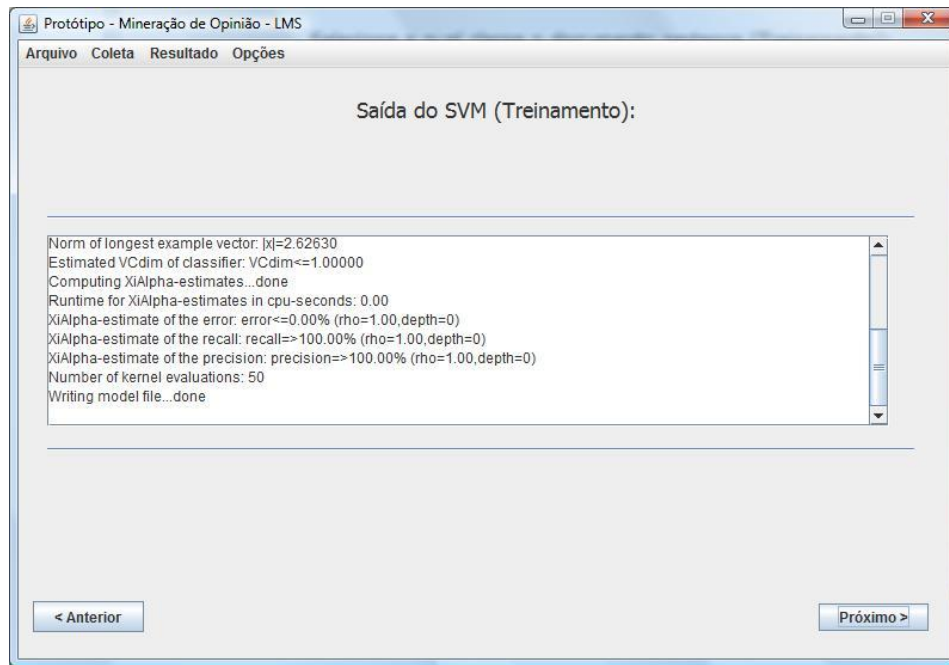


Figura 23 Tela Saida Treinamento.

Uma tela semelhante à tela de treinamento é mostrada à seguir, sendo necessário agora o usuário classificar as mensagens manualmente para realizar o teste do modelo gerado. Novamente realiza-se o pré-processamento de cada mensagem classificada e escreve-se as saídas em um novo arquivo (nome até então concatenado com “-TESTE”).

Clicando-se em próximo após finalizar a classificação, tem-se a saída do *svmlight* para o teste (Figura 24), onde ele mostra quantas mensagens foram classificadas corretamente e quantas foram classificadas de maneira errada.

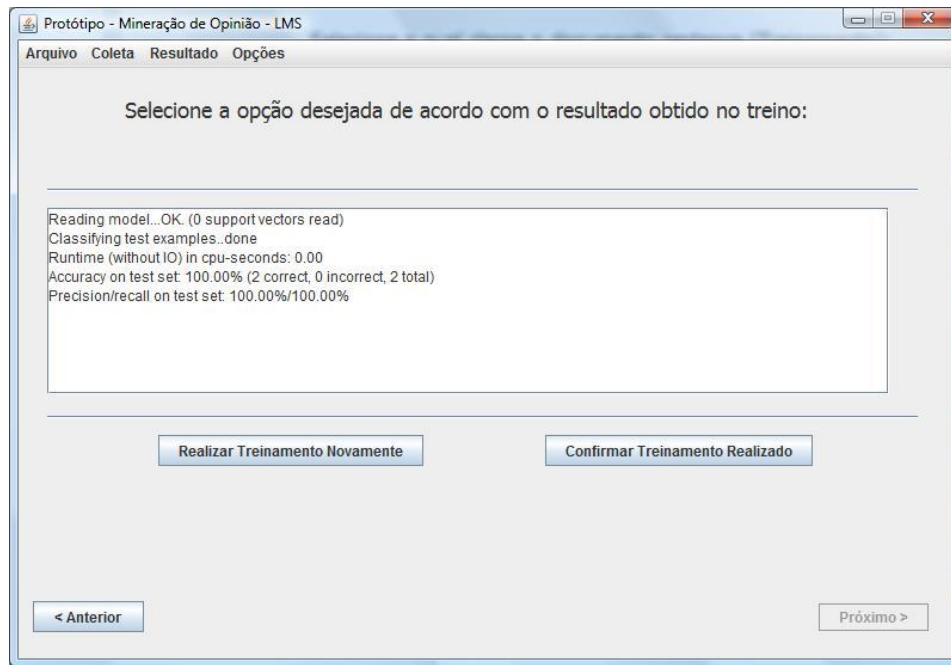


Figura 24 Tela Saída Teste.

De acordo com a acurácia alcançada, o usuário pode optar por realizar todo o treinamento novamente (caso ele considere que a taxa de acerto não foi suficiente) ou ele pode confirmar o treinamento e prosseguir para a classificação automática das mensagens desconhecidas. Se ele optar por realizar o treinamento de novo, o protótipo voltará para a tela de especificação das quantidades desejadas para treino e teste (validação). Caso a opção seja a de confirmar o treinamento, segue-se para a próxima tela (Figura 25) através do clique no botão Próximo.



Figura 25 Tela Saída Classificação.

Antes de submeter os dados ao *SVM*, realiza-se o pré-processamento desses textos restantes e escrita no arquivo com o nome anterior concatenado com “-PREPROCESSADO”, semelhante ao processo realizado anteriormente (com a diferença da classe agora ser, por padrão, 0, já que deseja-se classificá-la).

A tela da Figura 25 demonstra a saída fornecida pelo *svmlight* e o usuário deve apenas confirmar a classificação para acessar a tela seguinte (Figura 26).

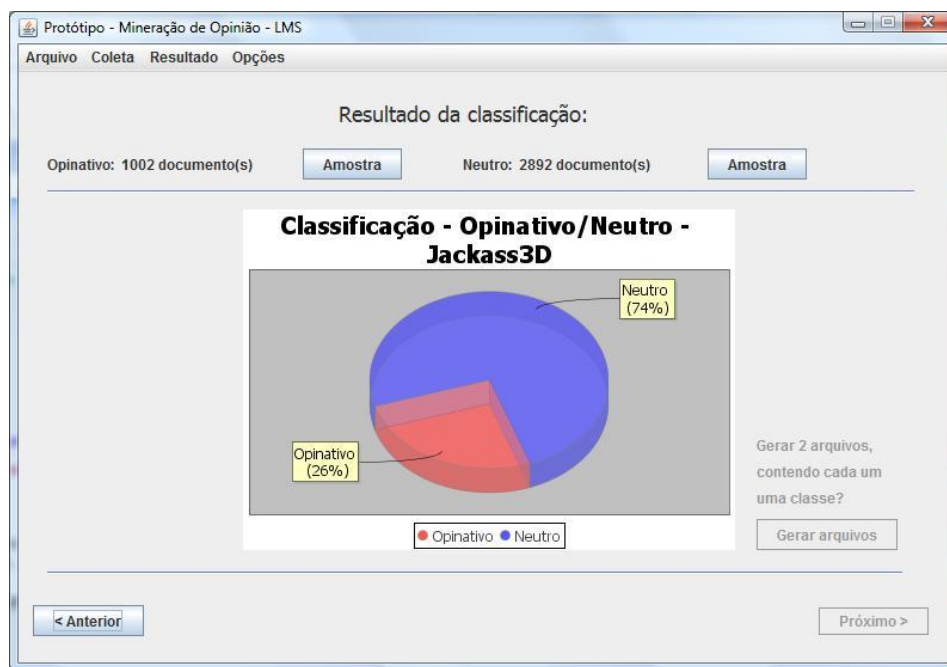


Figura 26 Tela Resultado.

Tem-se então a exibição da quantidade classificada como pertencente às classes estipuladas pelo usuário e um gráfico mostrando como ficou a divisão dessa classificação, para fornecer uma melhor visualização. No canto direito na parte de baixo da tela, existe a opção de gerar dois arquivos contendo os textos que foram classificados como pertencentes à cada classe em questão. Isso é para possibilitar a realização de uma nova classificação utilizando o arquivo de alguma classe desejada. Dois botões ao lado das quantidades encontradas permitem que o usuário visualize amostras de mensagens que foram classificadas, bastando que o usuário especifique a quantidade que ele deseja ver. A Figura 27 demonstra como é a tela de exibição de amostras.

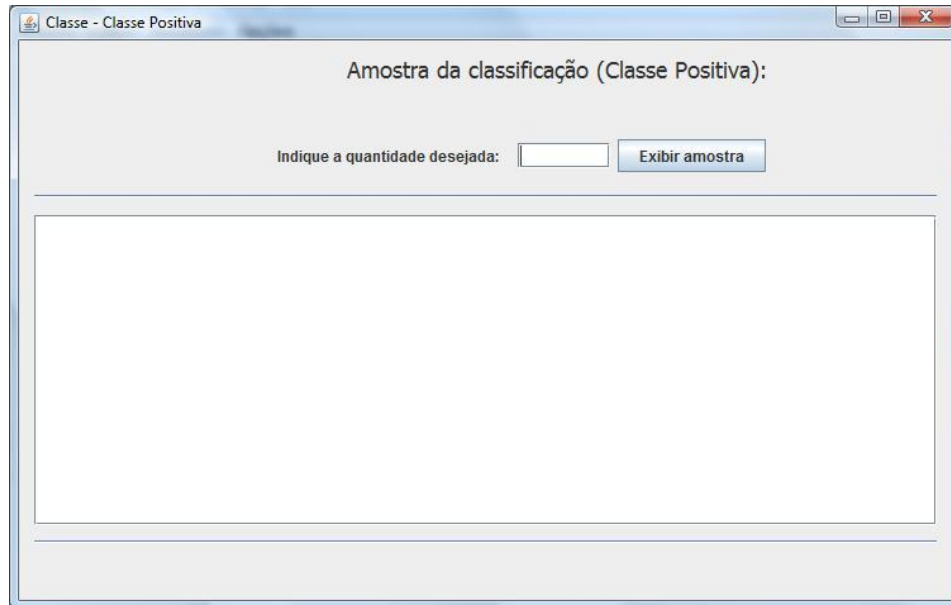


Figura 27 Tela Amostra.

Nessa etapa, é salvo em um banco de dados *MySQL* o gráfico deste resultado obtido (representado pelo objeto *graficoModel*). Ele é armazenado para futuras consultas de resultados anteriores.

8.2 Utilizar modelo previamente treinado

A opção **Utilizar modelo previamente treinado** possibilita que o usuário realize uma classificação sem ser necessário treinar e testar o algoritmo novamente, apenas se utiliza um modelo gerado em uma classificação anterior.

Ao se clicar nessa opção, uma tela requisitando a informação da localização de dois arquivos (o modelo e o arquivo a ser classificado) é mostrada (Figura 28).

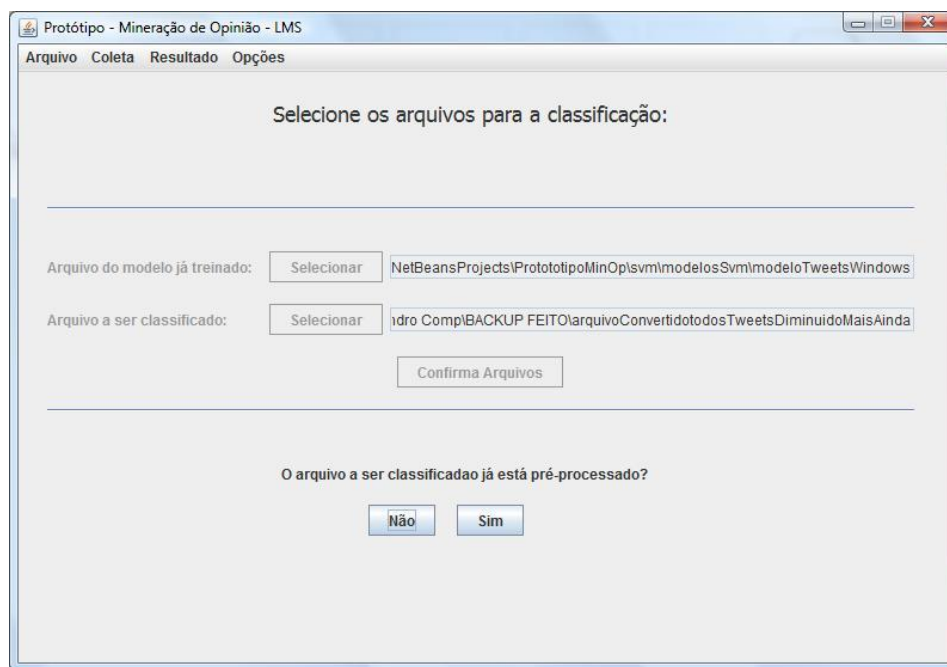


Figura 28 Tela Utiliza Modelo Treinado.

Ao se confirmar os arquivos especificados, pergunta-se ao usuário se o arquivo a ser classificado já se encontra pré-processado, ou seja, representado em vetor de características. Caso esteja, realiza-se a classificação final e mostra-se o resultado. Caso contrário, o usuário é levado às duas telas iniciais do processo de mineração completo (selecionar a língua desejada e quais palavras deseja remover do texto).

Um comentário importante está relacionado à maneira como o *Tf-Idf* é calculado nesse caso. Necessita-se utilizar o dicionário e lista invertida gerados quando se treinou o *SVM* e não um novo cálculo com o arquivo a ser classificado (pois geraria incoerência entre o treinamento e o que se deseja classificar). Palavras que são novas, ou seja, não estavam contidas no dicionário utilizado no treinamento, são descartadas pois são teoricamente irrelevantes, considerando-se que o modelo foi bem treinado.

8.3 Converter base de texto JSON

Esta opção serve para converter os objetos *JSON* coletados do *Twitter* para apenas textos representando as mensagens que os usuários postam. Abaixo tem-se um exemplo de um objeto *JSON* retornado por uma coleta realizada:

```
{
  "text": "@coolsmartphone Indeed - Windows 7 is
fantastic!",
  "in_reply_to_screen_name": "coolsmartphone",
  "in_reply_to_user_id": 14562469,
  "in_reply_to_status_id": 5803594543,
  "source": "<a href='\"http://www.tweetdeck.com/\"'
rel='\"nofollow\"'>TweetDeck</a>",
  "created_at": "Tue Nov 17
23:50:34 +0000
2009",
  "geo": null,
  "truncated": false,
  "user": {
    "profile_sidebar_fill_color": "252429",
    "profile_background_tile": false,
    "profile_sidebar_border_color": "181A1E",
    "statuses_count": 2111,
    "followers_count": 454,
    "description": "Geek, blogger, husband,
father. Thoughts Media Inc. is my
company.",
    "created_at": "Fri Apr 11 22:17:20 +0000
2008",
    "friends_count": 25,
    "geo_enabled": false,
    "profile_background_color": "1A1B1F",
    "profile_image_url": "http://a1.twimg.
com/profile_images/77853364/Japan_2008_-_Day_4_-_
_073_normal.jpg",
    "favourites_count": 0,
    "location": "Calgary,
AB",
    "following": null,
    "verified": false,
    "notifications": null,
    "profile_text_color": "666666",
    "protected": false,
    "screen_name": "jasondunn",
    "time_zone": "Mountain Time (US &
Canada)",
    "name": "Jason
Dunn",
    "profile_link_color": "2FC2EF",
    "id": 14365108,
    "profile_background_image_url": "http://s.twimg.com/a/1258496601/imag
es/themes/theme9/bg.gif",
    "utc_offset": -

```

```
25200, "url": "http://www.jasondunn.com"}, "id": 5810624870, "favorited": false}
```

O que será de interesse é a mensagem postada pelo usuário, que neste caso específico é: “@coolsmartphone Indeed - Windows 7 is fantastic!”. Realiza-se então a extração do par nome/valor (*text/mensagem*) dos objetos *JSON* coletados e escreve-se as mensagens em um novo arquivo, de nome semelhante ao arquivo coletado (com “Normal” concatenado ao caminho original).

É importante citar que cada objeto *JSON* é representando em apenas uma linha no arquivo, o que facilita seu tratamento e utilização.

Ao clicar na opção **Converter base de texto *JSON*** no menu Arquivo, solicita-se ao usuário a escolha de um arquivo para realizar a conversão.

Selecionado o arquivo, verifica-se se o mesmo contém objetos *JSON*, retornando-se uma mensagem ao usuário caso o arquivo fornecido não seja um arquivo viável para a conversão. Se o arquivo for constituído por objetos *JSON*, exibe-se a tela com o progresso da conversão sendo realizada (Figura 29).

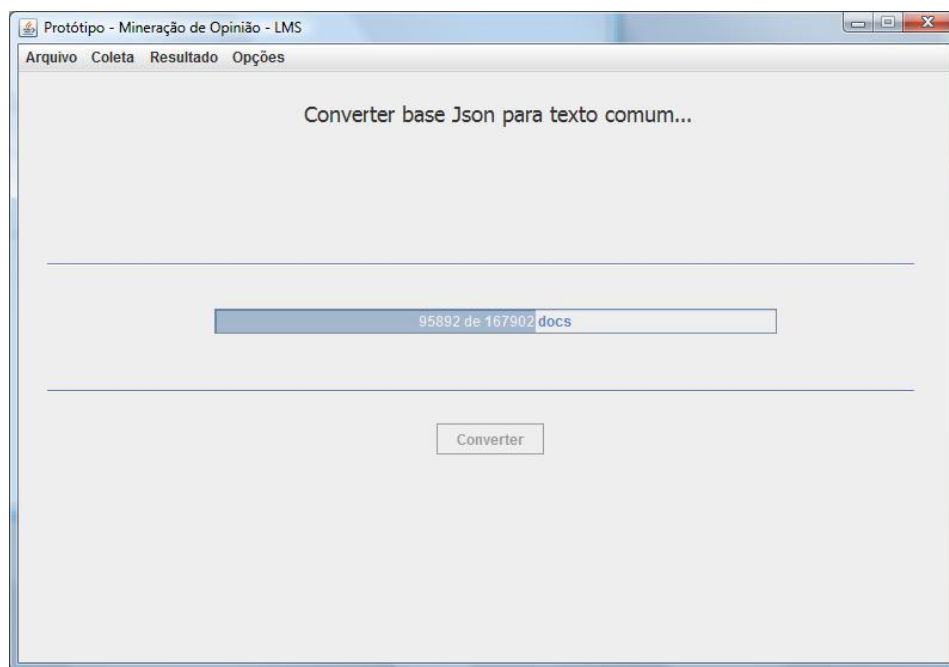


Figura 29 Tela Converte *Json*.

8.4 Identificar idiomas de um arquivo

Essa opção realiza a identificação, mensagem por mensagem, dos idiomas em que cada uma está. Para realizar tal tarefa, optou-se por utilizar a API do Google conhecida como *Google AJAX language API* (em português, API AJAX de idioma do Google). Mais especificamente, utilizou-se a função de detecção fornecida por ela.

Para realizar as chamadas, bastou fazer requisições à *url*: <http://ajax.googleapis.com/ajax/services/language/detect?v=1.0&q=> concatenado com a mensagem que se deseja identificar o seu idioma (tomando o cuidado de substituir os espaços por “%20”).

Atenção especial teve que ser dada à quantidade de requisições realizadas em um determinado período de tempo, pois na documentação da *API*

não há especificações acerca da quantidade limite permitida. Através de testes empíricos, estipulou-se um tempo de vinte minutos para realizar um intervalo na identificação de idiomas, não sobrecarregando o servidor e evitando receber mensagens de suspeita de abuso de uso da *API* e possível bloqueio do serviço.

Ao selecionar essa opção, exibe-se a tela onde o usuário deve selecionar o arquivo desejado para que se identifique em qual língua cada mensagem está. Ao se selecionar o arquivo, a tela para iniciar a conversão é mostrada (Figura 30).

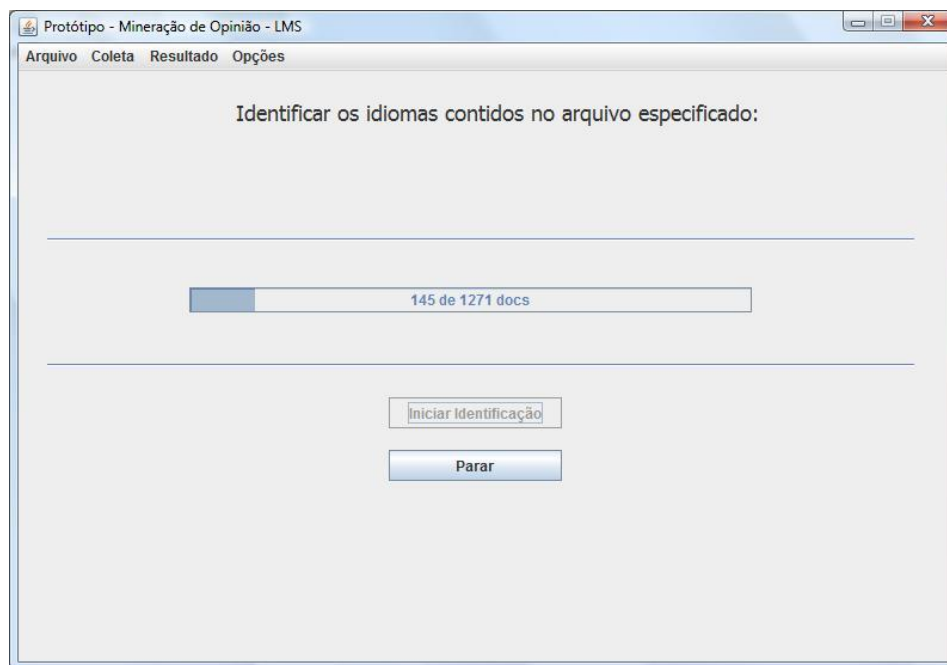


Figura 30 Tela Identifica Idiomas.

O andamento da indentificação é demonstrado na barra de progresso no centro da tela. O usuário possui a opção de parar a qualquer momento. Uma nova pergunta é realizada então ao usuário, desejando saber se este deseja salvar

essa identificação até o momento, gerando novos arquivos já organizados para continuar a identificação à partir do ponto em que parou.

Caso o processo seja realizado por completo (sem o usuário clicar em parar), a saída é salva em um novo arquivo, que possui caminho igual o original (sendo a diferença o acréscimo de “-IDLINGUAS” no nome do arquivo).

8.5 Remover mensagens não relacionadas

Ao se coletar as mensagens desejadas, vários ruídos também são adquiridos. Esta opção permite, através da especificação de trechos e/ou palavras que considera-se que a mensagem não deve ter, remover esse textos não relacionados ao objeto de estudo. Por exemplo, ao se coletar textos sobre a *Apple*, inevitavelmente veio junto mensagens de pessoas comentando que gostam de maçã. Pode-se também remover mensagens que foram identificadas como *spam* pelo usuário, após uma rápida olhada sobre o *corpus*.

A tela que é apresentada ao se clicar nessa opção é a seguinte (Figura 31):

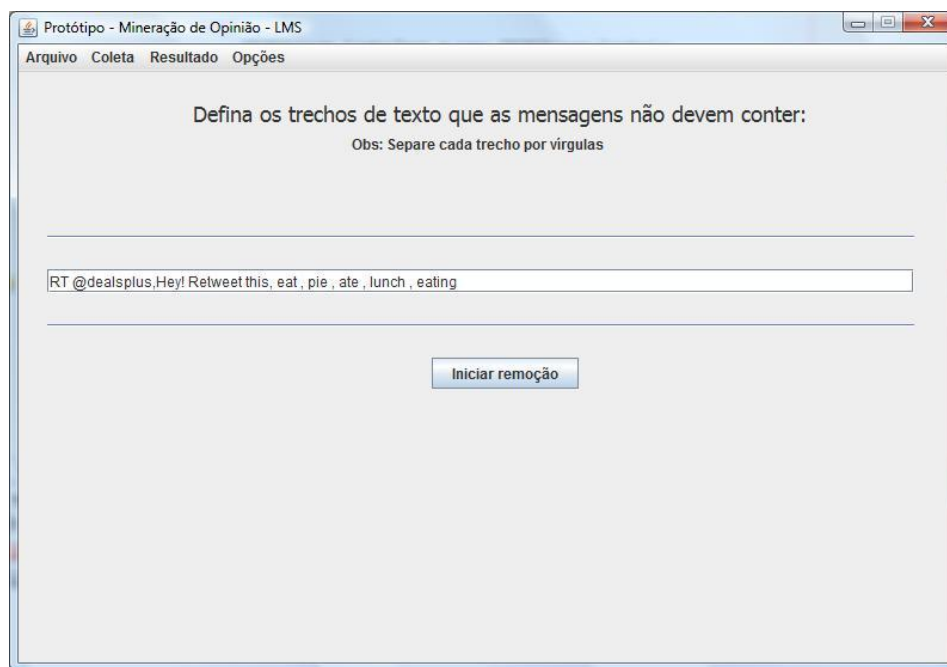


Figura 31 Tela Remove Mensagens Indesejadas.

Cada trecho a ser removido deve ser separado por vírgulas, como informa a própria interface. Ao se clicar em **Iniciar remoção**, uma barra mostrando o progresso é demonstrada logo abaixo do botão.

8.6 Realizar coleta no Twitter

O protótipo em questão lida com textos coletados do *Twitter*. Para realizar tal coleta, utilizou-se a ferramenta *cURL*, que nada mais é que uma ferramenta de linha de comando para transferir dados através de sintaxes da URL. A API selecionada do Twitter foi a *Streaming API*, sendo que o método utilizado foi o *statuses/filter* com o parâmetro de busca *track*.

Ao clicar na opção **Realizar coleta no Twitter**, a seguinte tela é exibida (Figura 32):

Protótipo - Mineração de Opinião - LMS

Arquivo Coleta Resultado Opções

Defina os parâmetros para coleta no Twitter:

Nome de usuário: testeMinera

Senha: ●●●●●●

Nome do arquivo de saída:

Defina aqui as palavras-chave desejadas:

ou

Arquivo com palavras-chave (tracking):

Figura 32 Tela Define Coleta.

Através da Figura 32, pode-se perceber a necessidade de um nome de usuário e sua respectiva senha. Essas informações devem ser de uma conta válida no *Twitter*, pois a *Streaming API* exige autenticação para que se realize a coleta. Deve-se especificar o nome do arquivo que conterà a saída e especificar as palavras-chave desejadas para direcionar a coleta. Estas podem ser definidas de duas maneiras: ou através da indicação da localização do arquivo chamado *tracking* que contém as palavras desejadas ou a especificação das mesmas através da interface visual do protótipo (o que acabará gerando um arquivo chamado *tracking* contendo as palavras digitadas).

Ao confirmar os dados, é mostrada uma tela indicando que está sendo realizada a coleta, que pode ser interrompida a qualquer momento pelo usuário.

8.7 Ver resultados anteriores

A opção **Ver resultados anteriores** disponível no menu Resultado possibilita ao usuário rever gráficos e amostras de classificações realizadas anteriormente. Cada gráfico (representado pelo objeto `graficoModel`) é salvo em um banco de dados *MySQL* no momento em que é mostrado ao final de um processo de classificação, como visto anteriormente. Ao se clicar nessa opção, exibe-se uma tela (Figura 33) contendo uma *combobox* para escolha do resultado que se deseja ver. Esta *combobox* é preenchida de acordo com as entradas obtidas do banco de dados, ou seja, os resultados armazenados nele anteriormente.

Escolhe-se então um resultado entre as opções disponíveis:

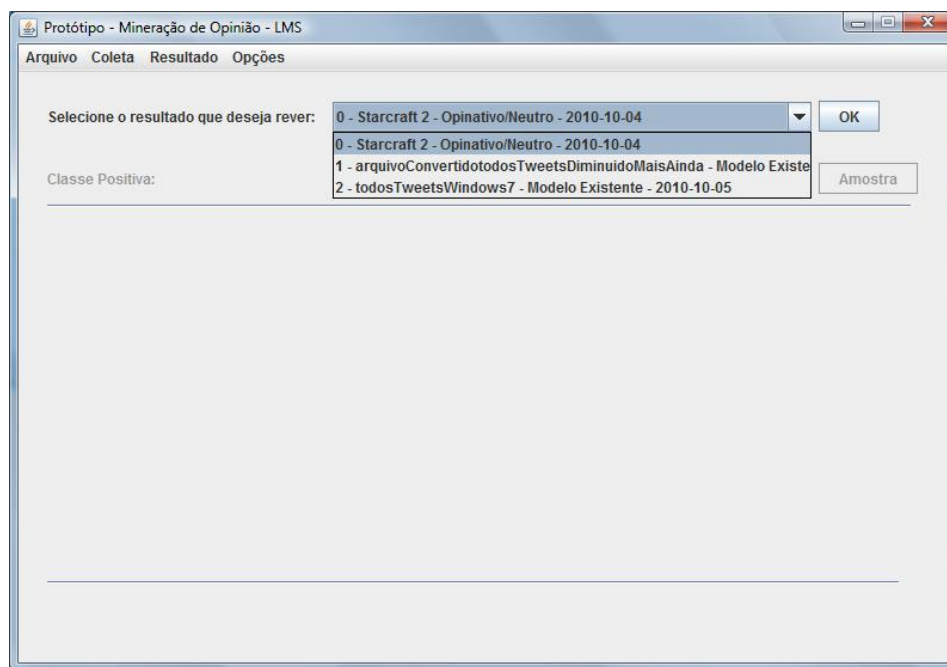
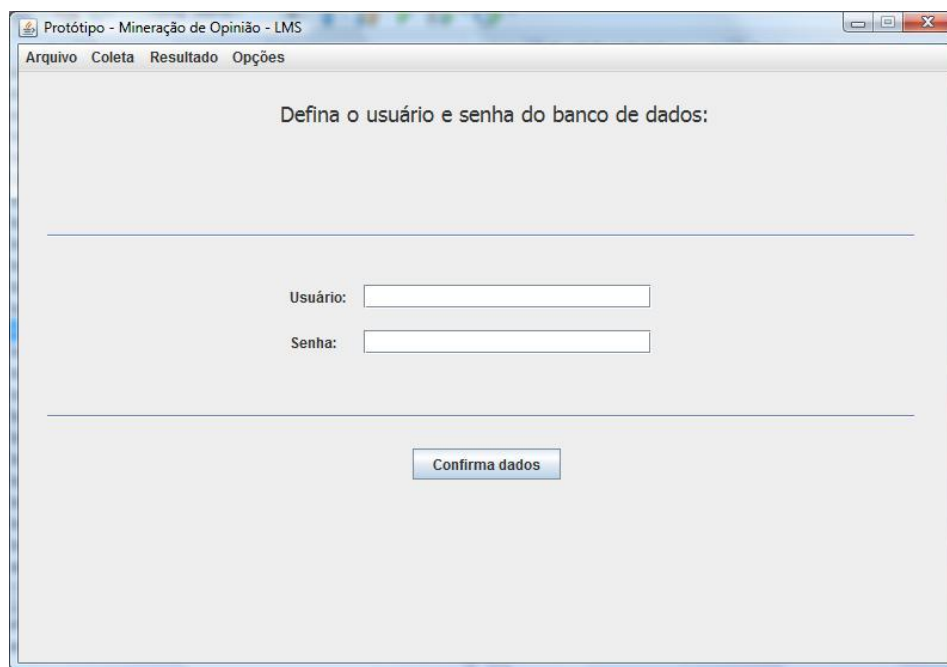


Figura 33 Tela Resultados Anteriores com combobox clicada.

Ao clicar no botão OK, exibi-se o gráfico representando aquele resultado e possibilita-se a visualização de amostras, caso os arquivos não tenham sido apagados ou movidos de seu endereço original.

8.8 Definir banco de dados

O menu Opções tem a opção que permite definir o nome de usuário e senha para acessar o banco de dados da máquina em questão. É necessário que o usuário carregue o banco do protótipo antes de tentar realizar operações que irão envolver o uso do mesmo. A Figura 34 contém a tela para a realização desta tarefa.



Protótipo - Mineração de Opinião - LMS

Arquivo Coleta Resultado Opções

Defina o usuário e senha do banco de dados:

Usuário:

Senha:

Confirma dados

Figura 34 Tela define banco de dados.

8.9 Organização de arquivos e pastas

Como pôde ser visto nas demonstrações anteriores, cada conversão de arquivo realizada gera um novo arquivo, contendo o nome do arquivo copiado concatenado com uma descrição de seu novo estado. Todas essas conversões geram arquivos no mesmo endereço do arquivo original, em geral.

Dentro do diretório do protótipo, algumas pastas foram criadas para prover uma melhor organização para os usuários.

O diretório `./svm` contém os dois executáveis que são chamados para realizar o treinamento e classificação, além de duas subpastas: `modelosSvm` e `saidasSvm`. Elas contém, respectivamente, os modelos gerados quando se treina o SVM e os resultados calculados por ele para cada classificação realizada.

O diretório `./coleta` armazena os arquivos que contém o que foi coletado do *Twitter* através do protótipo.

O diretório `./curl` contém o executável da ferramenta de mesmo nome, que é utilizado para realizar a coleta.

No diretório padrão, dois arquivos merecem especial atenção: `puntuacao` e `stopwords`. Eles contém, respectivamente, quais pontuações e palavras devem ser removidas das mensagens sendo analisadas. O usuário pode editar esses arquivos a qualquer momento para modificar e acrescentar as palavras que deseja, tomando o cuidado de especificar uma por linha.

Caso o usuário, na hora da coleta, especifique as palavras-chave pela interface, será criado no diretório do protótipo o arquivo `tracking` com as respectivas palavras.

9 RESULTADOS E DISCUSSÃO

O protótipo desenvolvido demonstrou-se funcional, pois foi capaz de realizar o processo de análise de sentimento requisitado pelo usuário de maneira razoável. Todas as etapas da metodologia para mineração de textos foram implementadas e estão disponíveis para utilização através de um processo passo-a-passo que resulta na classificação final desejada.

A implementação foi realizada de forma modularizada permitindo a adição de novas funcionalidades em trabalhos futuros realizados por desenvolvedores que queiram abranger outros meios de coleta e mineração de opinião.

Para testá-lo, foi utilizada uma coleta realizada no *Twitter* na época do lançamento do *Windows 7*, na qual continha 3156 *tweets*. Realizou-se então a identificação dos idiomas, e, ao se selecionar apenas mensagens em inglês, ficou-se com 1885 mensagens. Selecionou-se aleatoriamente 357 mensagens para treinar o *SVM* e 150 para teste. As duas amostras foram classificadas manualmente entre opinativas e neutras, para se obter apenas as mensagens que cotinham opinião sobre o assunto. O *SVM* conseguiu uma acurácia de 80% nessa classificação. Ao se classificar as mensagens restantes (1376 *tweets*), obteve-se 1268 mensagens neutras e 108 mensagens opinativas (resultado encontra-se representado na Figura 35). Com essa quantidade baixa de mensagens contendo opiniões, não foi possível realizar a separação entre mensagens positivas e negativas, pois não se dispunha de uma quantidade considerável para treinar e testar o algoritmo novamente.

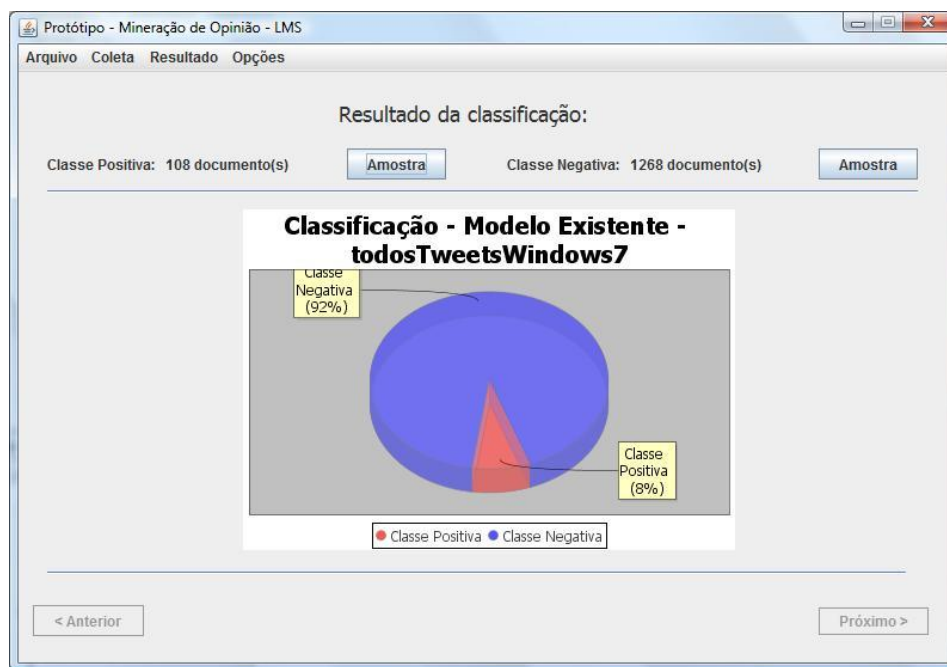


Figura 35 Resultado da classificações entre Neutro x Opinativo no caso do *Windows 7*.

Com o intuito de tentar realizar o processo de análise de polaridade, realizou-se uma coleta mais significativa no *Twitter* de mensagens sobre a *Apple*. Obteve-se 168257 mensagens e estas foram submetidas à opção de remoção de mensagens indesejadas. Objetivou-se remover mensagens que falavam do alimento maçã, e não da empresa em si. Tal remoção foi realizada através da presença de termos como “*pie*”, “*eat*”, entre outros, porém é uma tarefa extremamente complicada conseguir filtrar completamente o documento. Esse é um dos grandes problemas ao se lidar com termos que compartilham significados com outras palavras utilizadas no cotidiano.

Após essa remoção, restaram 138054 documentos em vários idiomas diferentes. Após um processo demorado de identificação de línguas (devido ao método utilizado de fazer requisições à *API* do Google), optou-se por analisar apenas mensagens em inglês (resultando num total de 109213 documentos).

Como a classificação entre neutro e opinativo no caso do *Windows 7* retornou resultados razoáveis com a quantidade utilizada para treino, optou-se por utilizar quantidade semelhante com esse corpus: 350 mensagens para treinamento e 150 para teste. Após a classificação manual dessas mensagens, obteve uma acurácia de 88% pelo *SVM*.

Ao se realizar a classificação final com as mensagens restantes (108713 documentos) e o modelo gerado, gerou-se o seguinte resultado (Figura 36):

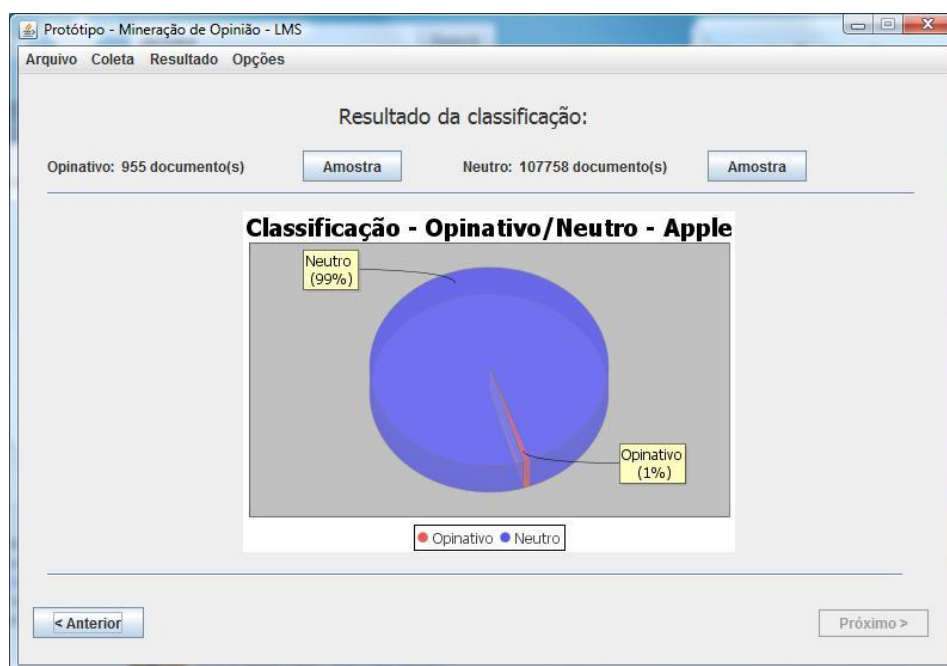


Figura 36 Tela contendo o resultado da classificação Neutro x Opinativo sobre a *Apple*.

Pode-se perceber que a imensa maioria das mensagens são neutras, assim como observado na classificação realizada com o *corpus* sobre o *Windows 7*.

Com as 955 mensagens classificadas como opinativas, realizou-se todo o processo descrito anteriormente, optando-se agora por realizar uma classificação

entre positivo e negativo. Devido à quantidade reduzida de mensagens, selecionou-se 300 mensagens para treino e 100 para teste.

Durante o treinamento, foi possível observar que ainda restaram algumas mensagens neutras (o classificador anterior não foi capaz de realizar uma separação totalmente correta entre as as mensagens, como se esperava, visto que o teste retornou uma taxa de 88%). Tais mensagens acabam gerando um ruído complicado de lidar, e como foi exposto anteriormente, esse é um dos primeiros problemas enfrentados por um sistema de análise de opinião. Outro ponto importante a ser discutido é o fato relacionado à dificuldade de interpretação das mensagens. Várias são as mensagens que não contém seu sentido explícito ou possuem ambiguidades. Como exemplo, segue a mensagem abaixo:

“Apple be fucking up. First they give us these weak ass batteries then the short ass chargers. Fuck you Apple! I still love you though =)”

A mensagem possui claramente um teor negativo com relação à *Apple*. Porém, o autor conclui dizendo que ainda ama a empresa. A que classe deve-se considerar esta mensagem como pertencente?

Outro exemplo requer uma análise mais sutil:

“I’m at the apple store.I will not buy an ipad. I will not buy an ipad.I will not buy an ipad.I will not buy an ipad.I will not buy an ipad.”

Essa mensagem por si só não fornece subsídios para considerá-la como uma opinião negativa ou positiva. Pode-se imaginar que o autor dela está tentando se conter para não comprar o produto que ele deseja muito, o *ipad*, mas também nada garante que ele não esteja afirmando veementemente que não comprará o produto devido à alguma característica ruim que ele observou ao estar na loja.

Um último comentário sobre as mensagens encontradas demonstra outro problema complexo de lidar. Os usuários do *Twitter* tem como costume citar

uma mensagem de um usuário (através da função *Retweet*), adicionando um pequeno comentário seu no início da mensagem. Tem-se então um problema simples como um usuário comentando uma mensagem de elogios à um produto dizendo que ele discorda da opinião do outro usuário. A mensagem então tem um conteúdo global positivo, porém seu teor, neste caso, é negativo, devido à discordância do usuário em questão com a mensagem original.

Como foi possível perceber, vários são os exemplos de documentos que contém mensagens complexas de serem interpretadas. Tais mensagens representam a grande dificuldade de se realizar esse processo de classificação automática.

Retornando ao processo sendo feito, treinou-se o *SVM* e o teste retornou 80% de acurácia. O resultado final ficou assim distribuído (Figura 37):



Figura 37 Classificação final entre Positivo x Negativo sobre a *Apple*.

Através da análise dos resultados e mensagens utilizadas para treino e teste, pode-se questionar essa distribuição encontrada e a validade do modelo gerado para classificação.

Observou-se durante o treinamento, várias mensagens repetidas (os chamados *Retweets*, no linguajar do *Twitter*), que foram classificadas como positivas ou negativas (houve casos para ambas as classes). Cabe aqui uma explicação do porque da não remoção dessas repetições. Considerou-se que um usuário está endossando a opinião contida ali naquela mensagem ao postá-la novamente e portanto, a quantidade de repetições influi no quanto se está falando bem ou mal de determinado assunto.

O que é realmente um problema é o teste do modelo gerado levar à falsa conclusão de que este está gerando bons resultados. No teste, ele pode classificar de maneira correta várias dessas mensagens repetidas (afinal, ele pode ter sido treinado com alguma dessas mensagens e portanto, irá classificá-las corretamente), mas não saber classificar mensagens com conteúdos diferenciados. Ele terá uma porcentagem considerável de acertos, devido à essas mensagens, mas na verdade, seu treinamento pode não ter sido bom para as mensagens que ele deverá classificar em seguida.

As amostras analisadas indicam que o classificador não foi capaz de realizar uma boa classificação entre positivos e negativos, tendo a acurácia do teste levado à uma falsa impressão de um certo grau de correte de mesmo.

Tem-se então a possibilidade do treinamento realizado não estar bom ou o método utilizado para representação dos documentos não ser o mais apropriado para o tipo de análise sendo feita e/ou para os tipos de textos analisados.

Na tentativa de identificar o real motivo do insucesso do classificador, realizou-se a remoção das mensagens repetidas para evitar que sua capacidade de generalização seja comprometida. Ao se remover esses documentos, no

entanto, obteve-se uma quantidade baixa de mensagens (restaram apenas 284), o que inviabiliza o novo treinamento e posterior teste.

Para atingir uma maior abrangência de temas e tentar identificar melhor as dificuldades de classificação entre mensagens positivas e negativas, mais uma coleta e análise foi realizada, desta vez tratando-se sobre cinema. Escolheu-se um filme que estava sendo muito falado no *Twitter* (estava presente nos *Trending Topics* do mesmo, que indica os assuntos mais comentados). O título do filme em questão é *Jackass 3D*. Obteve-se 5535 mensagens e estas foram submetidas à identificação de idiomas, seleção de uma língua apenas, treinamento, teste, separação entre mensagens neutras e opinativas, enfim, todo o processo como explicado anteriormente. O resultado obtido nesse caso específico demonstra que, quando o assunto é sobre um filme, as pessoas tendem a emitir mais suas impressões sobre o objeto de estudo. A Figura 38 demonstra a distribuição obtida.

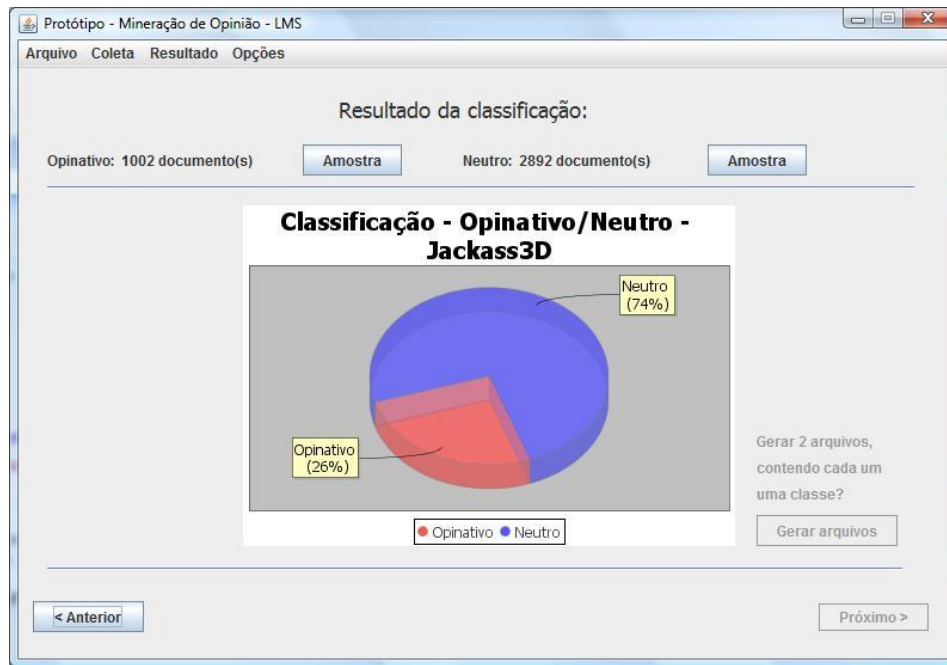


Figura 38 Distribuição obtida após classificação entre neutro e opinativo sobre o filme *Jackass 3D*.

Com a quantidade de mensagens contendo opiniões obtida, realizou-se o processo de classificação entre positivos e negativos. Através do treinamento e teste realizado de maneira e com quantidades semelhantes às especificadas nas classificações anteriores, obteve-se uma acurácia de 80.92%. O resultado final se encontra representado na Figura 39.

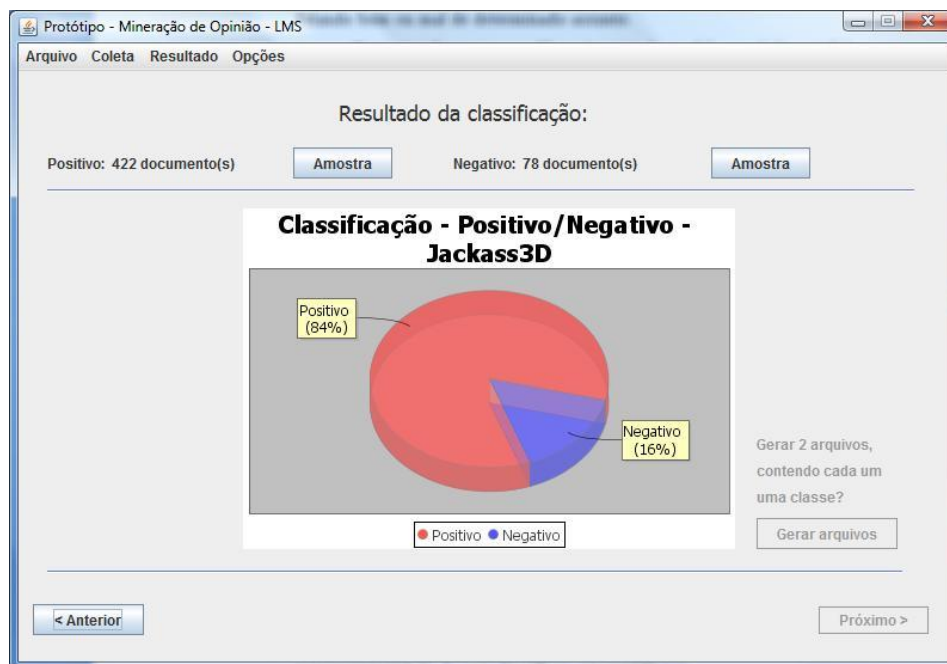


Figura 39 Distribuição entre mensagens com conteúdo positivo e negativo sobre o filme *Jackass 3D*.

Algumas ponderações sobre o resultado obtido devem ser feitas. Através da acurácia alcançada (80.92%), pode-se considerar que o método utilizado está sendo razoável para a realização de classificação entre positivos e negativos. Essa porcentagem encontrada está próximo do estudo realizado por Pang, Lee e Vaithyanathan (2002), no qual foi encontrado, utilizando os mesmos métodos, uma acurácia de 73% para o problema de classificação de polaridade em comentários de filmes. No mesmo trabalho, resultados utilizando-se apenas a presença de termos (peso dos *tokens* sendo 1 quando o documento o contém e 0 quando não) obteve uma quantidade de acertos maior (83%). No entanto, tal método não está disponível no protótipo desenvolvido, mas é de fácil implementação, podendo ser adicionado futuramente.

Através da análise das amostras, pode-se perceber que o classificador realizou uma boa separação entre positivos e negativos, ocorrendo alguns erros como foi observado pela instância de teste.

Certas mensagens merecem menção por demonstrarem a dificuldade de se realizar as classificações propostas. Um grande problema enfrentado é quando o usuário tem opiniões mais críticas sobre o filme, apontando pontos negativos e positivos, o que torna a mensagem mais complexa de ser analisada. O exemplo abaixo demonstra bem essa situação:

“There is some seriously LOL funny stuff in "Jackass 3D". Problem is, there's just as much (if not more) dumb gross-out stuff, which I hate.”

Outro exemplo, que repete a mesma ideia do anterior, mostra o problema de identificar a opinião geral expressada:

“Omg jackass 3 is disgusting funny as shit tho”

Durante o treinamento, no qual foi necessário a classificação manual de várias mensagens (como o método de aprendizagem de máquina supervisionado requer), pôde-se perceber uma grande quantidade de mensagens mostrando como o filme gerou sentimentos conflitantes nos espectadores (em geral, considerou-se o filme muito engraçado porém nojento em várias partes). Tais depoimentos constituem uma dificuldade até para o leitor que os está analisando, mostrando, portanto, que a dificuldade da classificação automática é legítima.

Retornando à classificação entre neutros e opinativos realizada para possibilitar a aquisição de mensagens apenas contendo opiniões, alguns textos também são interessantes de serem citados. Dois problemas grandes merecem considerações. O primeiro, diz respeito a mensagens que não contém nenhum indicativo aparente de serem opinativos mas o são em sua essência:

“I accidentally just called Jackass 3D a film...(It's been a long day...)”

A mensagem é opinativa pois dá uma visão pejorativa do filme. A pessoa nem mesmo o considera um filme. No entanto, não há palavras que indiquem a negatividade da mensagem, geralmente presentes em textos críticos.

O segundo ponto a ser comentado é quando a mensagem é opinativa mas não com relação ao assunto sendo averiguado, o que a torna problemática para ser colocada em uma determinada classe. Se ela for classificada como contendo opinião, ela será um ruído na próxima separação, pois não se poderá defini-la como contra ou a favor do filme. Se for classificada como neutra, seus termos, que são opinativos em sua essência, constituirão um desvio para o classificador gerar sua função de separação entre as classes, podendo levar a um resultado pior. Por exemplo, a mensagem abaixo exhibe sua opinião sobre outro filme ao invés do sendo analisado:

“Just saw the movie RED instead of JACKASS 3D... I think I made the right choice. It was an enjoyable film. :)”

Pode-se argumentar que a parte que diz que a pessoa acha que fez a escolha certa indique uma direção negativa para o filme que se deseja analisar. Porém, o trecho *“It was an enjoyable film”* é claramente positivo e seria um problema classificar essa mensagem como positiva.

Enfim, através dos exemplos demonstrados, é possível perceber as dificuldades encontradas no processo de análise de sentimento e mineração de opinião. Claro que também tem-se várias mensagens claras e diretas, nas quais o usuário expressa sua opinião de maneira objetiva, mas estas não foram citadas justamente por não apresentarem dificuldades de análise. Os exemplos acima são apenas alguns dos problemas encontrados que dificultam o processo que se deseja realizar.

A tabela abaixo representa de maneira concisa os resultados obtidos nos testes realizados (as quantidades representam quantos documentos foram considerados como pertencentes às classes especificadas):

Assunto/Tipo	Neutro x Opinativo (quantidade de docs)	Positivo x Negativo (quantidade de docs)
Windows 7	1268 – 108	—
Apple	107758 – 955	447 – 108
Jackass 3D	2892 – 1002	422 – 78

10 CONCLUSÃO

Foi possível observar que o *Twitter* não serve como uma boa fonte de opiniões no quesito produtos de *software* ou empresas relacionadas à tecnologia, nesses casos específicos. O que se notou foi uma grande quantidade de mensagens divulgando notícias sobre os assuntos especificados e poucas expondo sua opinião acerca do objeto de estudo. Pode-se imaginar portanto que o *Twitter* é uma boa fonte de divulgação e marketing de produtos para os usuários, onde as notícias se espalham rápido e em um curto período de tempo.

Tais constatações foram obtidas à partir de duas instâncias de teste e portanto, não podem ser tomadas como conclusões efetivamente verdadeiras, necessitando de outros estudos afim de comprovar ou não as suas validades.

Quando o assunto foi cinema, houve uma alteração na distribuição entre mensagens neutras e positivas, mostrando que os usuários tem uma tendência maior de comentarem sobre os filmes que acabaram de assistir.

O protótipo implementado não se torna obsoleto caso considere-se o *Twitter* um meio fraco de provimento de mensagens opinativas pois o mesmo está plenamente capaz de trabalhar com textos coletados em outras fontes, bastando apenas obter previamente o *corpus* e adaptá-lo à entrada do protótipo (um texto por linha). E sendo ele um protótipo construído de forma modularizada, ainda é passível de alterações para realizar a coleta em outras fontes, utilizar outros métodos de mineração de opinião, etc, adquirindo maior abrangência.

A não realização de uma ótima separação entre mensagens positivas e negativas também não invalidam o protótipo pois, além das várias funções implementadas que funcionam corretamente e retornam um resultado satisfatório, tem-se a possibilidade de adição de novos métodos na tentativa de realizar um processo ainda melhor, como dito anteriormente.

O grande mérito da implementação realizada é o fornecimento de um meio fácil e rápido de realização de novos testes na área de mineração de opinião com o intuito de efetivamente avaliar os métodos disponíveis. Também é fornecida uma base de dados na qual os resultados de classificações realizadas são armazenadas, possibilitando que o usuário faça uma análise ao longo do tempo.

Futuras adições envolvem a criação de um gráfico que engloba todos os resultados anteriores de determinado assunto, facilitando assim a visualização da linha temporal das classificações e quantidades encontradas em cada classe relacionadas a um assunto específico. Tal representação permite que um analista estude as distribuições encontradas em cada período e discuta o porque dos momentos de alta e baixa nas análises de seu objeto de estudo.

O agendamento de períodos de coletas também pode ser uma funcionalidade interessante de se utilizar em alguns casos, sendo interessante também a construção de uma interface *web* para o protótipo. Por fim, novos métodos de pré-processamento, classificação, identificação de línguas, filtro de mensagens além de outras fontes de coleta poderão ser adicionados com o intuito de expandir as possibilidades oferecidas pelo protótipo.

REFERÊNCIAS BIBLIOGRÁFICAS

ARANHA, Christian Nunes; VELLASCO, Marley Maria Bernardes Rebuszi (Orientadora). **Uma abordagem de pré-processamento automático para mineração de textos em português: sob o enfoque da inteligência computacional**, Tese de Doutorado - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, 2007.

CARRILHO JUNIOR, João Ribeiro; PASSOS, Emmanuel Piseces Lopes (Orientador). **Desenvolvimento de uma Metodologia para Mineração de Textos**. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, 2007, 98p.

CORTES, Corinnaand; VAPNIK, Vladimir. **Support-vector networks**. Machine Learning, 20(3):273–297, 1995.

COSTA CORDEIRO, João Paulo da. **Extracção de Elementos Relevantes em Texto/Páginas da World Wide Web**. Tese de Mestrado – Departamento de Ciência de Computadores, Faculdade de Ciências da Universidade do Porto. Junho de 2003.

CROCKFORD, Douglas. **Introducing JSON**. Disponível em: <http://www.json.org/> Acesso em: 10 jun. 2010.

CURL. **cURL Man Page**. Disponível em: <http://curl.haxx.se/docs/manpage.html> Acesso em: 9 ago. 2010.

GRÜTZMANN, A. ; ZAMBALDE, A. L. ; ESMIN, A. A. A. ; SANTOS, L. M. **Framework para a geração de inovação de produtos a partir de conhecimento extraído da web**. In: 7th CONTECSI International Conference on Information Systems and Technology, 2010, São Paulo. Abstract and Proceedings 7th CONTECSI. São Paulo : TECSI EAC FEA USP, 2010. p. 1468-1482.

GOOGLE. **API AJAX de idioma do Google**. Disponível em: <http://code.google.com/intl/pt-BR/apis/ajaxlanguage/> Acesso em: 13 set. 2010.

HAN, Jiawei; KAMBER, Micheline. **Data Mining: Concepts and Techniques, Second Edition**. Morgan Kaufmann, 2006.

JOACHIMS, Thorsten. **Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning**. B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

JOACHIMS, Thorsten. **Text categorization with suport vector machines: Learning with many relevant features**. In ECML-98, 10th European Conference on Machine, pages 137–142, Chemnitz, Germany, April 1998.

KIM, Elsa; GILBERT, Sam. **Detecting Sadness in 140 Characters: Sentiment Analysis and Mourning Michael Jackson on Twitter**. Web Ecology Project. Aug. 2009.

LEA, Richard. **'Twitter' declared top word of 2009**. The Guardian, Reino Unido, 30 nov. 2009. Disponível em: <http://www.guardian.co.uk/books/2009/nov/30/Twitter-declared-top-word-of-2009>. Acesso em: 27 dez. 2009.

LO, Yee W.;POTDAR, Vidyasagar. **A Review of Opinion Mining and Sentiment Classification Framework in Social Networks**. 3rd IEEE International Conference on Digital Ecosystems and Technologies, 2009.

MANNING, Christopher D.; RAGHAVAN, Prabakhar; SCHÜTZE; Hinrich. Term frequency and weighting. In: _____ **An Introduction to Information Retrieval**. England: Cambridge University Press, 2009. p 117-120. Disponível em: <http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf>. Acesso em: 29 dez. 2009.

PANG, Bo;LEE, Lilian. **Opinion Mining and Sentiment Analysis**. Foundations and Trends in Information Retrieval 2(1-2), pp. 1–135, June 2008.

PANG, Bo; LEE, Lilian; VAITHYANATHAN, Shivakumar. **Thumbs up? Sentiment Classification using Machine Learning Techniques**. Proceedings of EMNLP 2002, pp. 79–86.

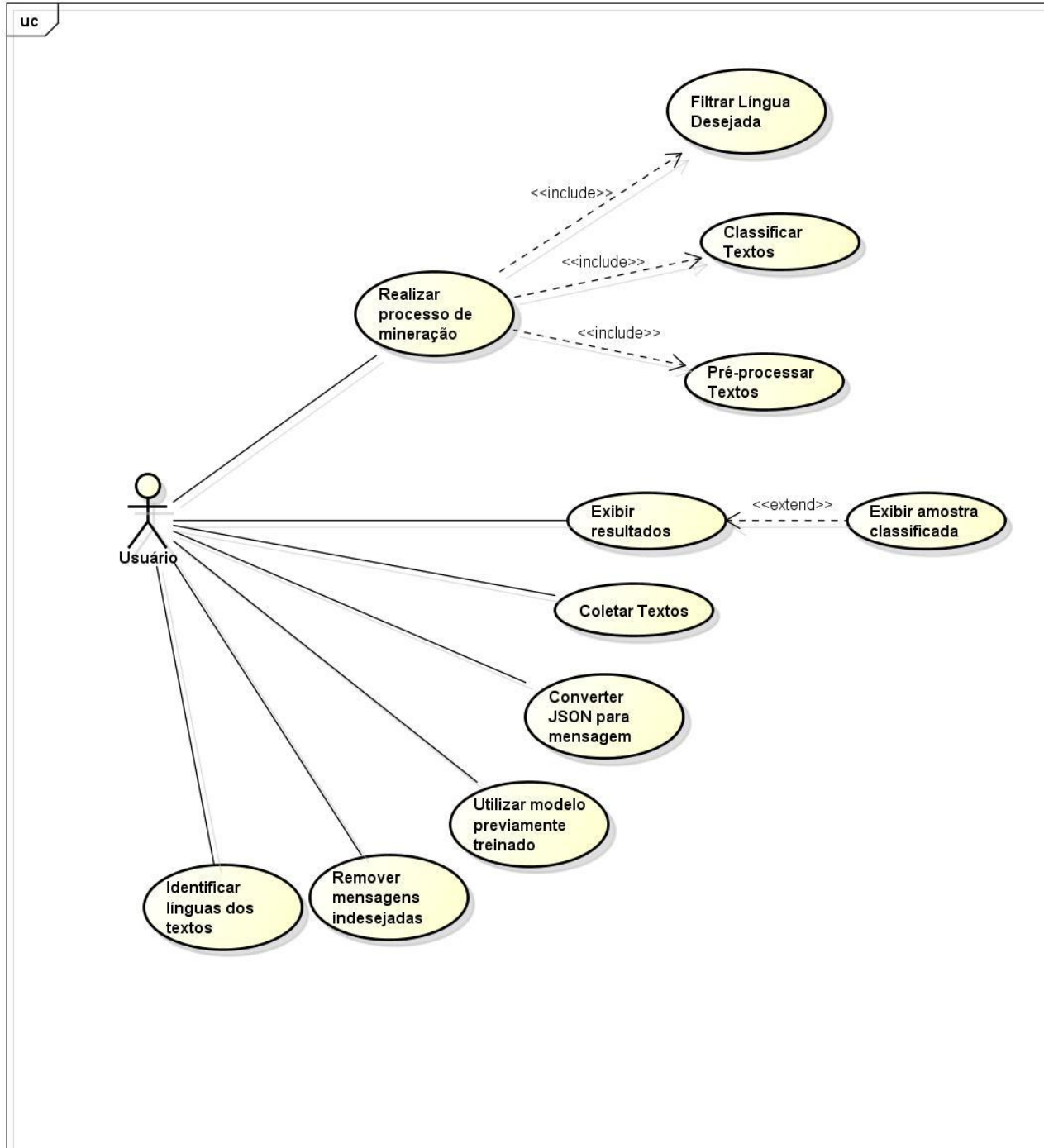
SOARES, Fábio de Azevedo; PASSOS, Emmanuel Piseces Lopes (Orientador); VELLASCO, Marley Maria Bernardes Rebuzzi (Orientadora). **Mineração de Textos na Coleta Inteligente de Dados na Web**. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, 2008.

SANTOS, L. M. ; ESMIN, A. A. A. ; ZAMBALDE, A. L. ; NOBRE, F. M. **Twitter, sentiment analysis and product development: how much users are expressing their opinions?** In: 7th CONTECSI International Conference on Information Systems and Technology, 2010, São Paulo. Abstract and Proceedings 7th CONTECSI. São Paulo : TECSI EAC FEA USP, 2010. p. 1445-1453.

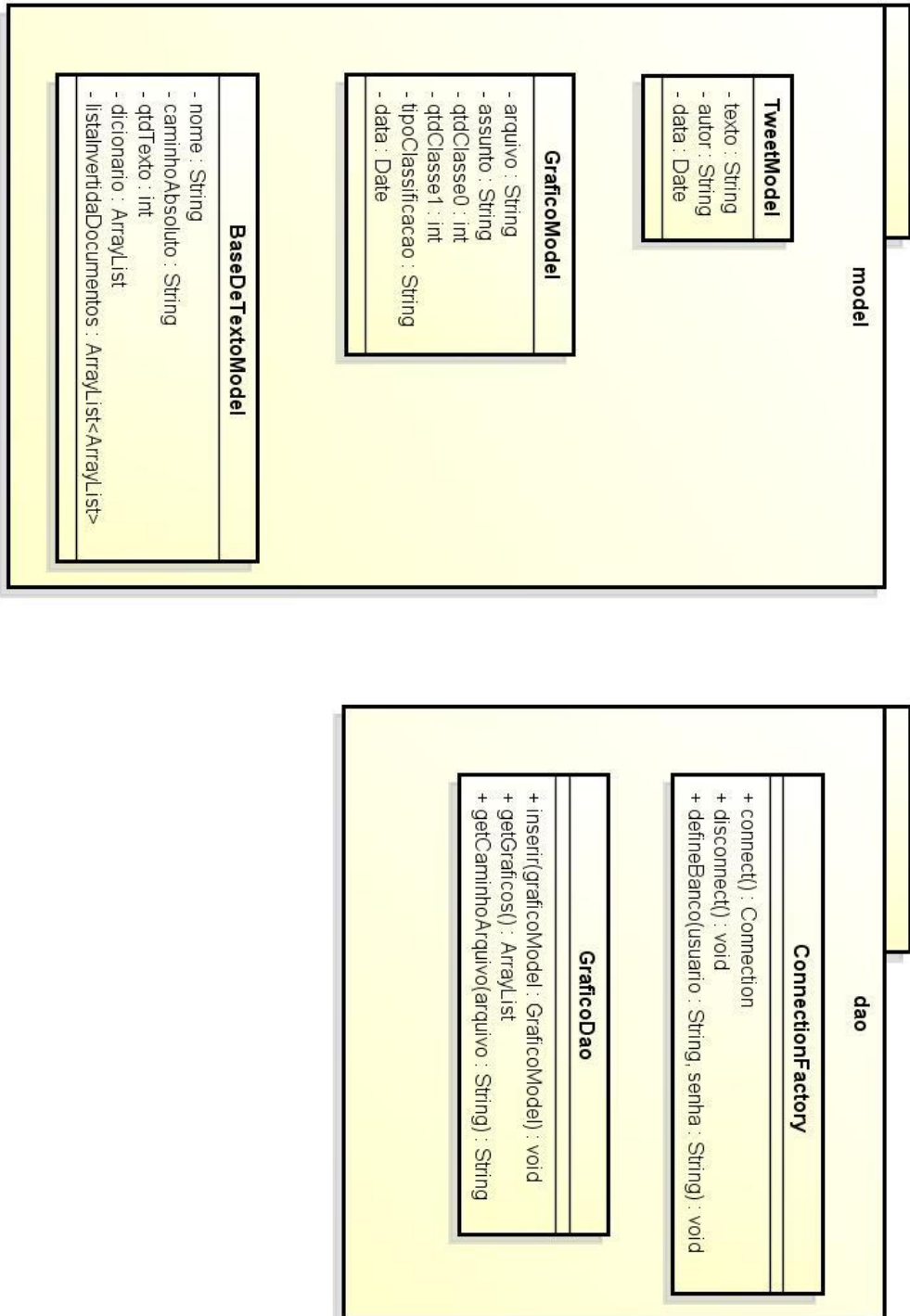
TWITTER. **Streaming API Documentation**. Disponível em: http://dev.twitter.com/pages/streaming_api. Acesso em: 28 set. 2010.

ZAGHLOUL, Waleed; LEE, Sang M.; TRIMI, Silvana. **Text classification: neural networks vs support vector machines**. Emerald Group Publishing Limited, 2009.

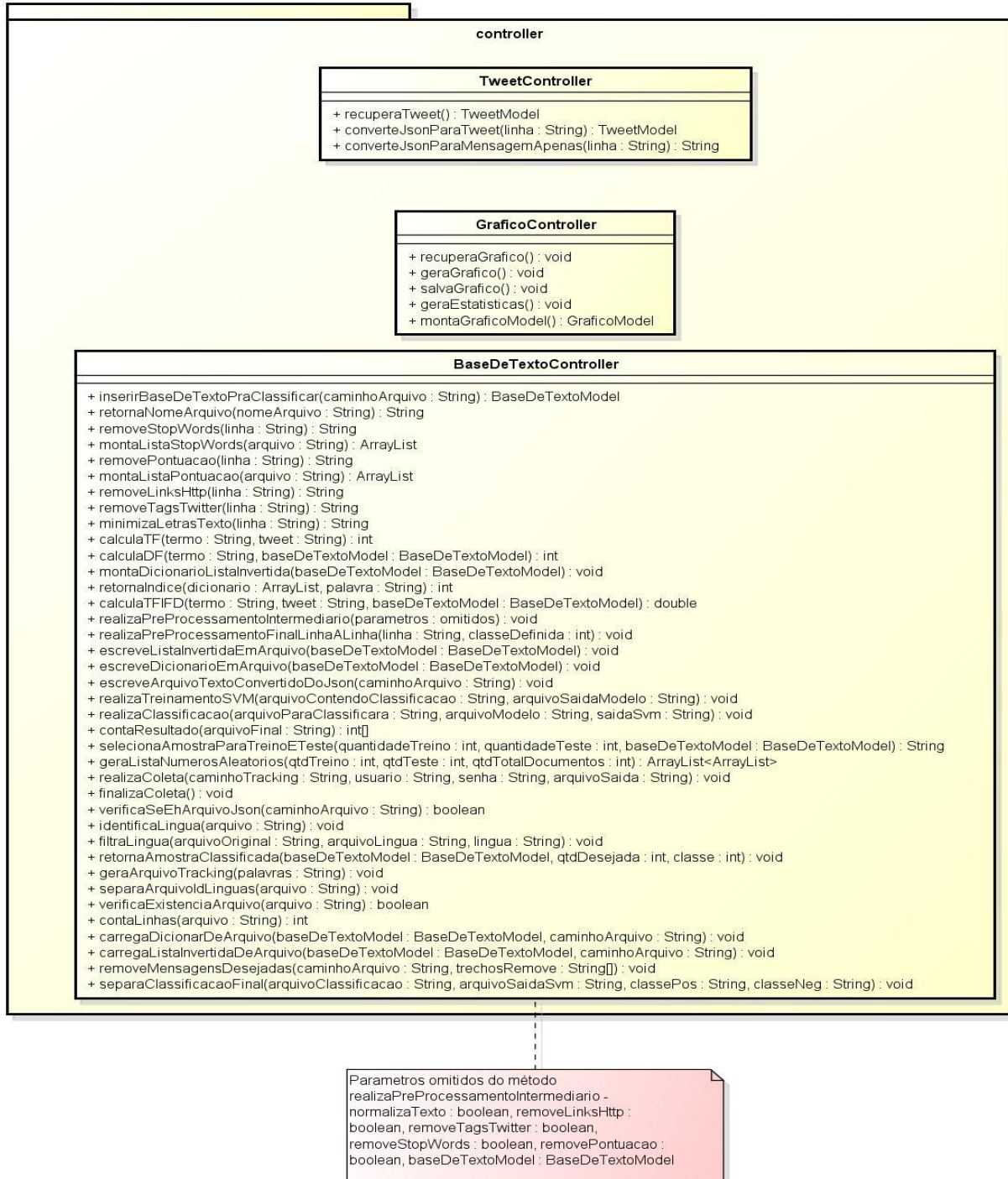
Apêndice 1 – Diagrama de Casos de Uso



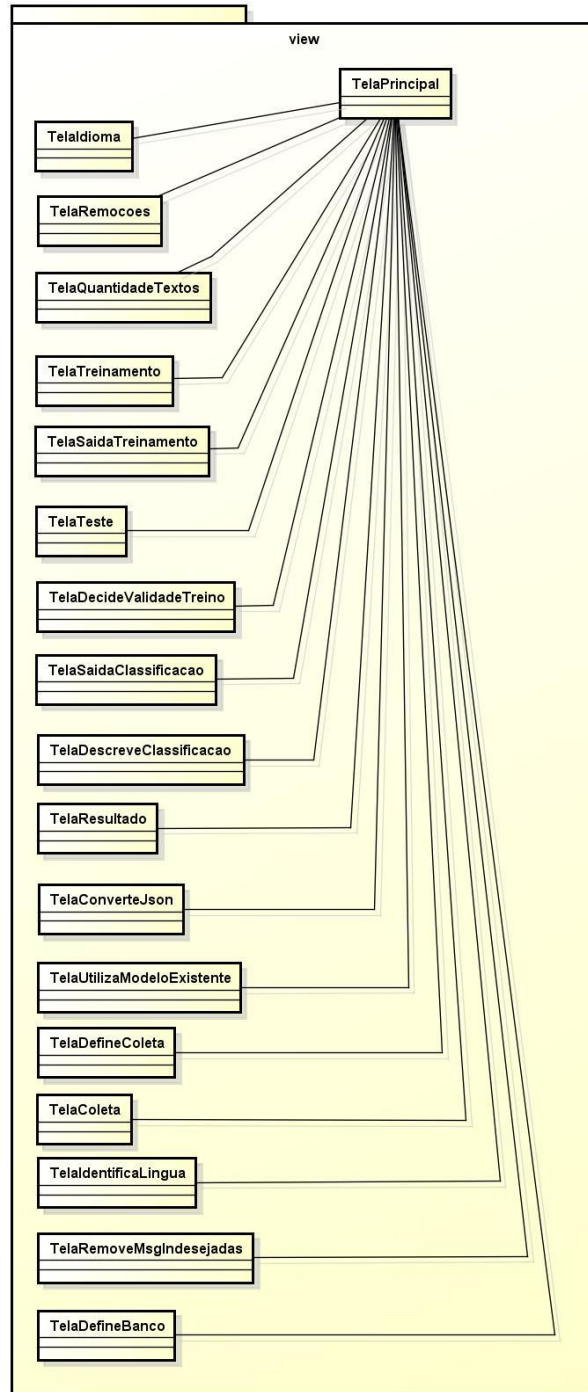
Apêndice 2 – Diagrama de Classes – Pacote model-dao



Apêndice 3 – Diagrama de Classes – Pacote controller



Apêndice 3 – Diagrama de Classes – Pacote view



Apêndice 4 – Diagrama de Classes – Atributos e métodos minimizados

