UNIVERSIDADE FEEVALE

PAULO RICARDO SCHWANCK HINKEL

CLASSIFICAÇÃO DE TEXTOS E AUXILIO A TOMADA DE DECISÃO

Novo Hamburgo

2011

PAULO RICARDO SCHWANCK HINKEL

CLASSIFICAÇÃO DE TEXTOS E AUXILIO A TOMADA DE DECISÃO

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel em
Ciência da Computação pela
Universidade Feevale

Orientador: Rodrigo Rafael Villarreal Goulart

Novo Hamburgo

2010

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial:

A minha namorada Marta Rocha, meu pai José Ricardo Hinkel, minha mãe Eroni Schwanck Leffa Hinkel, meu irmão José Henrique Schwanck Hinkel, professor e orientador Rodrigo Goulart, amigos e professores.

RESUMO

Com o aumento significativo da criação de jornais online e da inclusão de usuários sem formação acadêmica neste meio, procurou-se desenvolver mecanismos para auxiliar a tomada de decisão durante a postagem de textos em um site notícias. A principal funcionalidade desenvolvida é a classificação automática de textos a qual tem por finalidade auxiliar o usuário a manter a organização do site. A segunda funcionalidade é a indexação ou seleção, também automática, de tags ou termos mais importantes do texto postado. Tais soluções são resultado de um trabalho desenvolvido na área de Ciências da Computação utilizando técnicas de Aprendizado de Máquina Supervisionado da Inteligência Artificial a fim de utilizar uma coleção de exemplos para treinar um algoritmo de aprendizagem de máquina e assim construir um classificador de textos jornalísticos.

Palavras-chave: Classificação. Inteligência Artificial. Aprendizagem de Máquina. Aprendizagem supervisionada. Processamento da Linguagem Natural.

ABSTRACT

With the significant increase of Online newspapers and the inclusion of users without academic degree in this atmosphere, mechanisms were developed to assist decision making during text posting on a news website. The main functionality developed is the automatic text classification which aims to help the user to maintain the organization of the site. The second functionality is the automatic indexing or selection, of tags or most important terms of the posted text. Such solutions are the result of a work developed in the computer science area using learning techniques of supervised artificial intelligence machine to use a collection of examples to train a machine learning algorithm and so building a journalistic text classifier.

Keywords: Classification. Artificial Intelligence. Learning, Machine. Supervised learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
AI	Artificial Inteligence (Inteligência Artificial)
RNA	Redes Neurais Artificiais
SVM	Support Vector Machine (Máquina de Vetor Suporte)
TF	Term Frequency (Freqüência do Termo)
IDF	Inverse Document Frequency (Freqüência Inversa do Documento)
PLN	Processamento da Linguagem Natural
LN	Linguagem Natural

SUMÁRIO

INTRODUÇÃO

A categorização de textos não é uma preocupação recente. Trabalhos nesta área já são realizados desde a década de sessenta. Contudo, nos últimos anos os estudos nessa área se ampliaram acompanhando a capacidade de processamento dos computadores, o que veio a facilitar os estudos.

Auxiliar na categorização de textos jornalístico é um desafio cada vez mais importante a medida que as notícias passaram a ser veiculadas em maior número na internet do que em jornais impressos. Hoje existem muitas ferramentas cujo objetivo é postar notícia na Internet, como as seções "você repórter" em portais como Terra, Globo e até os próprios blogs de notícia. O volume de informação é imenso e a partir deste cenário surge o questionamento, quanto desta informação está segmentado coerentemente de forma a auxiliar o leitor, a saber, o que está lendo?

As ferramentas para geração de notícias na internet são práticas e fáceis de manusear possibilitando que sejam rapidamente publicadas e consequentemente apresentem erros tanto de escrita quanto de categorização. Contudo, seria muito mais fácil evitar erros se ferramentas de inteligência artificial validassem o texto e apresentassem automaticamente ao usuário soluções como a classificação.

O principal objetivo geral deste trabalho é criar mecanismos que auxiliem os usuários a postarem textos na internet. Os específicos são a categorização automática de textos utilizando técnicas de Processamento de Linguagem Natural (PLN) e Aprendizado de Máquina (AM) Supervisionado com a elaboração de um Classificador capaz de identificar padrões de novos exemplos a fim de classifica-los a categorias previamente definidas. Outro mecanismo é selecionar uma lista de palavras mais importantes do texto utilizando técnicas de pré-processamento de Linguagem Natural (LN) para isso. Desta forma, espera-se qualificar o processo produtivo da publicação de textos jornalísticos na Internet.

Este documento está organizado em três capítulos. O primeiro, Metodologias para Classificação de Textos apresenta as etapas necessárias para a criação de um classificador, dentro deste capítulo as principais etapas são Processamento da Linguagem Natural, Aprendizado de Máquina, Preparação dos Dados, Pré-Processamento, a ferramenta Weka e Algoritmos de Aprendizado de Máquina. O segundo é a Descrição do projeto que descreve a motivação e maneira como o projeto foi concebido. O terceiro apresenta a Execução que

descreve cada etapa da implementação e construção do protótipo classificador. Ao final deste documento são apresentadas as conclusões e propostas de trabalhos futuros.

1 METODOLOGIAS PARA CLASSIFICAÇÃO DE TEXTOS

As metodologias para classificação de textos são todos os processos pertinentes à criação de um classificador automático de textos. Um classificador é construído com o processamento de exemplos os quais são responsáveis pelo aprendizado de máquina. Para que tenha um bom aprendizado se faz necessária uma preparação eficiente dos dados, partindo da coleta dos textos, passando por técnicas de Pré-Processamento e concluindo na tabela de exemplos. A tabela de exemplos contém os exemplos necessários para que o algoritmo de aprendizado possa aprender e gerar um classificador.

Dessa forma, todas as etapas para a construção do classificador serão apresentadas neste capítulo.

1.1 Processamento da Linguagem Natural

O Processamento de Linguagem Natural (PLN) é a primeira questão importante a se entender no processamento digital de textos, isso porque designa a área que estuda os problemas de compreensão automática de línguas humanas naturais. Dividindo-se em sistemas para a geração de linguagem natural e sistemas para a compreensão de linguagem natural, sendo o primeiro para converter dados de computador em linguagem compreensível ao ser humano, e o segundo o contrário, transformando dados compreensíveis aos humanos para representações em formato computacional.

Segundo OLIVEIRA (2002) o fato de se processar uma linguagem natural permite aos seres humanos comunicarem-se com computadores de uma forma mais próxima a deles, ou seja, uma linguagem a qual já estão acostumados. Desta forma, eliminar-se-ia o aprendizado de uma linguagem artificial ou formas inusitadas de interação.

O processamento da linguagem natural aborda a análise de conhecimento morfológico, sintático, semântico e pragmático. A análise morfológica estuda a construção das palavras, com seus radicais e afixos, que correspondem a partes fixas ou variáveis das palavras, como as inflexões verbais. A análise sintática diz respeito ao estudo das relações formais entre palavras (JURAFSKY, 2000). A análise semântica é o processo de reconhecer sentenças de uma linguagem visando à representação de seu significado, baseado nas construções obtidas na análise sintática. A pragmática diz respeito ao processo da forma que a

linguagem é utilizada para comunicar os significados obtidos na análise semântica sobre as pessoas e seu contexto (ABRAHÃO, 1996).

Contudo, as principais aplicações da PLN no processamento digital de textos, são:

• No pré-processamento de textos: subdividir o texto em unidades fonéticas, lexicais, gramaticais, semânticas ou discursivas, de acordo com o objetivo da tarefa em questão;

• Classificar (Etiquetar) automaticamente as unidades do texto, segundo classes pertinentes à tarefa: morfossintáticas (PoS-tagger), sintáticas (Parser), semânticas (Parser Semântico ou Interpretador), discursivas (Parser discursivo). Em cada caso, é necessário definir linguagens de anotação, usadas para representar as classes: etiquetas, relações, estruturas (p.ex. árvore sintática).

• Mapear representações: da Linguagem Natural (LN) para uma representação sintática, semântica ou discursiva; e dessas para LN – Interpretação e Geração de LN.

Através do Processamento de Linguagem Natural podem-se citar as seguintes aplicações:

- Sistemas tradutores de textos (TA).
- Sistemas de sumarização automática (SA).
- Sistemas de categorização de textos.
- Sistemas de recuperação de informação (RI).
- Sistemas de extração de informação (EI).
- Sistemas de auxilio a escrita.

O Processamento da Linguagem Natural auxilia na classificação de textos através de classes pertinentes a tarefas: morfossintáticas, sintáticas, semânticas e discursivas. Essas classes são utilizadas no pré-processamento dos textos, onde são preparados os atributos para serem processados por algoritmos de aprendizado de máquina.

As classes Morfossintáticas são conhecidas por Pos-Target e Post-target e são responsáveis por relacionar as origens sintáticas dos termos do texto, como por exemplo: substantivos, verbos, adjetivos, advérbios, etc (CHARNIAK, 1997). Classes Morfossintáticas também podem auxiliar no trabalho de radicais onde se se pode dividir uma palavra em um radical + morfema gramatical conforme o exemplo:

- menina = menin + a
- cadeiras = cadeira + s
- fazia = faz + ia

Neste exemplo a palavra menina é dividida em “menin” e “a”, sendo “menin” é o redical e o “a” indica o gênero. A mesma operação é feita no segundo exemplo onde a “cadeira” é o radical e o “s” indica o plural. Assim como acontece com o terceiro exemplo reduzindo todas as conjugações verbais a seu radical. Desta forma um sistema atribui uma relevância maior aos atributos do texto em razão da sua frequência, isso por que evita que muitos termos com mesmo sentido sejam contabilizados separadamente como, por exemplo, a seguinte frase:

“O sopro de vento empurrou lentamente a velha porta entre-aberta.”.

Nesta frase os radicais devem ser separados da seguinte forma: “sopr”, “vent”, “empurr”, “lent”, “velh”, “entr” e “abert”.

O processamento sintático é responsável por analisar a estrutura das palavras e textos, retirando atributos desnecessários para o processamento digital como pontos e caracteres específicos. Já o parser léxico processa cada caractere do texto verificando se pertence a determinado alfabeto (CHAPMAN, 1987). As classes sintáticas e léxicas organizam o texto separando-o em palavras construindo assim uma estrutura de dados, em geral uma árvore. Por exemplo, a frase “O sopro de vento empurrou lentamente a velha porta entre-aberta” pode ser analisada de acordo com a árvore morfossintática apresentada na Figura 1.

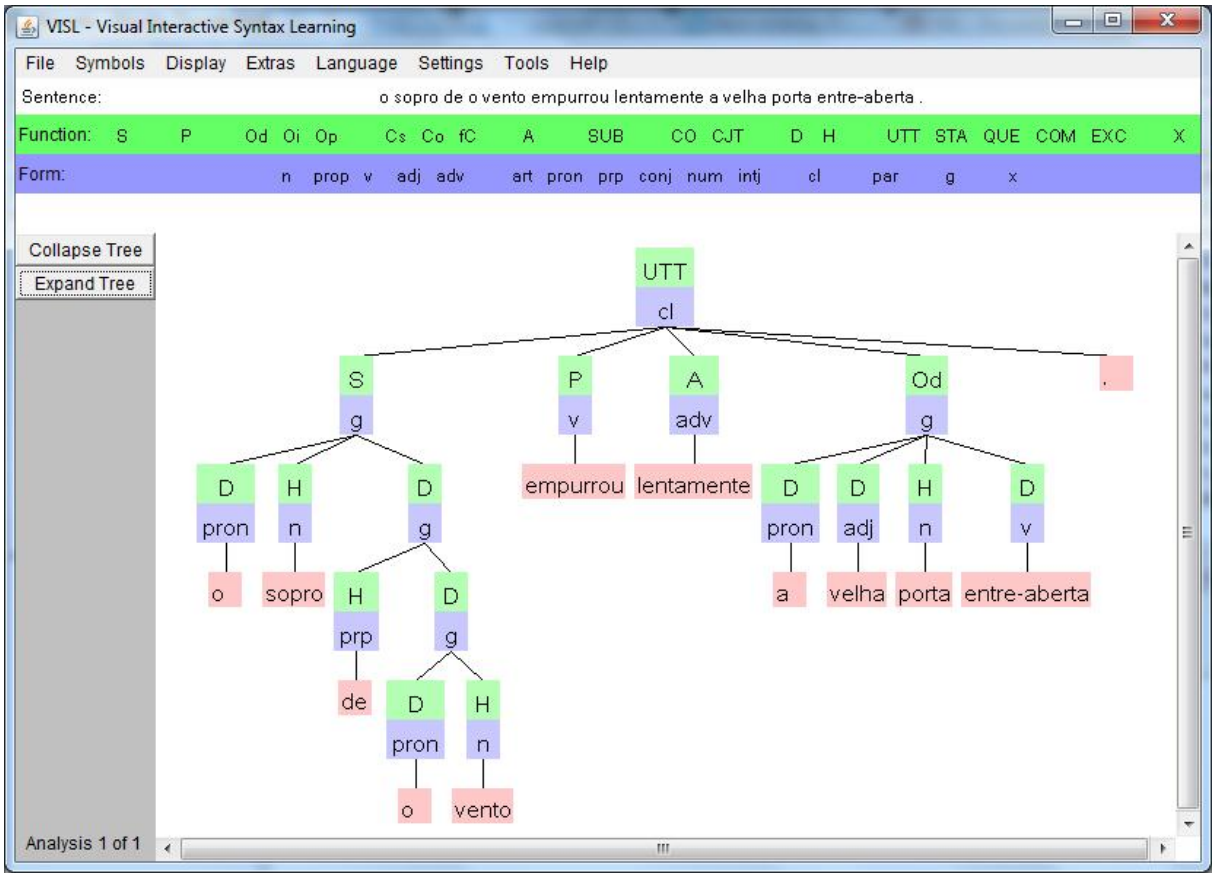

Figura 1 - Exemplo de árvore morfossintática.

A árvore morfossintática da Figura 1 foi gerada através do site VISL da Universidade de Dinamarquesa Syddansk Univers (http://beta.visl.sdu.dk/). Na árvore foram separadas as palavras da seguinte forma:

- O - pronome pessoal
- Sopro - nome, substantivo
- De – preposição
- O - pronome pessoal
- Vento - nome, substantivo
- Empurrou – verbo
- Lentamente – advérbio
- A – pronome pessoal
- Velha - adjetivo
- Porta - nome, substantivo
- Entre-aberta – adjetivo

Já o Parser discursivo é responsável por relacionar diversos textos, estruturando o discurso de forma a conectar sentenças provenientes de diferentes documentos e estabelecendo relações entre si. Utilizado por exemplo para saber que informações se complementam, quais são conflitantes, quais fatos antecedem outros, causas e suas consequências, etc (AFANTENOS, 2004).

Reconhecer características de palavras em textos é pertinente para a criação de um Corpora anotado. Corpora anotado é um conjunto de características do texto devidamente segmentando e organizado como tabela de exemplos. Desta forma pode-se processar a tabela de exemplos em um algoritmo de aprendizado de máquina, e assim, criar um classificador de textos.

Tais razões fazem do estudo do Processamento da Linguagem Natural interessante para a construção de atributos para o processamento em algoritmos de Aprendizado de Máquina.

1.2 Aprendizado de Máquina

Considerando que a automatização da categorização de texto é um dos benefícios do Aprendizado de Máquina (AM), os conceitos e técnicas da mesma serão abordados neste capítulo. O AM é uma área da Inteligência Artificial (IA) cujo objetivo é desenvolver sistemas chamados inteligentes, capazes de adquirir conhecimento de forma automática. Desta forma, os sistemas se tornam capazes de tomar decisões baseados em experiências acumuladas por meio de aprendizagem ou treinamento.

Segundo Mendel & McLaren (1994): “Aprendizado é o processo pelo qual um sistema inteligente é adaptado através do estímulo do ambiente no qual está inserido. O tipo de aprendizado é determinado pela maneira através da qual esta adaptação é realizada.”.

De uma maneira mais prática, Mitchell (1997) explica da seguinte forma: "Um programa computacional aprende a partir da experiência *E* em relação a uma classe de tarefas *T* com medida de desempenho *P*, melhora com a experiência *E*". A partir disto, pode-se usar o seguinte exemplo:

Jogo de Damas:

Tarefa *T*: jogar damas.

Medida de desempenho *P*: porcentagem de jogos vencidos contra oponentes.

Experiência de treinamento *E*: praticar jogos contra si mesmo.

Através do exemplo do Jogo de Damas é possível compreender o comportamento das variáveis: tarefa, desempenho e experiência na construção de sistemas inteligentes.

Conforme a Figura 2, o aprendizado de máquina pode ser separado nos seguintes tipos: aprendizagem supervisionada e aprendizagem não supervisionada.

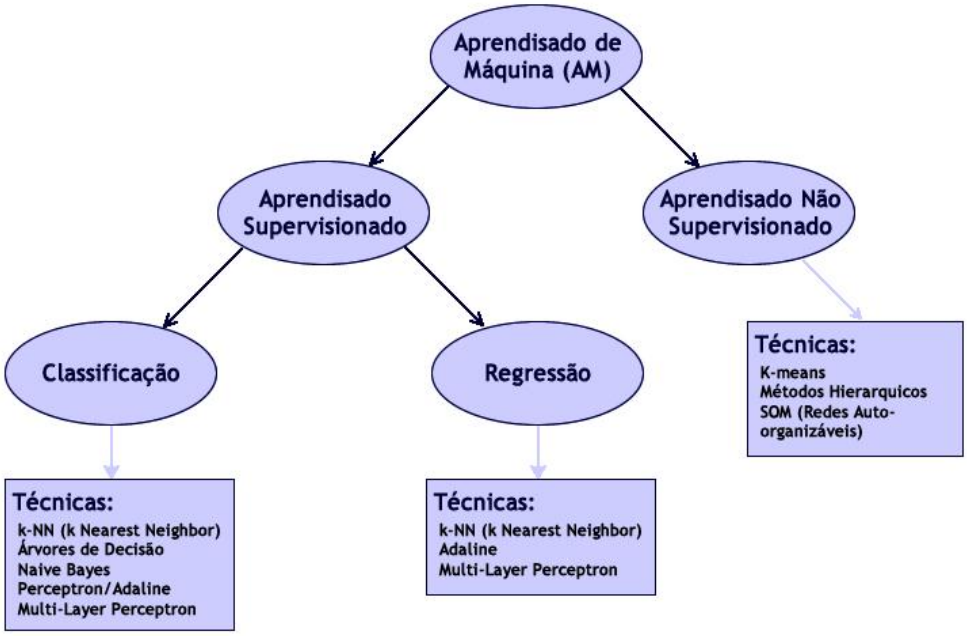

Figura 2 - Tipos de Aprendizado de Máquina.

A Figura 2 mostra os tipos de Aprendizado de Máquina e Técnicas referentes ao tipo. O primeiro nível é o AM que se divide em dois níveis: Aprendizado de Máquina Supervisionado e Aprendizado Não Supervisionado, onde o Aprendizado de Máquina Não Supervisionado tem como exemplo as técnicas: K-meams, Métodos Hierárquicos e SOM (Redes Auto Organizáveis). O Aprendizado de Máquina Supervisionado se divide em dois tipos criando o terceiro nível da Figura 2, onde o primeiro a Classificação que tem com exemplo de técnicas: K-NN (K Nearest Neighbor), Árvore de Decisão, Naive Bayes, Perceptron/Adaline e Multi-Layer Perceptron; e o segundo a Regressão com técnicas como: K-NN, Adaline e Multi-Layer Perceptron.

Esses tipos de aprendizagem de máquina serão melhores definidos e explicados a seguir.

1.2.1 Aprendizagem Supervisionada

Na aprendizagem de máquina supervisionada, o algoritmo de aprendizado (indutor) recebe um conjunto de exemplos de treinamento para os quais os rótulos da classe são conhecidos. Cada exemplo é descrito por um vetor de valores (atributos) e pelo rótulo da classe associada. O objetivo do indutor é construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não classificados. Para rótulos de classe discretos esse problema de categorização é chamado de classificação e para valores contínuos como regressão.

O processo do aprendizado supervisionado parte dos dados de treinamento para o indutor resultando em um classificador, conforme a Figura 3.

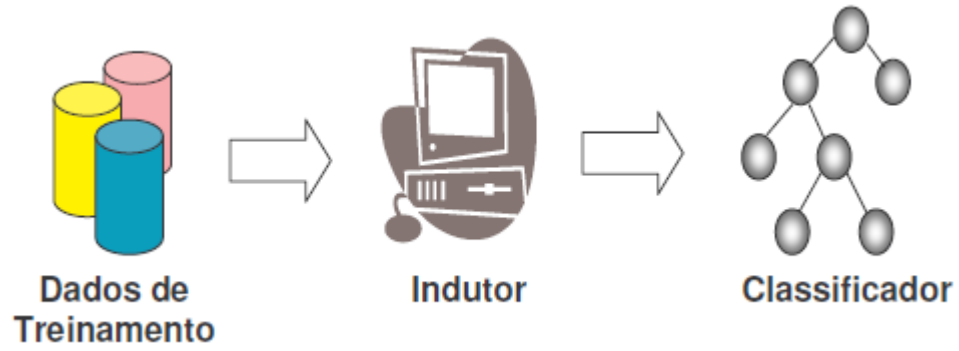

Figura 3 - Processo do Aprendizado Supervisionado.

A Figura 3 apresenta o processo de aprendizado de máquina cujos dados de treinamento são representados pela tabela de exemplos, algoritmos de aprendizado de máquina e geração de um classificador como, por exemplo, a árvore de decisão.

Os dados para treinamento serão apresentados no capítulo 1.4.3 TF-iDF. Dados que servem de entrada para os algoritmos indutores de aprendizagem de máquina supervisionados como, por exemplo, Método J48 que é uma implementação do algoritmo C4.5, Naive Bayes e Support Vector Machines, que serão apresentadas respectivamente nos capítulos 1.6.1, 1.6.2 e 1.6.3. O Classificador será apresentado no Capítulo 1.5 Weka.

1.2.2 Aprendizagem não supervisionada

No aprendizado não supervisionado, o indutor é eliminado, pois é o algoritmo que avalia os conceitos. São apresentadas apenas as entradas, a saída não é conhecida. É utilizada

uma técnica, tal que, para exemplos semelhantes, o sistema responde de forma semelhante formando clusters. Este processo é conhecido como descobridor de regularidades ou redes auto organizadas, devido à propriedade básica do seu funcionamento (BARRETO, 2001).

No aprendizado não supervisionado, são dadas condições para realizar uma medida independente da tarefa da qualidade da representação que o sistema deve aprender, e os parâmetros livres do sistema são otimizados em relação a esta medida. Uma vez que o programa computacional tenha se ajustado às regularidades estatísticas dos dados de entrada, ele desenvolve a habilidade de formar representações internas para codificar as características da entrada e, desse modo, criar automaticamente novas classes (HAYKIN, 2001).

No aprendizado não supervisionado a tabela de exemplos não terá uma classificação e por isso não pode ser treinada por um indutor. Desta forma os algoritmos necessários para gerar um classificador são bem diferentes.

Contudo, para escolher entre aprendizado supervisionado ou não supervisionado para a classificação de textos basta ver se a amostragem de textos está previamente rotulada ou não. Isto por que se eles estão rotulados, os algoritmos supervisionados terão melhor capacidade de aprender, caso não estejam rotulados, os algoritmos não supervisionados tem a capacidade de gerar uma classificação e aprender.

1.3 Preparação dos dados

Para efetuar o processamento digital de textos precisa-se previamente dos dados. O capítulo coleta dos textos explica como foram pesquisados e coletados dados a fim de efetuar a tokenização para fazer testes de aprendizagem de máquina.

1.3.1 Coleta de textos

Para efetuar a coleta de textos para o processamento digital, pode-se usar de diversas fontes tais como: arquivos de fontes acadêmicos, revistas eletrônicas, páginas de internet, artigos científicos, etc. No entanto, a forma de coleta pode mudar de acordo com o tipo de informação e a fonte. Existem, por exemplo, sites que contem conjuntos de textos organizados para processamento digital (também conhecidos como corpus), outro exemplo é a utilização de programas que efetuam copia do conteúdo dos sites salvando em um diretório local formando um corpus. A coleta de textos é a base para construção de exemplos a serem processados pelos algoritmos de Aprendizado de Máquina.

Um exemplo de site que disponibiliza corpus para download é o NILC (http://www.nilc.icmc.usp.br/) do Núcleo Interinstitucional de Linguística Computacional da Universidade de São Paulo (USP) de São Carlos - SP. Neste site, existem diversos trabalhos relacionados à inteligência artificial, assim como também de aprendizado de máquina.

Um exemplo de corpus encontrado no NILC é o TeMário (sigla de 'TExtos com suMÁRIOs') construído com vistas à Sumarização Automática de Textos, no âmbito do Projeto EXPLOSA (EXPLOração de métodos diversos para a Sumarização Automática). Para encontrar informações do Explosa e efetuar o download o corpus TeMário pode-se acessar o link do aluno Thiago Alexandre Salgueiro Pardo da USP: http://www.nilc.icmc.usp.br/~thiago/.

O corpus TeMário é formado por 100 textos organizados, conforme mostra a Tabela 1.

Jornais	Seções	Nº de Textos	Nº Palavras	Média de Palavra\Texto
Folha de São Paulo	Especial	20	12.340	617
	Mundo	20	13.739	686
	Opinião	20	10.438	521
Jornal do Brasil	Internacional	20	12.098	604
	Política	20	12.797	639
	Total	100	61.412	
	Média		12.282	613

Tabela 1 - Características do Corpus de textos-fonte.

A Tabela 1 apresenta números importantes que representam o resultado do corpus TeMário. Informações como fonte das informações, seção, número de textos, número de palavras e média de palavras por texto. Tais dados são importantes para saber características do texto e quais metodologias de processamento interessam ser aplicadas.

Apesar de existirem bases de textos prontas para processamento digital de textos, usar essas bases nem sempre é a melhor opção. Isso por que uma base de conhecimento geralmente tem características baseadas em necessidades. Ao fazer um trabalho com objetivos específicos pode ser mais vantajoso construir uma base de conhecimento mais direcionada.

O TeMário, por exemplo, foi construído para ser utilizado com algoritmos de Aprendizado de Máquina Semi-Supervisionado, ou seja, é uma base de textos que pode não dar o resultado desejado quando processado em Aprendizado de Máquina Supervisionado.

Ao fazer um trabalho com objetivos específicos, pode ser mais vantajoso construir uma base de conhecimento mais direcionada. Determinados domínios não possuem corpora anotado, seja pela inexistência ou pela cobrança pelo acesso.

Uma das maneiras de construir um copora anotado é através da coleta de textos da internet. Existem programas que fazem parte desse tipo de trabalho. Eles são chamados de Mirroring que vem do inglês espelhamento, pois fazem copias fieis de Sites. Um exemplo de programa para o mirroring é o Wget desenvolvido pelo grupo de software livre GNU (GNU, 2010).

O Wget possibilita o uso de diversos parâmetros entre eles: recursividade, fixar domínio específico, fixar um formato de arquivo, definir um tempo de processamento, entre outros. Através desta ferramenta é possível efetuar uma cópia de todas as páginas de um site salvando-as em formato HTML ou outros com as mesmas características, como: shtml, xhtml e htm. Por exemplo, para fazer o Mirror do site “http://www.lancenet.com.br” são necessários parâmetros no programa Wget. Estes parâmetros e o resultado são apresentados na Figura 4 e Figura 5.

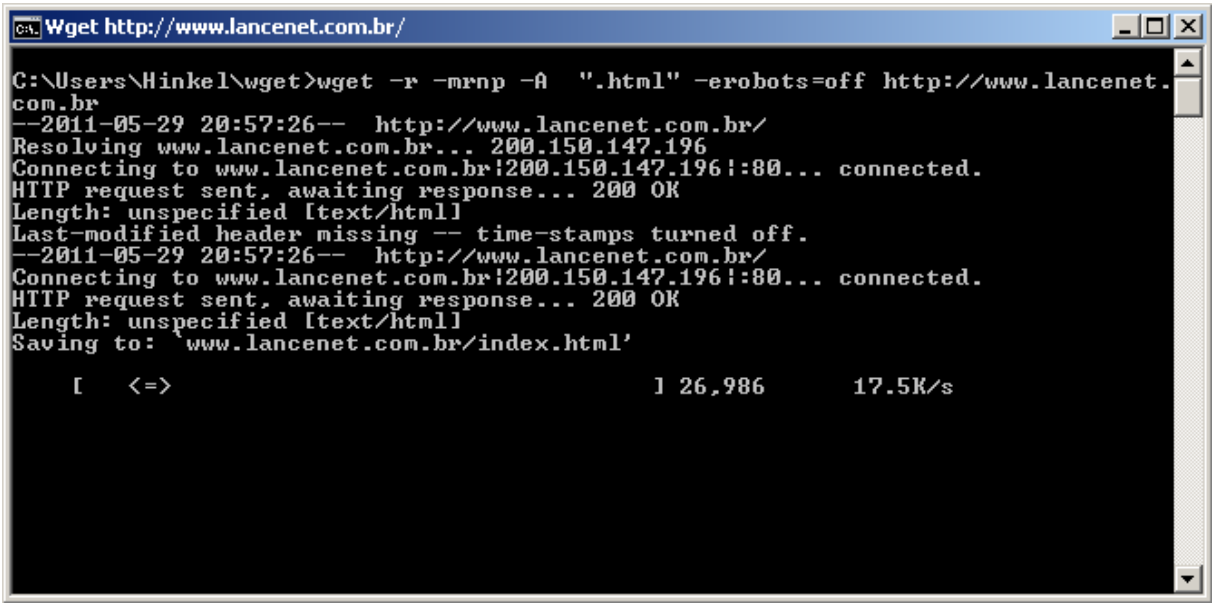

Figura 4 - Exemplo do comando Wget no Microsoft Dos.

A Figura 4 mostra o exemplo do comando Wget sendo executando no Microsoft Dos para baixar arquivos do site http://www.lancenet.com.br. O wget do exemplo tem os seguintes comandos: -r executa em modo recursivo; -mrnp executa o recursivo sem permitir sair do domínio especificado; -A ".html" baixa apenas os arquivos em formato Html; -erobots=off executa o recursivo em diretórios que o arquivo rebot do servidor informa que não pode ser acessível. Os arquivos são salvos no diretório onde o programa é executado.

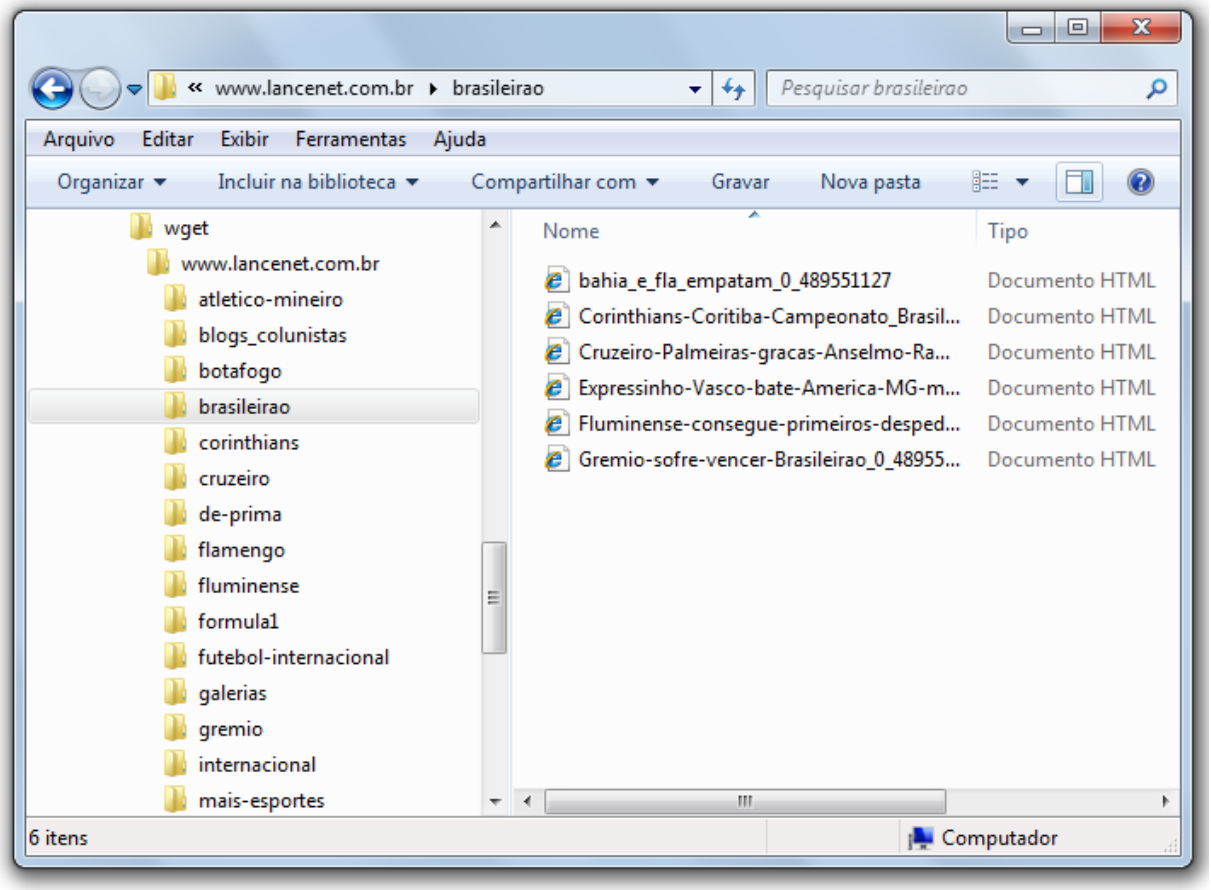

Figura 5 - Exemplo da árvore de pastas resultantes do Wget.

A Figura 5 mostra a árvore de pastas resultantes do Wget no site http://www.lancenet.com.br. Cada diretório é uma sessão do site, como por exemplo: atlético-mineiro, blogs_colunistas, botafogo, etc. Para classificar os textos pode-se pegar cada diretório e trata-lo como uma categoria e os textos do diretório como exemplos.

Como um arquivo em formato HTML é uma linguagem de marcação, seu código fonte possui diversos atributos que são pertinentes somente ao interpretador e não ao processamento digital de textos. Para eliminar esses atributos, uma opção é utilizar programas que convertem arquivos do formato HTML para Arquivo de Texto em formato TXT. Essa

conversão é responsável por eliminar totalmente os códigos de marcação do HTML, restando assim, somente o texto da página. Uma ferramenta que faz essa conversão é o HTML2TEXT desenvolvido pelo grupo de software livre GNU (MBAYER, 2006). No capítulo 3 Execução do projeto, a Figura 13 demonstra o texto divido em três partes sendo a primeira a versão em HTML e a segunda a versão convertida de HTML para TXT. O comando utilizado para fazer a conversão está apresentado na Figura 16– Tela do Html2text sendo executando no Linux Ubuntu..

A nomenclatura dos diretórios e textos pode contribuir para a organização da base de textos. Isso por que é mais fácil construir algoritmos computacionais para percorrem diretórios padronizados. Um bom exemplo disso é separar os documentos por categoria, onde cada diretório conteria o nome da categoria. Desta forma podem-se aplicar algoritmos de pré-processamento de texto independentemente para cada categoria e analisar os resultados separadamente. Considerando o exemplo da Figura 5, podemos identificar as seguintes categorias: atletico_mineiro, blogs_colunistas, botafogo, brasileirao, corinthians, de-prima, cruzeiro, flamengo, fluminense, formula1, futebol-internacional, galerias, gremio, internacional, etc.

Com a coleta de textos organizados pode-se aplicar Tokenização para separar os termos do texto e processar esses termos com técnicas de Pré-Processamento a fim de construir uma tabela de exemplos.

1.3.2 Tokenização

O processo de tokenização possibilita a separação dos elementos constituintes do texto, identificando cada palavra que poderá compor a representação estatística do exemplo. O primeiro passo nesse processo de separação dos elementos do texto ocorre com a quebra do texto em palavras, mais precisamente tokens, através da identificação de tokens delimitadores se obtém as melhores características presentes no texto.

Para gerar uma lista de tokens delimitadores se faz necessário observar características do texto, como espaços, vírgulas, pontos, caracteres especiais, entre outros. Contudo essas caraterísticas podem mudar de acordo com o tipo de texto, fonte e a língua. Pode-se citar a seguinte lista de exemplos de tokens delimitadores cuja fonte é um site de notícias da internet de língua portuguesa e a categoria das notícias é esporte: “ ”, “\n”, “ \t”, “ \r”, “.”, “;”, “:”, “!”, “?”, “(”, “)”, “<”, “>”, “[”, “]”, “+”, “\”. Como exemplo, tem-se a frase a seguir:

- “O sopro de vento empurrou lentamente a velha porta entre-aberta.”.

O resultado é a lista de tokens considerando espaços como delimitadores: “o”, “sopro”, “de”, “vento”, “empurrou”, “lentamente”, “a”, “velha”, “porta”, “entre-aberta”.

Um desafio da tokenização é a separação de termos compostos como, por exemplo, o nome da cidade de São Paulo. Exemplo de frase com termos compostos:

- “A italiana Michela Ciminello veio a São Paulo para estudar no final de 2010”.

Lista de tokens desejável para frase: “a”, “italiana”, “Michela Ciminello”, “veio”, “a”, “São Paulo”, “para”, “estudar”, “no”, “final”, “de”, “2010”.

A tokenização dos termos do texto cria o vinculo necessário para que o sistema consiga, através dos atributos delimitadores, montar estruturas de dados. Essas estruturas de dados são conhecidas como vetores ou até mesmo arrays. As estruturas de dados são utilizadas para separar uma base de dados em termos e, desta forma, processar termo a termo sem perder a referência das bases de dados.

Após efetuar o processo de Tokenização, a base de dados fica pronta para efetuar a etapa de pré-processamento, que é apresentada no próximo capítulo.

1.4 Pré-Processamento

A fase de pré-processamento inicia tão logo os dados são coletados e organizados na forma de um conjunto de dados. O principal objetivo desta fase de pré-processar os dados é gerar um novo conjunto de termos que possa aprimorar os resultados do algoritmo de extração de conhecimento.

É importante salientar que o pré-processamento é um processo semiautomático, isso porque, a pessoa que for tratar os dados deve identificar características que podem ser aperfeiçoadas. Desta forma, os processos de pré-processar os textos podem ser feitos várias vezes até que a base de exemplos seja suficientemente boa para a extração de conhecimento.

Neste capítulo serão apresentados três tópicos de pré-processamento de dados, são eles: Lista de Stopwords, Stemming e TF-iDF.

1.4.1 Lista de Stopwords

O primeiro tópico apresentado no pré-processamento dos dados é a identificação das palavras que podem ser desconsideradas nos passos posteriores do processamento dos dados.

Nesta fase tenta-se retirar tudo que não agrega valor ao texto, resultando assim, em uma lista de palavras ou conjuntos de caracteres a serem excluídos. Este conjunto de palavras é chamado de Stopwords.

Stopwords são palavras que geralmente se repetem muito e, além disso, não apresentam um conteúdo semântico significativo no contexto em que se encontram no texto, ou seja, são palavras consideradas irrelevantes na análise do texto. Geralmente trata-se de palavras auxiliares ou conectivas (por exemplo: e, para, então, muito, eles, elas), que não fornecem nenhuma informação que venha a representar o conteúdo dos textos. Na construção de uma lista de Stopwords são acrescentadas palavras como advérbios, preposições, pronomes, artigos e também podem ser adicionadas palavras que apresentam uma incidência muito alta em uma coleção de documentos (i.e. presente em todos os documentos) e que, portanto não apresentam influência na categorização dos textos (DIAS, 2005).

Para retirar a lista de Stopwords do texto pode-se utilizar da técnica de tokenização. É possível aplicar a tokenização nos textos formando várias estruturas de dados e em seguida aplicar a lista de Stopwords formando uma única estrutura. Tendo estrutura referente aos textos e a estrutura referente às Stopwords pode-se construir um algoritmo que combine as mesmas, de forma que ao processar em um texto um termo que contenha uma Stopword, esse termo é eliminado. Desta forma o texto fica mais leve para os futuros processamentos e com termos mais relevantes.

Na Tabela 2, é apresentada uma lista de artigos gramaticais da língua portuguesa retirados do site Brasil Escola (BRASIL ESCOLA, 2002). Esta lista de artigos pode ser usada como um exemplo de lista de Stopwords.

.

O	A	da	Nos	Per
os	ao	das	na	pelo
a	aos	dum	nas	pelos
as	à	duns	num	pela
um	às	duma	nuns	pelas
uns	de	dumas	numa	
uma	do	em	numas	
umas	dos	no	por	

Tabela 2 - Exemplo de Lista de Stopwords com artigos gramaticais em português.

A Tabela 2 é um exemplo simplificado de lista de Stopwords, e pode ser usada para exemplificar a aplicação da remoção de palavras em uma frase. Por exemplo:

- “O sopro de vento empurrou lentamente a velha porta entre-aberta.”.

A frase sem as Stopwords fica assim:

- “sopro vento empurrou lentamente velha porta entre-aberta”.

Nesta frase os artigos: “o”, “de” e “a” foram devidamente removidos.

O pré-processamento, neste caso, trata de retirar cada uma das palavras relacionadas na stoplist fazendo com que a aplicação uma base de dados mais limpa e confiável. No próximo capítulo serão tratados algoritmos de radicalização conhecidos na bibliografia como Stemming.

1.4.2 Stemming

Outro tópico importante de pré-processamento e cuja natureza é o Processamento de Linguagem Natural (PLN) são os algoritmos de Radicalização ou também denominados de Stemming, consistem em uma normalização linguística, em que as palavras e suas variantes são simplificadas a uma forma comum, em um “quase radical” ou stem (ORENGO, 2001), resultando na diminuição do número de palavras armazenadas na base de dados. É válido salientar que o radical resultante da normalização de uma palavra não é necessariamente igual a sua raiz linguística.

Esse processo é feito através de um algoritmo que processa as palavras seguindo o fluxo do diagrama da Figura 6 (ORENGO, 2001).

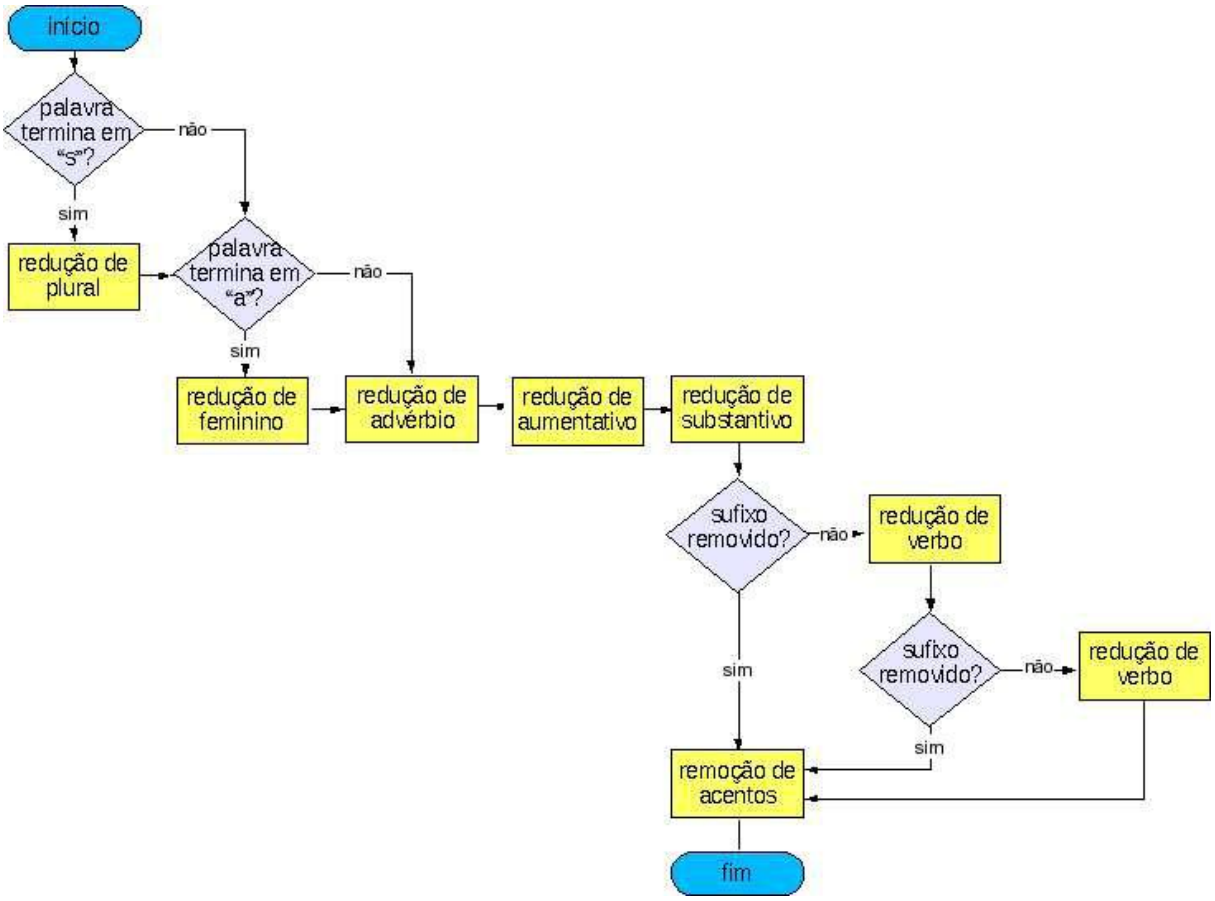

Figura 6 - Passos do algoritmo de radicalização.

O algoritmo da Figura 6 (ORENGO, 2001), leva em conta as classes morfológicas das palavras, executando uma série de passos de remoção de sufixos conhecidos. Os passos são aplicados na seguinte sequência: 1. Redução do plural; 2. Redução do feminino; 3. Redução do advérbio; 4. Redução do aumentativo e do diminutivo; 5. Redução das formas nominais; 6. Redução das terminações verbais; 7. Redução da vogal temática; e 8. Remoção dos acentos. Por exemplo, a redução ao seu radical da seguinte frase:

- "O sopro de vento empurrou lentamente a velha porta entre-aberta.".

Radicais: "sopr", "vent", "empurr", "lent", "velh", "port", "entr" e "abert".

Uma maneira de aplicar esses conceitos é montar um dicionário de radicais (Dicionário de Stemmer). Para isso, um conjunto de palavras que corresponde a um determinado radical é organizado de forma que o Software ao processar aquela palavra a substitua pelo seu radical. Desta forma, o resultado é uma lista de palavras reduzida e confiável.

O próximo tópico de pré-processamento não trata as palavras individualmente como é caso da lista de Stopwords e Stemming. Isso porque, o TF-iDF tem por objetivo criar uma relação de atributos a serem processados pelos algoritmos de extração de conhecimento.

1.4.3 TF-iDF

O terceiro tópico de pré-processamento é cálculo de frequência de termos no texto através do método TF-iDF. Este método, TF-iDF, leva em consideração a frequência do termo em relação a todos os documentos da coleção, o peso é associado ao termo na proporção da frequência do termo no documento e na proporção inversa da quantidade de documentos em que o termo aparece.

Para que a aplicação do TF-iDF seja possível é necessário abstrair algumas informações do texto e aplicar a fórmula a baixo, conforme é explicado a seguir:

$$a_{ik} = f_{ik} * \log (N / n_i)$$

Onde N representa a quantidade de documentos da base de dados, n_i a quantidade de documentos em que o i-ésimo termo aparece, f_{ik} é a frequência do termo no documento e o resultado a_{ik} é o peso atribuído ao termo. O modelo TF-iDF considera a frequência de cada termo em todos os documentos como medida inversa da sua capacidade de representar especificamente cada documento (THORSTEN, 1997), ou seja, quanto mais documentos o termo aparecer menos representativo na especificação do documento o mesmo será. Se um termo aparece cinco vezes mais em um documento do que em outro documento, não se pode concluir nada a respeito de sua relevância, pois o tamanho dos textos pode contribuir para a ocorrência maior em um deles. Todavia, se um termo aparece em cinco vezes mais documentos da coleção que os documentos do que outro termo aparece, esse termo que aparece em menos documentos terá um maior peso de decisão na categorização. Como por exemplo, o cálculo do TFiDF de seis documentos:

- Documento 1: Astronautas vão trabalhar por mais de 6 horas no espaço.
- Documento 2: Astronautas da Endeavour iniciam sua última caminhada espacial
- Documento 3: Bovespa tem quarto pregão seguido de alta e fecha semana com ganho
- Documento 4: Bovespa opera em alta no pregão desta quinta-feira

- Documento 5: Lula defende mobilização por reforma política, dizem sindicalistas
- Documento 6: Lula defendeu, segundo os sindicalistas, o barateamento das eleições.

TFiDF	Alta	Astronautas	Bovespa	De	Lula	Mobilização	No	Opera	Por	Pregão
DOC 1	0	0.0545	0	0.0545	0	0	0.0545	0	0.0545	0
DOC 2	0	0.0545	0	0	0	0	0	0	0	0
DOC 3	0.0545	0	0.0545	0.0545	0	0	0	0	0	0.0545
DOC 4	0.0545	0	0.0545	0	0	0	0.0545	0.1091	0	0.0545
DOC 5	0	0	0	0	0.0545	0.1091	0	0	0.0545	0
DOC 6	0	0	0	0	0.0545	0	0	0	0	0

Tabela 3 - Tabela de Exemplos resultante do cálculo de frequência TF-iDF.

A Tabela 3 apresenta os resultados do cálculo de frequência TF-iDF dos seis documentos acima citados. Para entender o cálculo pode-se usar o exemplo do documento um, o termo “Dentro” onde o resultado do TF-iDF é 0,0538. Para chegar nesse valor foi realizado o seguinte cálculo:

Cálculo: TF-iDF = (TF / TFT) * (DFT / DF);
Onde:
TF = número de vezes que o termo aparece dentro do documento. Resultado: 1.
TFT = número total de termos de todos os documentos. Resultado: 55.
DFT = número total de documentos. Resultado: 6.
DF = número de documentos que contém o termo. Resultado: 2.
Calcula-se: TFiDF do Astronautas (1 / 55) * (6 / 2) = 0,0545.

O método TF-iDF pode ser organizado em forma de tabela de exemplos servindo de suporte para os algoritmos de aprendizado de máquina. Para processar os exemplos em algoritmos de aprendizado de máquina pode-se usar a ferramenta Weka, que será apresentada no capítulo posterior.

1.5 WEKA

A ferramenta Weka (Waikato Enviroment for Knowledge Analysis) foi desenvolvida na Universidade de Waikato, Nova Zelândia. Este software implementa um grande conjunto de ferramentas para preparação dos dados, aprendizado de máquina e validação/verificação dos resultados.

O Weka dispõe de uma interface gráfica para a maioria de suas funções, permitindo ao usuário a seleção da fonte de dados, remoção dos atributos e seleção automática e manual de atributos que serão utilizados na descoberta de conhecimento. Além da preparação dos dados, o Weka oferece alguns algoritmos classificadores como o ZeroR, o Id3, o PART e o J48, algoritmos de descoberta de regras de associação como o Apriori e algoritmos de agrupamento como o K-Means, COBWEB e EM. Além da interface gráfica, o Weka disponibiliza uma série da APIs (Application Program Interface) que permitem a utilização de seus algoritmos.

Weka suporta alguns formatos de arquivos, como CSV e C4.5, além do formato próprio da ferramenta, o ARFF. Este é um formato de arquivo em texto, contendo atributos e seus respectivos dados, para serem manipulados pela aplicação. Pode-se utilizar a tabela de exemplos conforme a Tabela 3 formatando-a no layout do ARFF conforme a Figura 7.

```
1   @RELATION Classificador
2
3   @ATTRIBUTE Alta NUMERIC
4   @ATTRIBUTE Astronautas NUMERIC
5   @ATTRIBUTE Bovespa NUMERIC
6   @ATTRIBUTE De NUMERIC
7   @ATTRIBUTE Lula NUMERIC
8   @ATTRIBUTE Mobilizaca NUMERIC
9   @ATTRIBUTE No NUMERIC
10  @ATTRIBUTE Opera NUMERIC
11  @ATTRIBUTE Por NUMERIC
12  @ATTRIBUTE Pregao NUMERIC
13
14  @ATTRIBUTE class { ciencias,politica,economia }
15
16  @data
17  0,0.0545,0,0.0545,0,0,0.0545,0,0.0545,0,ciencias
18  0,0.0545,0,0,0,0,0,0,0,0,ciencias
19  0.0545,0,0.0545,0.0545,0,0,0,0,0,0.0545,politica
20  0.0545,0,0.0545,0,0,0,0.0545,0.1091,0,0.0545,politica
21  0,0,0,0,0.0545,0.1091,0,0,0.0545,0,economia
22  0,0,0,0,0.0545,0,0,0,0,0,economia
```

Figura 7 - Tabela de exemplos formatada para o WEKA em ARFF.

A Figura 7 apresenta a Tabela 3 formatada em ARFF para ser processada na ferramenta WEKA. O formato ARFF tem os seguintes atributos que devem ser configurados: @RELATION: referente ao titulo do arquivo; @ATTRIBUTE: lista de termos e seu tipo de valor (INTEGER, REAL e NUMERIC para números, STRING para letras, entre outros); @ATTRIBUTE class: onde devem ser colocados os entre chaves as classificações dos textos; @data: lista de valores separados por vírgula e no final de cada linha deve constar a categorização referente aos valores.

Para processar os textos é importante conhecer como funciona a ferramenta Weka. Conforme a Figura 8.

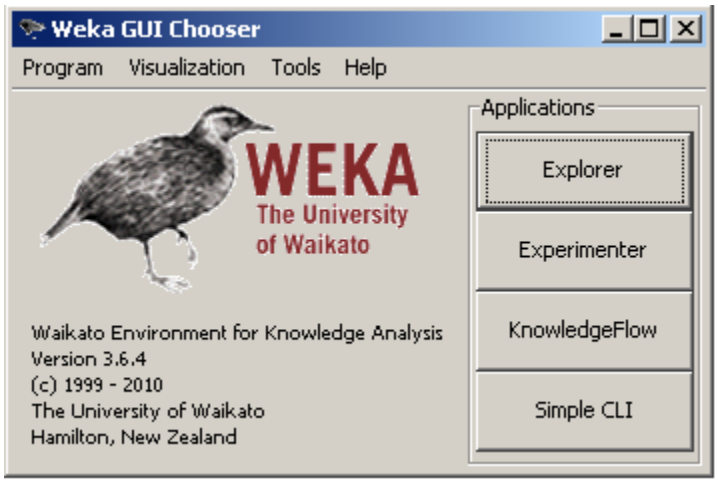

Figura 8 - Tela inicial da ferramenta Weka.

A Figura 8 é a tela inicial da Ferramenta de processamento de textos Weka. O ambiente que contem funcionalidades para pré-processamento, análise, classificação e pré-visualização de resultados é o Explorer. Já o Experimenter contem ferramentas de testes estatísticos entre esquemas de aprendizagem, o KnowledFlow é semelhante ao Explorer com o auxilio de draw-and-drop e a vantagem de suportar a aprendizagem incremental, por fim o Simple CLI onde é possível incluir execuções diretamente na linha de comando pelo Sistema Operacional.

Para entender como é organizada a tela de Explorer pode-se observar a figura 9.

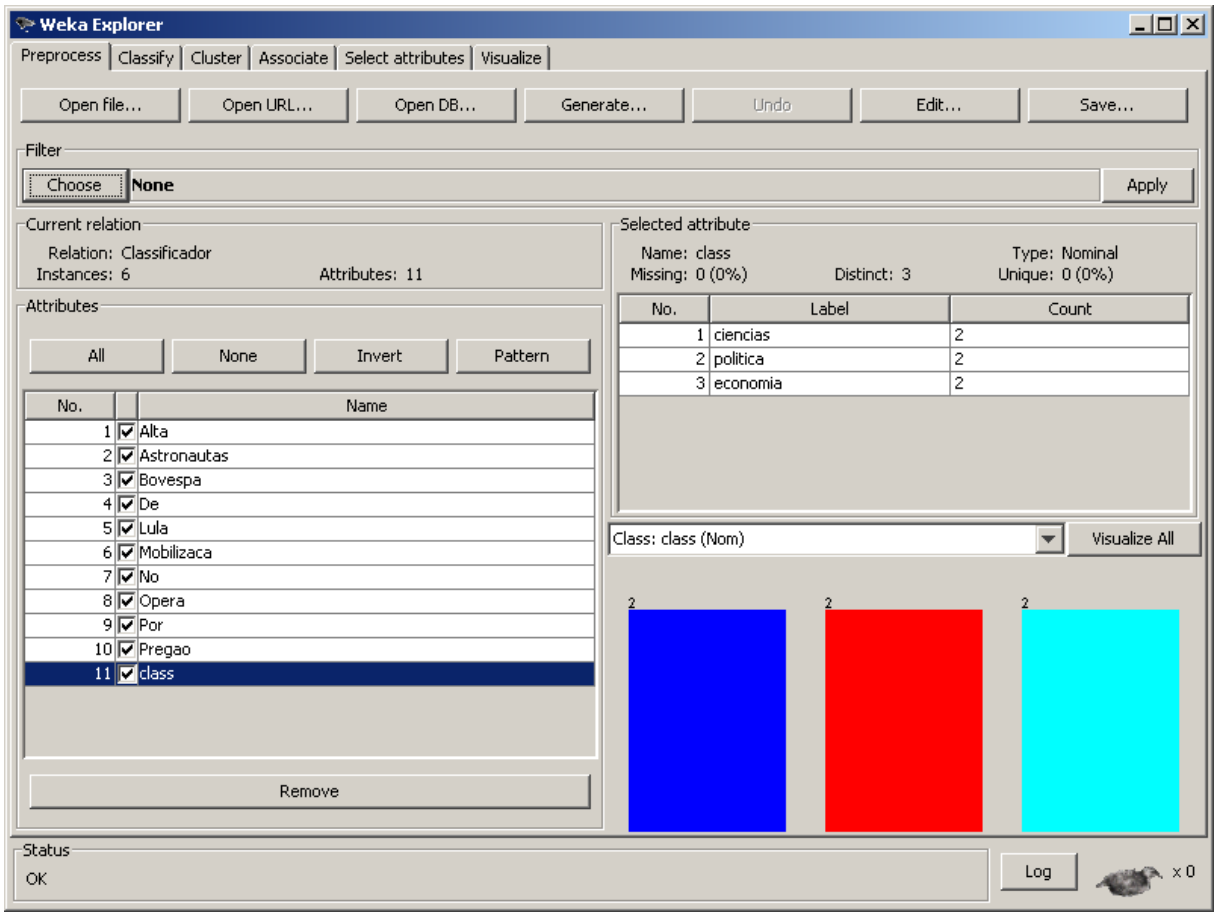

Figura 9 - Tela Explorer da ferramenta Weka.

A Figura 9 apresenta a tela Explorer da ferramenta Weka onde é possível fazer o upload do arquivo ARFF cujo layout demonstrado na Figura 7. Para fazer o upload basta clicar em “Open File...”, selecionar o arquivo e enviar. Depois se pode escolher métodos de filtros, selecionar os termos que deseja processar e visualizar o total de classificações.

No Capítulo de Algoritmos de Aprendizado de Máquina será utilizada a ferramenta Weka para demonstrar como processar os exemplos e gerar o classificador.

1.6 Algoritmos de Aprendizado de Máquina

Os algoritmos de Aprendizado de Máquina (AM) processam a tabela de exemplos gerada pelos métodos de pré-processamento como, por exemplo, a Tabela 3. Ao processar essa tabela de exemplos, o método de extração de conhecimento gera um arquivo final que serve de base para que o sistema classificador classifique novos exemplos.

Alguns algoritmos AM mais populares para classificar textos e que serão apresentados neste capítulo são: Método j48, Support Vector Machine (SVM) e Naive Bayes.

O Método j48 é o mais comum em trabalhos de pesquisa, pois ele possui uma característica que possibilita uma boa análise de seus resultados. Essa característica é que seu resultado é uma árvore de decisão, que pode ser visualizada e compreendida pelo usuário, diferente do resultado dos algoritmos SVM e Naive Bayes que geram o que se chama de caixa preta, ou seja, um resultado que só pode ser interpretado pelo programa classificador.

Ao montar uma tabela de exemplos podem-se delimitar características que implicam em um melhor resultado com determinados algoritmos de AM. Um exemplo disso é a quantidade de atributos, isso porque, quando a dimensionalidade de atributos é muito extensa o classificador SVM apresenta melhores resultados, diferente de uma dimensionalidade menor, cujos algoritmos j48 e Naive Bayes apresentam uma resposta melhor.

A compreensão desses algoritmos pode-se ver a seguir, começando pelo Método j48 e sua árvore de decisão, seguido pelas Redes Baysianas (Naive Bayes) e Support Vector Machine.

1.6.1 Método j48

O Método j48 é uma implementação na linguagem de programação Java baseada no algoritmo C4.5 desenvolvida na linguagem C e sua base de funcionamento se inspira no método TF-iDF (capítulo 3.2.3) e no algoritmo ID3, isso porque, sua estratégia é intitulada “divisão para conquistar” (QUINLAN, 1986). Especificamente o método j48 procura gerar uma arvore de decisão a partir de uma abordagem recursiva de particionamento da base de dados (GOLDSCHIMDT, 2005). A árvore de decisão é um método simples de representar o conhecimento extraído a partir de uma base de dados. Sua função é sistematizar os dados facilitando a decisão a ser tomada (RUSSEL, 2002).

Para gerar um classificador através do Weka utilizando o Método J48 pode-se utilizar do ambiente Explorer conforme a Figura 10.

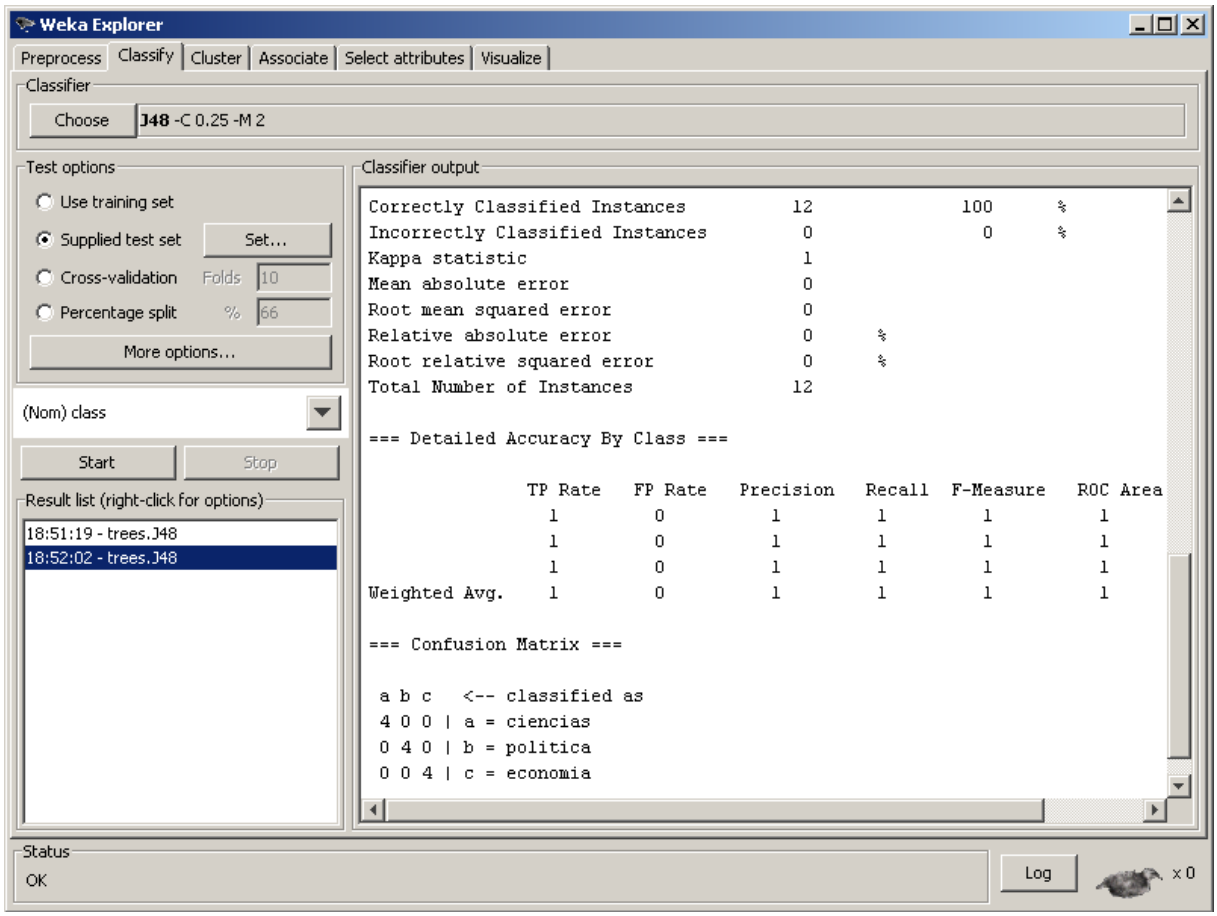

Figura 10 - Tela Classify da ferramenta Weka.

A Figura 10 representa a tela Classify (Classificação) da ferramenta Weka, que tem por objetivo processar a tabela de exemplos construindo um classificador. Depois de fazer o upload do arquivo e escolher os termos, pode-se utilizar dos métodos de classificação. Para escolher um método basta clicar na opção *Classifier* > *Choose*, onde é possível escolher um dos algoritmos de aprendizado de máquina. As opções Test Options são importantes para saber que tipo de teste ou treinamento o sistema vai fazer para realizar o treinamento dos dados. Uma opção interessante é o Cross-validation onde o algoritmo divide os exemplos em duas partes iguais e faz teste combinando exemplo a exemplo.

Após a escolha da opção de teste a ser realizada pelo classificador pode-se clicar no botão Start. Assim, tem-se como resultado no lado direito da Figura 7, a saída do classificador (Classifier Output) que mostra os dados estatísticos para avaliação do modelo gerado. Na imagem a classificação correta das instâncias fica em 100% porque o número de exemplos é muito pequeno, mas com um número bastante grande, esse resultado pode variar.

O Weka possibilita a visualização gráfica da árvore de decisão gerada pelo algoritmo J48 conforme a Figura 11. Para visualizar esta opção da ferramenta Weka é necessário clicar com o botão direito sobre o item que compõem os resultados processados (Result list), e escolher a opção “Visualize tree”.

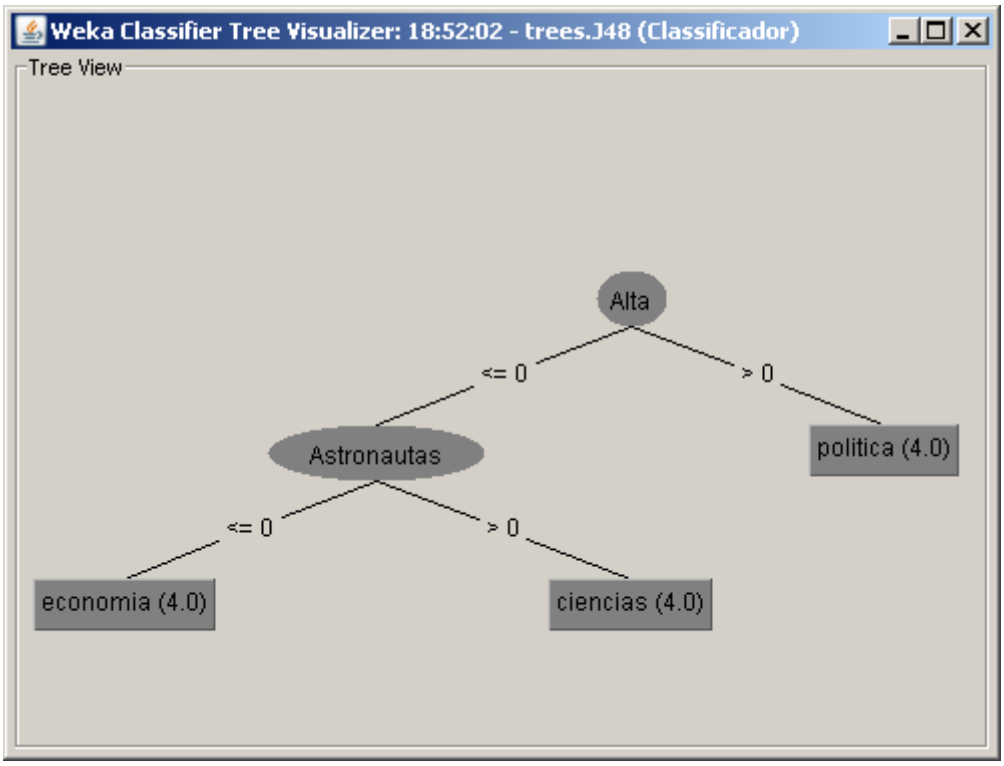

Figura 11 - Tela Tree Visualizer da ferramenta Weka.

A Figura 11 apresenta a árvore de decisão resultante do processamento do método de aprendizado de máquina J48 processada na ferramenta Weka. A árvore de decisão é realmente semelhante a uma árvore, contudo virada 180 graus onde a raiz está em cima e vai descendo com galhos e folhas. Sua estrutura é composta de nós de decisão, que contém testes sobre algum atributo do conjunto de dados, e folhas que correspondem a uma classe, ou seja, um diagnóstico ou classificação do atributo predito. Os atributos que devem ser colocados na raiz ou nos nós de decisão são escolhidos por um algoritmo específico.

Visualizando a Figura 8 atribui-se ao algoritmo classificador o seguinte processo: verifica se texto contém atributo “Alta”, se sim sua classificação é política, se não, verifica se contém o atributo “Astronautas”, se sim sua classificação é ciências, se não, é economia.

Desta forma, o método J48 processa uma tabela de exemplos e constrói uma árvore de decisão que pode ser traduzida a qualquer linguagem computacional e funcionar como um classificador.

1.6.2 Naive Bayes

Algoritmos de Naive Bayes, também conhecidos como algoritmos baysianos são classificadores estatísticos supervisionados, cuja função é chegar à probabilidade de um exemplo pertencer a uma determinada classe. Para isso, os métodos baysianos trabalham com uma abordagem probabilística para quantificar vetores de valores (atributos), isso porque melhores decisões podem ser tomadas por meio da análise dessas probabilidades e do conjunto de exemplos de treinamento disponíveis, ou seja, cada exemplo pode definir se deve aceitar ou descartar uma hipótese caso diminua ou aumente a probabilidade associada a essa hipótese.

Naive Bayes é uma das muitas técnicas populares que pode ser usada para classificação de textos eficientemente. Considerado algoritmo de aprendizado indutivo eficiente e eficaz para a AM e mineração de dados. Em alguns domínios o desempenho tem mostrado compatível com aprendizado de redes neurais e árvore de decisão. Em pró da eficácia e simplicidade, os algoritmos baysianos são bastante utilizados na classificação de textos (MITCHELL, 1997).

Naive Bayes é chamado assim a partir da independência condicional da classe, isto é, o efeito do valor do atributo de uma determinada classe é independente dos outros atributos e isso torna os algoritmos de redes baysianas considerados simples, de onde vem o Neive (Simples).

Para utilizar método de aprendizado de máquina baysiano na ferramenta Weka (Figura 7) basta selecionar o item "*Classifier"* > "*Choose"*, e escolher dentro do agrupamento "Bayes" a opção "Naive Bayes". Depois de escolher e processar o texto pode-se ver o método de organizar as informações atribuindo pontuações individuais para cada termo dentro de uma categoria.

O teorema de bayes tenta modelar a probabilidade de cada novo exemplo afim de que ela possa pertencer à determinada classe. Para isso, uma estratégia é chamada de Maximun a Posteriori (MAP) onde X (x1,x2, ..., xM) seja um exemplo qualquer e Y (c1,...,cL) o conjunto de classes possíveis para X, a probabilidade de X pertencer a uma classe Y.

Com essa dinâmica pode-se estimar com facilidade que o exemplo pertence à determinada classe, porem essa atribuição não é determinante quando processado a um grande número de exemplos e de parâmetros de classificação.

Isso ocorre pela independência da classe entre os atributos que descrevem os exemplos, o que diminui bastante o número de parâmetros que devem ser estimados (MITCHELL, 1997).

Mesmo com uma amostragem não tão fiel as características de uma determinada classificação, os algoritmos da família Naive Bayes costumam resultar em classificadores bastante confiáveis. E essa confiança só é possível por critérios de valor de confiança utilizados nos algoritmos.

A metodologia dos algoritmos baysianos e o método j48 trabalham muito bem quando a dimensionalidade de atributos é pequena, diferente do método Support Vector Machine, que será demonstrado a seguir, que trabalha melhor quando a dimensionalidade de atributos é muito extensa.

1.6.3 Support Vector Machines

Support Vector Machines (SVM) é um conjunto de algoritmos supervisionados que podem ser aplicados para obtenção de classificadores lineares e binários, ou seja, classificadores que separam os exemplos em dois possíveis resultados, sendo eles verdadeiro ou falso, positivo ou negativo, entre outros. Através desta separação, se alcança uma fronteira linear (hiperplano), posicionado esta fronteira entre os exemplos das duas classes conforme a Figura 12. A distância entre os exemplos e o hiperplano separador pode ser utilizada como indicador de confiança do algoritmo de Support Vector Machines (VAPNIK, 1998).

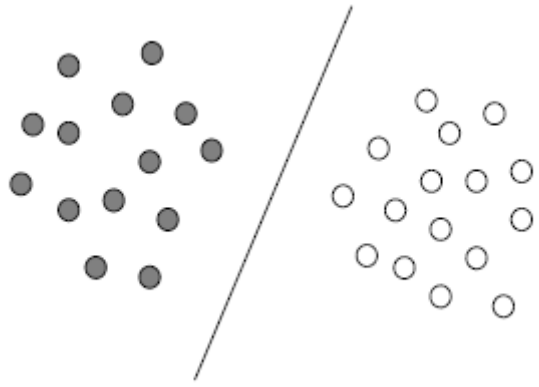

Figura 12 - Exemplo de Support Vector Machine.

A minimização de erros de classificação no conjunto de treinamento e a maximização da margem hiperplano são dois princípios que ajudam a limitar superiormente o erro do classificador em exemplos não vistos durante o treinamento. Porém, esses dois princípios podem ser vistos como um problema de otimização. Considere um conjunto de treinamento linearmente separavam com N exemplos rotulados {(X1 Y1), ..., (XN YN)}. Cada elemento do conjunto de treinamento tem o rótulo da classe igual a+1 para a classe positiva e a-1 para a classe negativa. Pode-se mostrar que o hiperplano W * X + b que resolve o problema de otimização minimiza os erros no conjunto de treinamento e maximiza a margem. Esse problema de otimização é chamado de forma primal (VAPNIK, 1998).

Porém, normalmente a forma primal é transformada em outro problema, chamado de dual, que pode ser resolvido por diversos algoritmos, entre eles o Sequential Minimal Optimization (SMO). E assim, existem outras modificações que procuram aperfeiçoar os resultados diminuindo o ruído. A modificação Soft Margin do SVM (CORTES e VAPNIK, 1995) aumenta consideravelmente o desempenho do processamento do classificador, porém, seus resultados não são perfeitamente linearmente separáveis por conta de algum erro de rotulação e de ruído nos dados.

Para utilizar método de aprendizado de máquina Support Vector Machine na ferramenta Weka (Figura 7) basta selecionar o item "*Classifier"* > "*Choose"*, e escolher dentro do agrupamento "Functions" a opção "LibSVM". Depois de escolher e processar os exemplos pode-se ver os resultados no Classifier Output.

Podem-se utilizar algoritmos combinados com a Ferramenta Weka para executar o classificador caixa preto gerado pelos métodos SVM e o Naive Bayes. Contudo não é tão simples quanto o Método J48 que gera uma árvore de decisão aplicável em qualquer linguagem de programação e, por esse motivo, foi o método escolhido para aplicar no protótipo.

2 DESCRIÇÃO DO PROJETO

Projeto do trabalho de conclusão surgiu da intenção de criar ferramentas que auxiliassem na construção de textos para um site de notícias. Tendo em vista que todo site de notícias possui categorizações previamente definidas, buscou-se automatizar esse processo facilitando o trabalho de quem publica textos e evitando erros. Sendo que tão importante quanto publicar textos é tornar esses textos acessíveis em redes de busca, para isso um atributo importante é selecionar os termos mais relevantes do texto os quais também se procurou fazer de modo automático.

Como a internet possui um número de usuários muito amplo se torna mais difícil criar um site de notícias onde apenas profissionais da área publiquem textos. É muito comum pessoas sem nenhum conhecimento acadêmico jornalístico publicarem textos diariamente na internet, seja por meio de blogs ou até mesmo redes sociais. O fato é que esses textos ganharam seu espaço e relevância através do grande número de acessos diários que recebem. Tendo em vista a importância de auxiliar jornalistas na categorização de textos, muito mais importante é auxiliar pessoas que nunca tiveram nenhuma especialização no assunto.

Para construir uma ferramenta que auxilie pessoas a categorizar e selecionar atributos mais importantes do texto foi feito um estudo em técnicas e algoritmos classificadores.

Através do estudo foi possível observar diversos trabalhos realizados para classificação de textos utilizando aprendizado de máquina supervisionada. Os trabalhos desenvolvidos tinham finalidades distintas, contudo tinham características semelhantes ao objetivo do trabalho, como por exemplo: possuírem categorias previamente definidas e ter a possibilidade de utilizar um número grande de exemplos.

Para trabalhar com Aprendizado de Máquina Supervisionado foi necessário pesquisar sobre técnicas de aprendizado de máquina, coleta de textos, processamento da linguagem natural, cálculos de frequência, algoritmos classificadores e árvores de decisão.

A construção do protótipo seguiu a mesma sequência dos estudos, pois foram necessários a coleta dos textos para a construção do corpus, o processamento da linguagem natural para efetuar um pré-processamento dos textos, os cálculos de frequência para a aplicação do TF-iDF na tabela de exemplos, os algoritmos classificadores para chegar até a ferramenta Weka e utilizar do Método J48 para construção do classificador e o estudo da árvore de decisão para implementar o classificador no algoritmo desejado conforme a Figura 13.

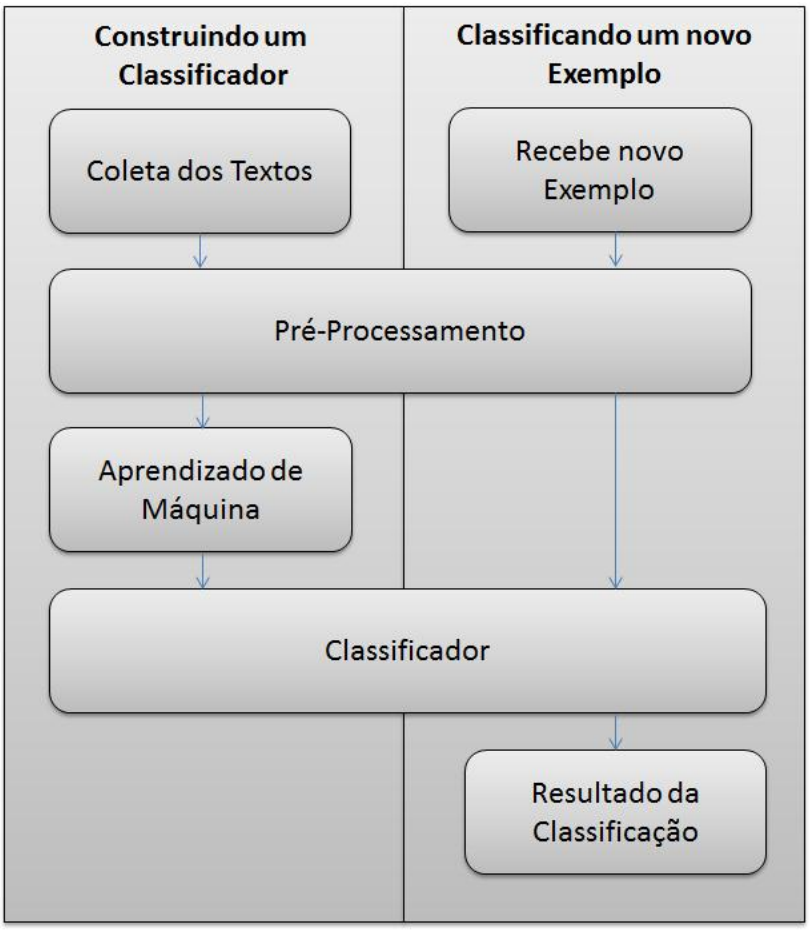

Figura 13– Esquema do Processo para a construção de um Classificador e do Processo para a classificação de novos exemplos.

A Figura 13 apresenta um esquema que demonstra a ordem dos processos para a construção de um classificador e a ordem dos processos para classificar um novo exemplo. Para a construção de um novo exemplo são necessários quatro passos que são: coleta dos textos, pré-processamento, aprendizado de máquina e, por fim, o classificador. Para a classificação de um novo exemplo também são necessários quatro passos, contudo a sequência é a que segue: recebe novo exemplo, pré-processamento, classificador e resultado da classificação.

A coleta dos textos para a construção de um classificador foi feita através da copia de textos de sites de notícias. Para efetuar a cópia dos textos de forma automática foi utilizada a ferramenta Wget. Os textos foram organizados em formato HTML separados por pastas as quais representam suas categorias. Depois de organizar os arquivos html foi utilizado a ferramenta HTML2TEXT para converter os arquivos de HTML para TXT. A seguir foram

utilizadas técnicas de programação na linguagem Php para extrair do arquivo apenas o texto da notícia eliminando outras informações contidas no documento.

No pré-processamento foram utilizadas técnicas de tokenização para dividir os termos do texto dos textos. Depois foi removida a lista de stopwords do texto. A seguir, utilizada a técnica de Stemming para reduzir textos do plural para o singular. Então foi utilizado o cálculo de frequência TF-iDF para construir a tabela de exemplos.

No aprendizado de máquina foi utilizada a ferramenta Weka para processar a tabela de exemplos através do algoritmo J48 resultando em árvore de decisão. Então foram estudadas as ramificações e nós da árvore de decisão para a construção do algoritmo classificador no sistema.

A classificação de novos exemplos começa com inserção de uma notícia pelo site de notícias, onde o sistema processa o texto efetuando um pré-processamento semelhante ao pré-processamento para construção do classificador. A diferença é que não é feito o cálculo de frequência TF-iDF nem a tabela de exemplos. Contudo a remoção de stopwords e a redução das palavras ao seu singular são efetuadas.

Após o pré-processamento os termos são processados no classificador do sistema o qual define a qual categoria o texto pertence. Além da categoria o sistema apresenta uma lista das palavras mais importantes do texto.

3 EXECUÇÃO

A execução do Trabalho de Conclusão é o desenvolvimento do protótipo de site de notícias com ferramentas que auxiliam na publicação de textos. Ferramentas como o auxilio na categorização e a relação de palavras chaves dos textos. O processo de desenvolvimento do sistema está representado na Figura 14.

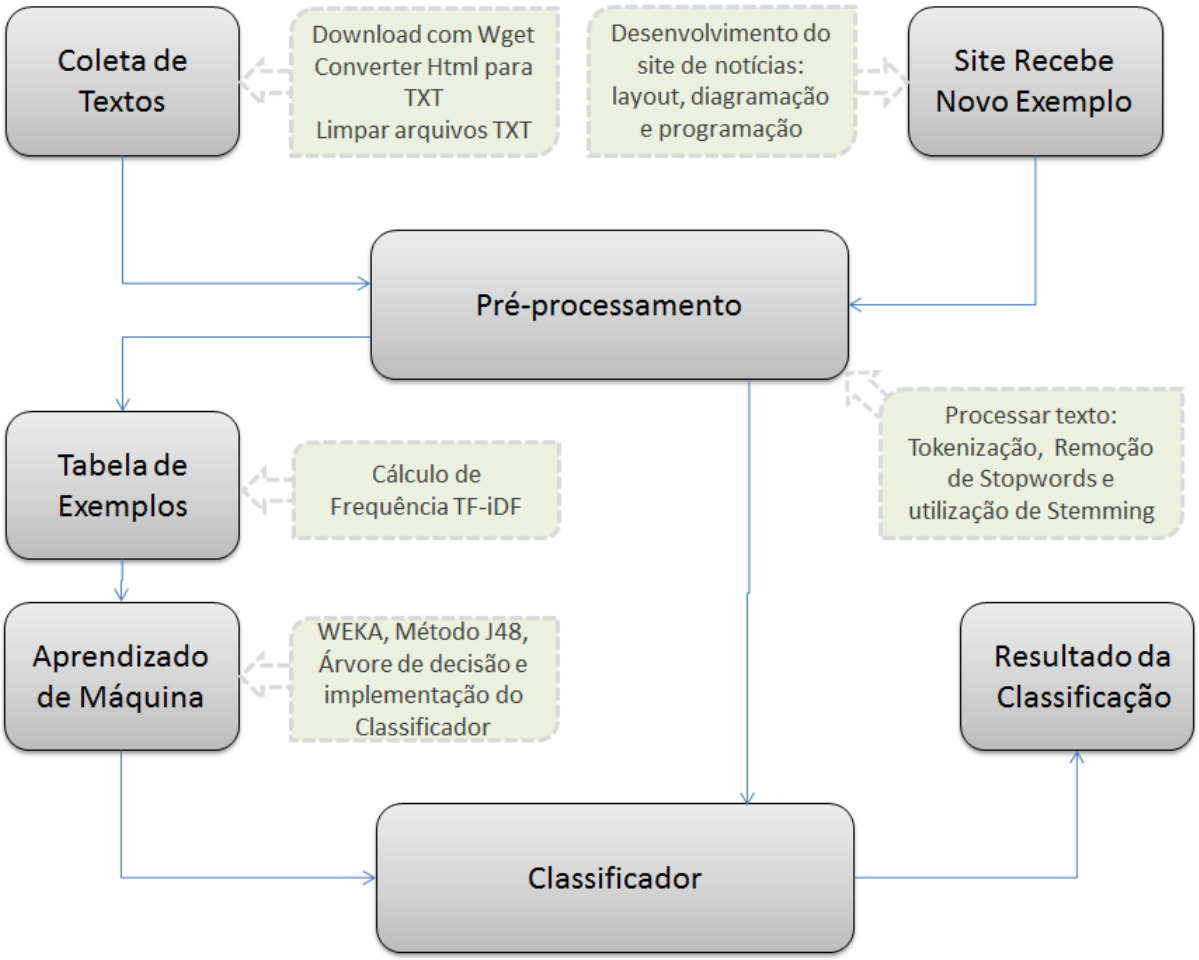

Figura 14 – Esquema completo da criação do Protótipo.

A Figura 14 apresenta o esquema de procedimentos necessários para fazer o protótipo do Trabalho de Conclusão. As atividades se dividem em dois grandes processos, sendo o primeiro a construção do classificador e se inicia com a coleta de exemplos, passa pelo pré-processamento, constrói a tabela de exemplos, passa pelo aprendizado de máquina concluindo na implementação do classificador. E o segundo é a construção de um site de notícias que recebe novos exemplos, efetua o pré-processamento, classifica o texto e demonstra o resultado da classificação.

A descrição do processo de desenvolvimento do protótipo será dividida em dois capítulos sendo o primeiro o Construção do Classificador e o segundo Classificação de Novos Exemplos, conforme segue.

3.1 Construção do Classificador

Para a construção do classificador se faz necessário uma série de etapas que serão descritas neste capítulo. Para visualizar o processo pode-se observar na Figura 10 os processos de Coleta de textos, Pré-processamento, Tabela de exemplos, Aprendizado de Máquina e Classificador.

A coleta de textos partiu da seleção de sites de notícias como, por exemplo, O Globo, JB, A Tarde, O Povo, Jornal NH, entre outros, então se buscou uma ferramenta para coletar as notícias destes sites de forma automática. A ferramenta utilizada para isso foi o Wget conforme citado do capítulo 1.3.1. Então se procurou testar qual dos sites de notícia selecionado possibilita a utilização do Wget para baixar os arquivos. Em alguns destes sites, os arquivos não estavam devidamente organizados não sendo possível saber a qual categoria pertencia como, por exemplo, o site da Zero Hora, e outros sobrescreviam muitas notícias num mesmo arquivo como, por exemplo, o site do Estadão, tendo por resultado um número muito pequeno de exemplos. Por fim foram escolhidos dois sites para coletar os textos: Terra e Uol.

Como um site grande de notícias possui um número muito extenso de publicações decidiu-se escolher seis categorias e coletar apenas textos que pertençam às mesmas. As categorias foram as seguintes: Esportes, Política, Economia, Ciências e Tecnologia e Educação.

Para efetuar o download dos arquivos especificamente dentro da categoria e do site desejado foi utilizado o comando de Wget conforme a Figura 15.

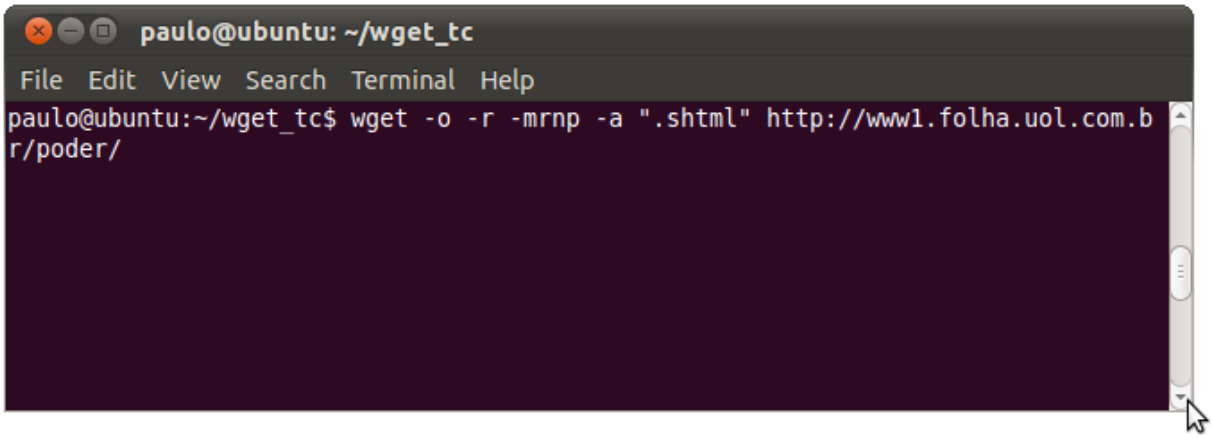

Figura 15 - Tela do Wget sendo executando no Linux Ubuntu.

A Figura 15 apresenta a linha de comando executada no Wget para efetuar o download de arquivos do site especificado com os seguintes parâmetros: “-o” cria um log de execução, “-r” efetua a leitura do diretório do site de forma recursiva, “-mrnp” limita o acesso aos arquivos pertinentes ao domínio e diretório específico, ‘-a “.shtml” ’ limita o download de arquivos apenas em formato “.shtml”. O diretório que está sendo utilizado para salvar os arquivos é o “wget_tc” e o site e seção que está sendo baixada são o “www.folha.com.br” seção “poder”, que representa política no site da folha.

O resultado do download dos arquivos dos sites deseja está representado na Tabela 4.

Fonte	Conteúdo	Quantidade
Site 1	Ciência	7391
	Educação	9438
	Esportes	8453
	Tecnologia	10249
Site 2	Ciência	1094
	Economia	6134
	Esportes	8119
	Política	7876
	Tecnologia	1811
TOTAL		60565

Tabela 4 - Tabela de quantidades de documentos coletados por site e categoria.

A Tabela 4 mostra os totais de textos que foram baixados através da ferramenta Wget. Após baixar os arquivos foram organizados os arquivos dividindo-os em pastas com o nome do site e o nome da categoria. Depois foi utilizada uma ferramenta para converter os arquivos de HTML para TXT. Essa ferramenta foi o HTML2TEXT conforme apresentado no capítulo 1.3.1. A Figura 16 o modo que foi usado para converter os arquivos de HTML para Txt.

Figura 16– Tela do Html2text sendo executando no Linux Ubuntu.

Na Figura 16 consta um exemplo do código utilizado para converter arquivos do formato HTML para TXT. O objetivo dessa linha de comando foi converter todos os arquivos cuja a estenção fosse ".shtml" do diretório "poder_txt" para Txt, para que isso fosse possível foi usado os determinados comandos: *"ls –l *.shtml"* para listar todos os arquivos ".shtml", "|" para concatenar novo comando, "*awk*" para executar uma determinada string, *'{system("html2text –o " substr($8,1,index($8, ".")-1) ".txt " $8)}'* onde o system executa o que está entre parênteses, *$8* representa o nome do arquivo que está sendo listado no momento, o trecho *substr($8,1,index($8, ".")-1)* ".txt " pega o nome do arquivo e substitui o .shtml pelo txt, e o *html2text* converte o arquivo que está sendo listando no momento do formato ".shtml" para o formato ".txt".

Depois de converter o arquivo para Txt ainda são utilizados alguns algoritmos desenvolvidos em php para quebrar o texto utilizando alguns atributos de corte. Então o algoritmo seleciona qual parte é mais extensa do texto e exclui o resto. Através dessa lógica é possível abstrair apenas a notícias, excluindo informações extras do texto como, por exemplo: publicidade, menu e textos relacionados ao site ou a notícia.

A Figura 17 demonstra o texto em formato HTML, em formato TXT e o TXT após o processamento dos algoritmos de limpeza.

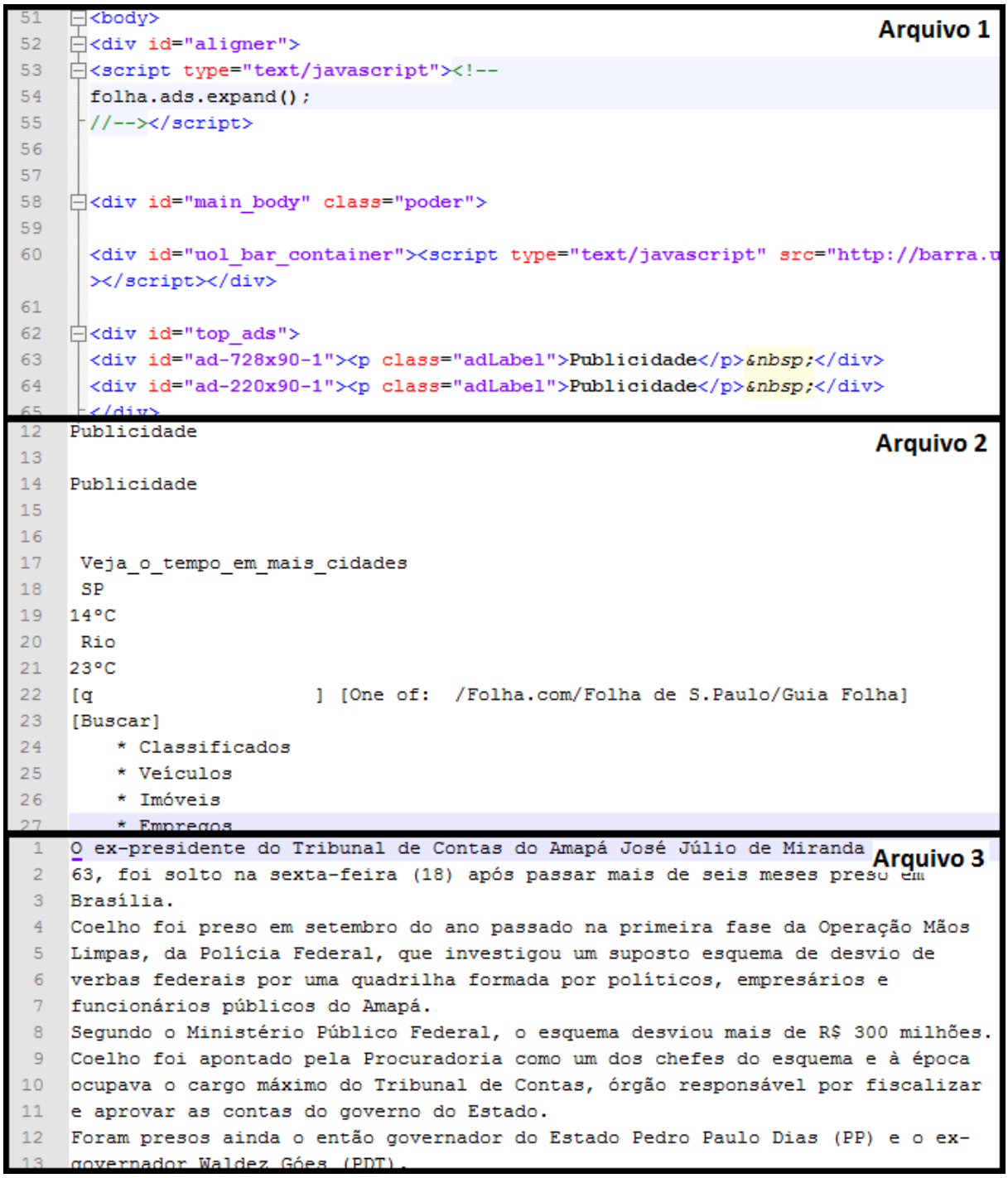

Figura 17– Demonstração dos arquivos Html, Txt e Txt limpo.

A Figura 17 demonstra três fases do processo de preparação dos textos depois do download dos arquivos do site utilizando o Wget. Conforme a Figura 17 o “Arquivo 1”

representa o arquivo em formato HTML onde é possível ver uma todas as tags do código fonte do arquivo, o “Arquivo 2” onde aparece o texto em formato TXT após ser convertido do HTML, e o “Arquivo 3” no qual aparece o texto após a limpeza dos textos que não faziam partes da notícia.

Essas etapas foram feitas para todos os arquivos baixados, formando assim um corpus de textos. Concluindo a construção do corpus o processo seguiu com o Pré-processamento.

No Pré-processamento foram realizadas as seguintes etapas: Tokenização, remoção de Stopwords, redução das palavras do plural para o singular através do Stemming. Para realizar o desenvolvimento destas etapas foi utilizada a Linguagem de Programação PHP 5.2.9-2 (PHP: Hypertext Preprocessor) compilada através do Servidor HTTP Apache 2.2.11. A seguir seguiremos falando de pré-processamento a iniciar por a utilização de Stemming.

A etapa de redução de termos do plural para o singular conforme capítulo 1.4.2 é bem complexa na língua portuguesa. Complexa porque não possui um padrão bem definido. Sabe-se que o plural do substantivo quando termina em vogal é adicionado um “s” no final da palavra, contudo existem exceções como, por exemplo, a palavra “Deus”. Ela está no singular, contudo termina em “s” e seu plural seria “Deuses”. Essa entre outras exceções dificultam bastante a criação de um algoritmo capaz de reduzir palavras do plural para o singular de maneira automática. Uma maneira seria selecionar todas as palavras que são exceções e não permitir que as reduzam ao singular da maneira simplificada. A função demonstrada na Figura 18 é uma função simplificada de redução de termos em português para o singular.

```php
<?php
    function radical_plural ($palavra)
    {
        $simples = substr($palavra,-2);
        if ($simples == 'as' || $simples == 'es' || $simples == 'is' ||
            $simples == 'os' || $simples == 'us')
        {
            $palavra = substr($palavra,0,-1);
        }
        $composto = substr($palavra,-3);
        if ($composto == 'ões') { $palavra = substr($palavra,0,-3).'ão'; }
        if ($composto == 'ãos') { $palavra = substr($palavra,0,-3).'ão'; }
        if ($composto == 'ães') { $palavra = substr($palavra,0,-3).'ão'; }
        if ($composto == 'les') { $palavra = substr($palavra,0,-3).'l'; }
        return $palavra;
    }
?>
```

Figura 18 – Código fonte em PHP da função que reduz palavras ao seu singular.

A Figura 18 demonstra o código fonte da função “radical_plural” responsável por reduzir termos em português do plural para o singular de forma simplificada. Conforme se pode ver a Linha 2 da Figura, a função recebe uma palavra de entrada, então pega os últimos dois caracteres da palavra através da função “substr($palavra,-2)” e testa se esses dois últimos caracteres são: “as”, “es”, “is”, “os” ou “us”. Se sim, a função retorna a palavra sem o último caractere. Além disso, a função testa também os últimos três caracteres (Linha 10), caso sejam: “ões” substitui por “ão”, “ãos” substitui por “ão”, “ães” substitui por “ão” e caso sejam “les” substitui por “l”. Desta forma todos os termos executados pela função serão reduzidos ao seu singular de forma simplificada, ou seja, não pré-vendo as exceções linguísticas do português.

Uma maneira de trabalhar com processamento de palavras é através de vetores (Array). Desta forma, podem-se criar mecanismos para salvar as informações que se deseja processar em vários Array e depois combinar esses Array fazendo os filtros desejados. Através disso, para efetuar o Pré-processamento dos textos, pode-se salvar a lista de termos delimitadores da Tokenização em um Array, a lista de Stopwords em outro Array e os termos do texto a sem processado em um terceiro Array, e por fim processar todos esses Array fazendo os filtros necessários como, por exemplo, o código demonstrado na figura 19.

```php
<?php
$texto = 'O sopro de vento empurrou lentamente as velhas portas entre-abertas.';
$arr_tokenizacao = array(' ','\n','\t','\r');
$texto = str_replace($arr_tokenizacao,' ',$texto);
$palavras_do_texto = explode(' ',$texto);
$arr_stopwords = array ('e','é','à','às','a','ao','o','aos','os','as','de','na','nas','no');
print 'Texto antes: '.$texto ;
$arr_caracteres_a_serem_removidos = array ('\"',',','.',';',':','[',']','{','}','ª','<','>');
print '<BR> Texto depois: ';
foreach ($palavras_do_texto as $valor_novo)
{
    $valor_novo = str_replace($arr_caracteres_a_serem_removidos,'',$valor_novo);
    $valor_novo = radical_plural($valor_novo);
    $ativa_word = true;
    foreach ($arr_stopwords as $value_stop)
    {
        if (ucfirst($value_stop) == ucfirst($valor_novo))
        {
            $ativa_word = false;
        }
    }
    if ($ativa_word)
    {
        $texo_a_salvar .= ' '.trim($valor_novo);
    }

}
print $texo_a_salvar;
?>
```

Figura 19 – Código fonte em PHP das rotinas de Pré-processamento.

A Figura 19 demonstra o código fonte dos algoritmos de Pré-processamento desenvolvidos na Linguagem PHP. Para entender esse código fonte é importante entender primeiro as variáveis declaradas, conforme segue:

- $texto: representa o texto de entrada;
- $arr_tokenizacao: representa um Array de termos delimitadores;
- $palavras_do_texto: representa um Array das palavras do texto;
- $arr_stopwords: representa um Array de Stopwords;
- $arr_caracteres_a_serem_removidos: representa um Array de caracteres a serem removidos.

O primeiro Array a ser combinado no algoritmo é na atribuição de valor da variável *$texto* (Linha 4) onde a lista de tokens é substituída por espaço com a utilização do comando de "*str_replace*". O comando "*foreach*" (Linha 10) inicia o primeiro laço do algoritmo onde todos os termos contidos no Array "*$palavras_do_texto*" são processados um a um. Dentro do primeiro laço o primeiro filtro a ser executado é a exclusão dos caracteres através do str_replace (Linha 12) que substitui por vazio a lista de valores do Array "*$arr_caracteres_a_serem_removidos*". A redução dos termos ao seu singular é feita através

da execução da função “radical_plural” (Linha 13, o algoritmo da função pode ser visto na Figura 14). A remoção das Stopwords se dá da Linha 14 a Linha 25 iniciando com a declaração de variável ”*$ativa_word*” como verdadeira, logo após o sistema faz um laço utilizando o “*foreach*” que lista todas as Stopwords. Então o “*if*” (Linha 17) verifica se o termo do texto pertence a algum termo da lista de Stopwords, se sim a variável ”*$ativa_word*” é marcada como falso e não entra no “*if*” da Linha 22 que salva as palavras do texto.

Em resumo o algoritmo processa a lista de tokens (Linha 4), remove caracteres indesejados (Linha 12), reduz as palavras ao seu radical (Linha 13) e remove as Stopwords (Linha 14 a Linha 25).

No algoritmo existem três momentos em que é solicitado que escreva a resposta através do comando “*print*”, são eles nas linhas: 7, 9 e 28, a resposta desta impressão pode ser vista na Figura 20.

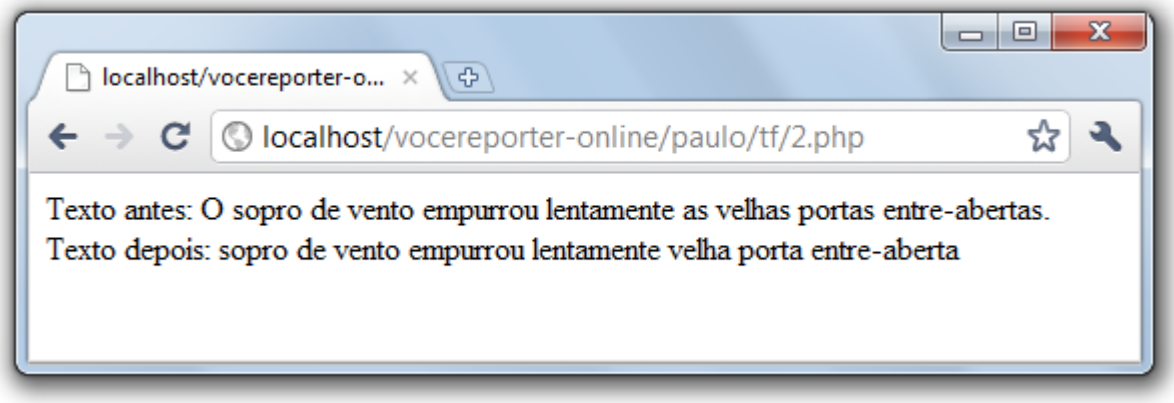

Figura 20 – Tela do Browser com resultando do Pré-processamento.

A Figura 20 mostra o resultado que o algoritmo demonstrado na Figura 19 solicita que seja impresso na tela. Na tela pode-se observar o texto antes de ser processado e o texto depois de ser processando. No texto depois de processado podem-se observar as seguintes alterações: não consta o “.” no final da frase o qual foi excluído na faze de remoção de caracteres, não consta também os termos: “O” e “as” removidos no processamento da lista de Stopwords e as palavras: “velhas”, “portas”, “entre-abertas” estão no singular conforme a função que de plural apresentada na Figura 18.

Depois de concluída a fase de Pré-processamento dos textos progride-se para a fase de construção da tabela de exemplos. O método utilizado para construção da tabela de exemplos foi através do cálculo de frequência TF-iDF o qual é responsável por definir o grau

de importância que um termo tem para com um conjunto de textos, conforme apresentado no capítulo 1.4.3.

Para processar um número de arquivos tão grande quanto é o caso do Pré-processamento e da construção da tabela de exemplos, os arquivos foram salvos em diretórios com o mesmo nome da categoria. Ao invés de processar texto a texto, os algoritmos foram desenvolvidos para receber uma lista de diretórios, então o sistema lê todos os arquivos de todos os diretórios executando as funções desejadas.

Como visto no capítulo que fala sobre o TF-iDF, para fazer o cálculo é necessário que o sistema processe todos os textos diversas vezes, isso porque necessita das informações do documento em comparação a todos os outros documentos. A partir desta necessidade o primeiro grande desafiou foi fazer o algoritmo processar os 60.565 textos baixados na fase de coleta dos dados. Isso porque teria que salvar um grande número de informação na memória do computador durante a execução do script o que é bastante complicado.

Durante os primeiros testes foi possível processar a quantidade máxima de 30 textos divididos em três categorias. Isso porque o sistema tinha de gerar a frequência total de cada termo no texto a cada processamento. Uma maneira de automatizar esse processo foi salvar nos diretórios apenas a relação de termo frequência de cada arquivo em relação ao texto. Desta forma foi possível chegar a processar 2100 textos divididos em três categorias. Então foram estudadas outras estratégias para compilar esses textos a fim de conseguir um resultado melhor, conduto eram soluções que demandavam tempo e para não comprometer a entrega do trabalho foram deixadas para o futuro. Tais soluções, assim como melhorias para o projeto serão descritas na conclusão.

Para calcular o TF-iDF de um atributo em relação a um conjunto de textos é necessário possuir quatro informações: número de vezes que o termo aparece dentro do documento (TF), número total de termos de todos os documentos (TFT), número total de documentos (DFT) e o número de documentos que contém o termo (DF). O algoritmo simplificado para demonstrar o cálculo de frequência é bastante grande, por isso foi separa em duas partes, sendo a primeira a Figura 21.

```php
<?
function Return_TF ($TERMO, $DOCS)
{
    $resultado = 0;
    $T = explode(' ',$DOCS);
    foreach ($T as $TERMS)
    {
        if (strtolower(trim($TERMS)) == strtolower(trim($TERMO)))
        {
            $resultado = $resultado + 1;
        }
    }
    return $resultado;
}
function Return_TFT ($D)
{
    $resultado = 0;
    foreach ($D as $DOCS)
    {
       $T = explode(' ',$DOCS);
       $resultado = count($T) + $resultado;
    }
    return $resultado;
}
function Return_DF ($TERMO, $D)
{
    $resultado = 0;
    foreach ($D as $DOCS)
    {
        $T = explode(' ', $DOCS);
        foreach ($T as $TERMOS)
        {
            $resultado = ($TERMOS == $TERMO) ? $resultado+1 : $resultado ;
        }
    }
    return $resultado;
}
?>
```

Figura 21 – Código fonte em PHP de funções para calcular o TF-iDF.

A Figura 21 demonstra o código fonte de três funções necessárias para calcular o TF-iDF de cada termo por documento. A primeira função “ReturnTF” tem por entrada dois valores, primeiro o termo, e segundo o documento, esta função retorna a frequência com que o termo de entrada aparece no documento. Para isso o *explode* (Linha 5) transforma cada termo do texto em um Array separado por “ ” (espaço), então é feito um loop (Linha 6) e que compara (Linha 8) termo a termo se é igual ao termo de entrada, se sim soma um no contador (Linha 10) e no final a função retorna valor total do contador (Linha 13).

A função “ReturnTFT” tem como entrada todos os documentos salvos no array *$D* e retorna o número total de termos de todos os documentos. Para isso é feito um Loop para listar todos os documentos (Linha 18), dentro de cada documento é utilizado o *explode* (Linha 20) para quebrar os termos em um Array de palavras separados por “ ” (espaço), então o *count* (Linha 21) para contar o número de termos do documento salvando no totalizador *$resultado*. Então o função retorna o total de termos de todos os documentos (Linha 23).

A função “Return_DF” tem como entrada o termo e a lista de documentos e tem como resultado o número de documentos que contém o termo. Para isso sistema faz um Loop (Linha 28) dos documentos e converte cada termo do texto em um item do Array *$T* (Linha 30) o qual é convertido em um novo Loop (Linha 31) e dentro deste é feita a verificação se é igual ou não ao temo de entrada (Linha 33), se sim é somando um no $resultado, retornado no final o total de documentos que conte o termo de entrada (Linha 36).

Essas funções são chamadas para calcular o TF-iDF de cada termo por documento conforme a Figura 22.

```
1  <?php
2  print 'TABELA';
3  $DFT = count ($D);
4  foreach ($D as $DOCS)
5  {
6      $T = explode(' ',$DOCS);
7      foreach($T as $TERMS)
8      {
9          if (!isset($TERMOS[$TERMS]))
10         {
11             $TERMOS[$TERMS] = 1;
12             print ', '.$TERMS;
13         }
14     }
15 }
16 $CONTA_DOCS = 1;
17 foreach ($D as $DOCUMENTO)
18 {
19     foreach ($TERMOS as $TERMO => $valor)
20     {
21         print ( Return_TF($TERMO, $DOCUMENTO) / Return_TFT($D)) * ($DFT / Return_DF($TERMO, $D));
22     }
23 }
24 ?>
```

Figura 22 – Código fonte em PHP da função que reduz palavras ao seu singular.

A Figura 22 demonstra o código fonte do algoritmo de cálculo de frequência TF-iDF. A primeira necessidade é conseguir uma lista de todas as palavras que contem em todos os documentos, para isso os documentos foram salvos em forma de Array na variável *$D*. Para selecionar esses temos o *foreach* (Linha 4) que lista todos documentos um a um dentro do

loop. Dentro dos documentos é necessário selecionar todas as palavras, e para isso é utilizado o *explode* que transforma todas as informações do texto, separadas por “ ” (espaço) contida na variável *$DOCS* e as salva em no Array *$T*. Então é feito o segundo (Linha 6) onde o algoritmo verifica se o termo já foi salvo na variável *$TERMOS* (Linha 9), caso não ele salva (Linha 11).

Então o sistema necessita fazer um novo laço para listar todas as palavras e chamar as funções que calculam a frequência dos termos. Desta forma o Loop de documentos lista todas as informações contidas no Array $D (Linha 17) e dentro deste tem um segundo Loop (Linha 19) que lista todas as palavras salvas anteriormente no Array *$TERMOS* (Linha 11) então o sistema imprime na tela o resultado do cálculo do TF-iDF (Linha 21). Na fórmula o total de vezes que um termo aparece em um documento (“Return_TF”) dividido pelo número total de termos de todos os documentos (“Return_TFT”) é multiplicado pela divisão entre o total de documentos (“$DFT”) e o número total de documentos que contém o termo (“Return_DF”).

A fim de limitar o número de termos da tabela de exemplos pode-se utilizar um ponto de corte. O ponto de corte é um limite de termos, por exemplo: 5000, com esse ponto de corte, uma tabela de exemplos pode ter até no máximo 5000 termos. Para definir o ponto ideal de corte pode-se usar de dois parâmetros: ordenar por maior frequência do termo, ou ordenar maior resultado no cálculo TFiDF. No desenvolvimento do protótipo foi utilizada a ordenação pela frequência do termo e o limite de 2100 atributos.

É possível montar uma tabela com esses exemplos para visualizar os resultados conforme a Figura 23 que é resultante pelo cálculo do TF-iDF dos documentos do sistema.

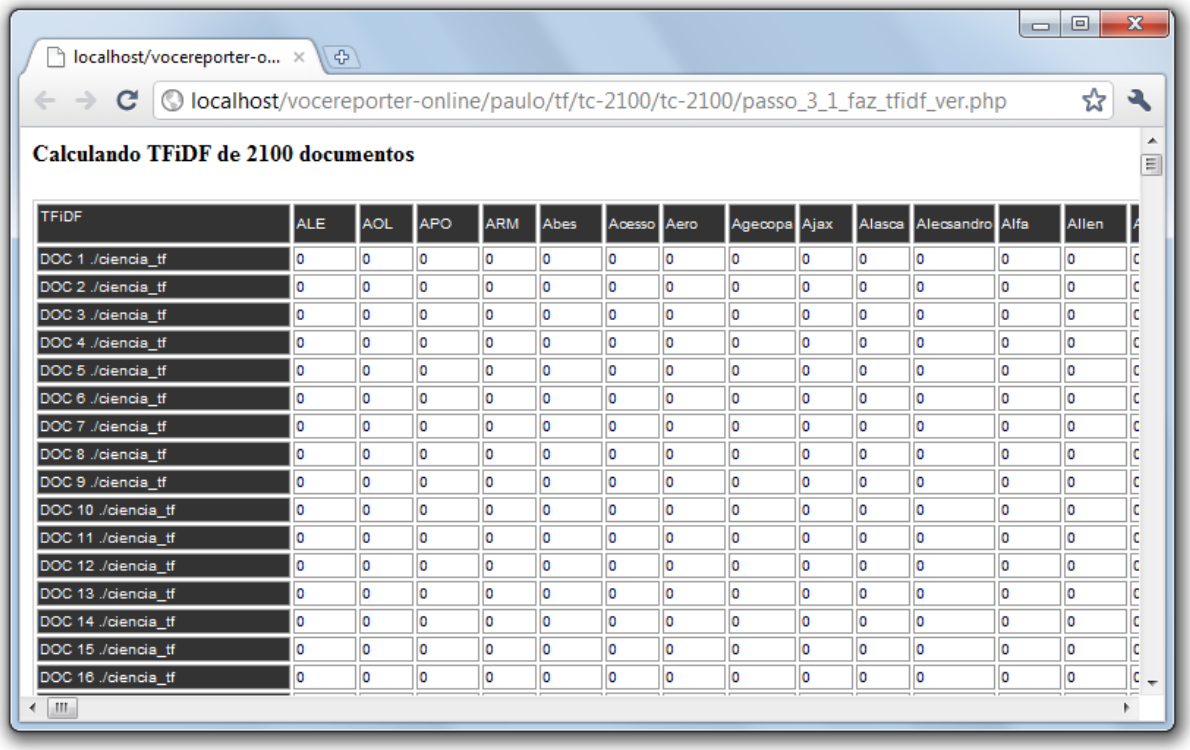

Figura 23 – Tela do Browser com resultando da Tabela de Exemplos.

Na Figura 23 está demonstrada a tabela de exemplos final com o cálculo de frequência nos 2100 exemplos divido em três categorias, sendo elas: Esportes, Ciências e Tecnologia. Após concluída a construção da tabela de exemplo ela foi adaptada para ser processada na ferramenta WEKA através do formato do arquivo ARFF demonstrado na Figura 5 e explicado no capítulo 1.5.

Depois de formatado o arquivo ARFF para ser classificado foi utilizado três algoritmos classificadores para testar o desempenho e analisar o resultado a fim de descobrir quais teriam melhores resultados. A forma de enviar o arquivo e de escolher o classificador está explicada nos capítulos: 1.6.1 Método J48, 1.6.2 Naive Bayes, 1.6.3 Support Vector Machine, onde em cada capítulo explica o método correspondente.

Outra opção que foi utilizada para testar a aprendizagem foi método de treinamento. Os métodos utilizados formam o Cross-validation e o Percentage Split. Onde no Cross-validation o classificador é avaliado por validação cruzada, ou seja, o conjunto de teste é dividido em partes iguais e a predição é aplicada a cada um separadamente. E o Percentage Split faz a predição baseada na porcentagem dos dados que o usuário determina na própria ferramenta. A quantidade de dados capturados para teste pela ferramenta depende do valor

atribuído ao campo porcentagem. A quantidade escolhida para testar o método Percentage Split foi 66%. Para demonstrar os resultados foram feitos gráficos conforme a Figura 24.

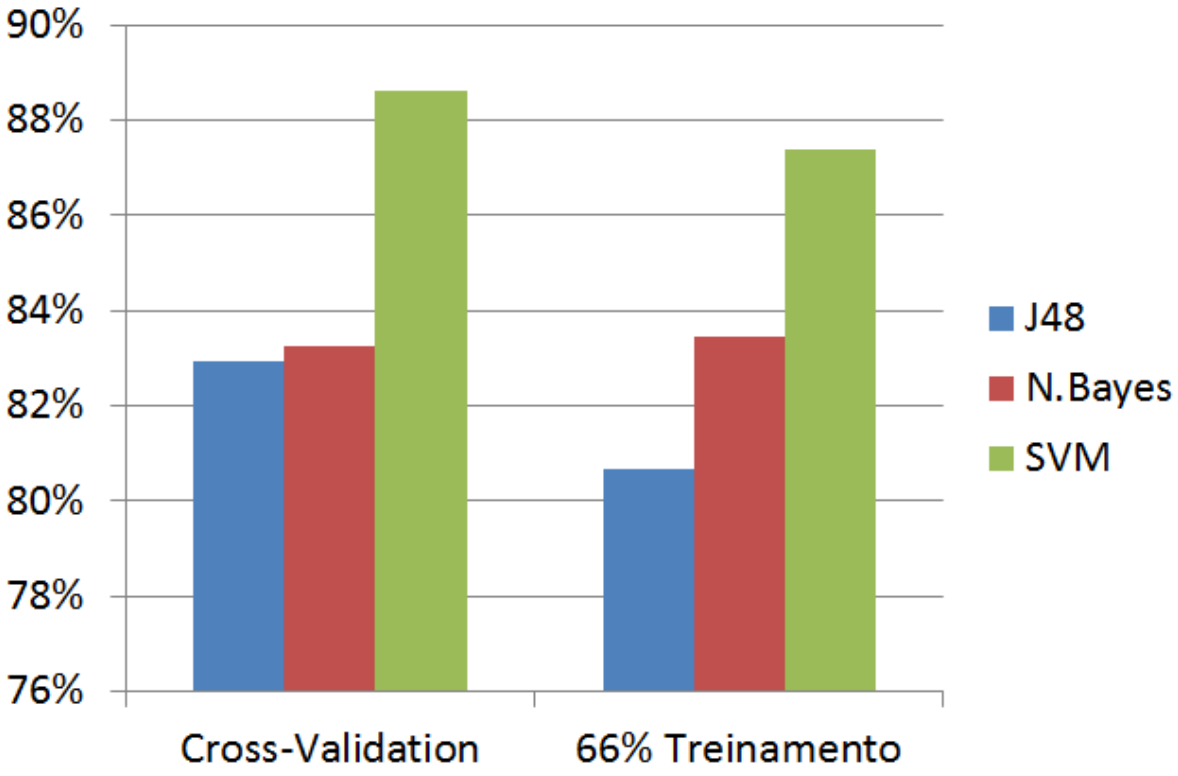

Figura 24 – Gráfico dos resultados dos métodos de AM divididos por tipo de treinamento.

Na Figura 24 estão apresentados os gráficos de desempenho dos algoritmos de Aprendizado de Máquina: J48, Naive Bayes e Support Vector Machine combinados aos métodos de Treinamento Cross-Validation e Percentage Split (porcentagem de treinamento). Através do resultado é possível notar que o método SVM teve desempenho próximo a 90% em ambos os métodos de treinamento, sendo que no método Percentage Split a vantagem foi de apenas 4% (83% a 87%). Contudo em uma análise geral os resultados foram bem próximos variando de 80,7% a 88,6%.

Na análise por categoria é possível ver o quanto os três algoritmos tem resultados parelhos conforme a Figura 25.

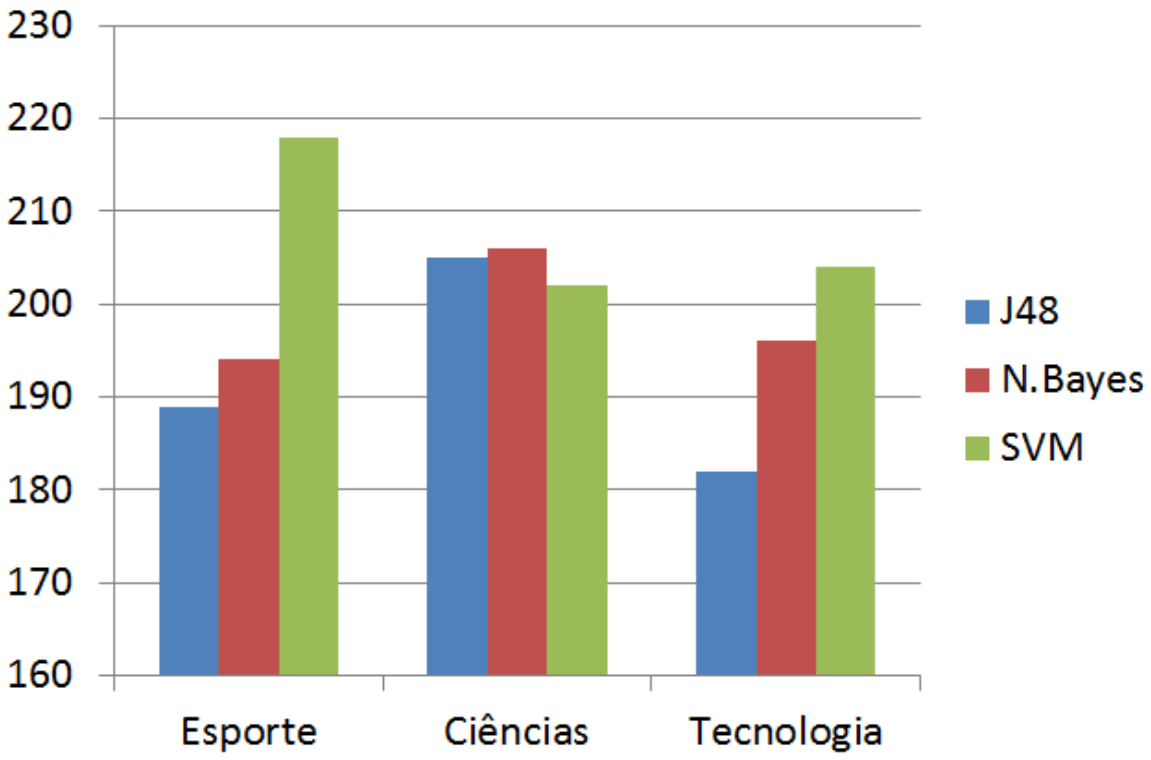

Figura 25 – Gráfico dos resultados dos Métodos de AM divido por categoria.

Na Figura 25 estão apresentados os gráficos de desempenho dos algoritmos de Aprendizado de Máquina por categoria utilizando o treinamento Percentage Split. Neste gráfico é possível ver que a grande diferença que o SVM adquiriu perante os outros métodos foi através da categoria de Esportes. Na categoria Ciências o método Naive Bayes chego a ter melhor resultado. O Método J48 aparentemente tem resultados bem inferiores, mas ainda sim tem um acerto nas classificações muito bom, sendo sempre superior a 80%. Na Figura 26 o mesmo teste é feito testando o método de treinamento Cross-validation.

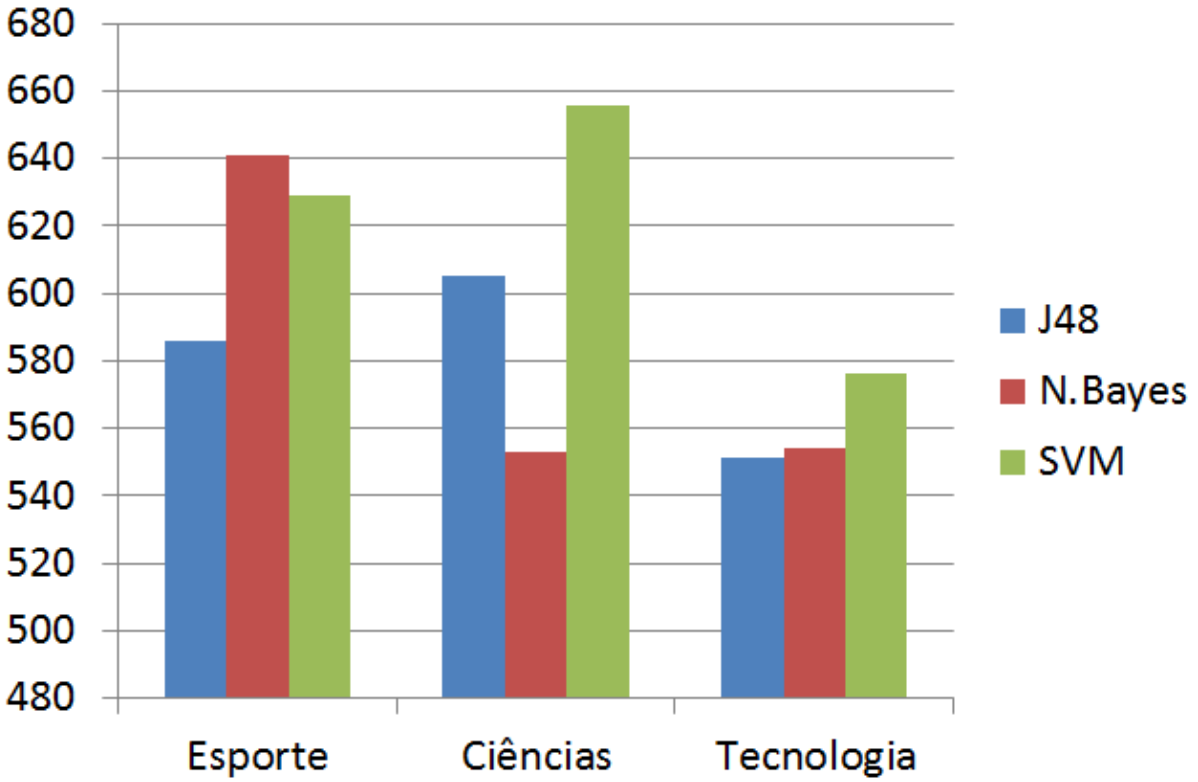

Figura 26 – Gráfico dos resultados dos Métodos de AM divido por categoria.

Na Figura 26 estão apresentados os gráficos de desempenho dos algoritmos de Aprendizado de Máquina por categoria utilizando o treinamento Cross-validation. Neste gráfico é possível ver resultados bem parelhos, sendo que o SVM se mantem com um aproveitamento melhor, contudo o índice mais alto deixou de ser na categoria de esportes e passou a ser na de Ciências. Contudo os métodos tiveram um bom equilíbrio de acertos por categoria. Acertos que podem ser facilmente influenciados pelo número de exemplos.

O Método J48 ficou um pouco abaixo dos demais classificadores, contudo seu resultado é bastante interessante, ainda mais porque ele gera uma árvore de decisão. Os algoritmos Naive Bayes e Support Vector Machine são caixas pretas, ou seja, geram um motor de classificação só podem ser lidos por sistemas classificadores adaptados e não podem ser vistos pelo usuário. A árvore de decisão, conforme visto no capítulo 1.6.1, pode ser interpretada visualmente pelo usuário.

A árvore de decisão resultante da aplicação no Método J48 possui 64 níveis e 127 nós de decisão. Parte dela pode ser visto na Figura 27.

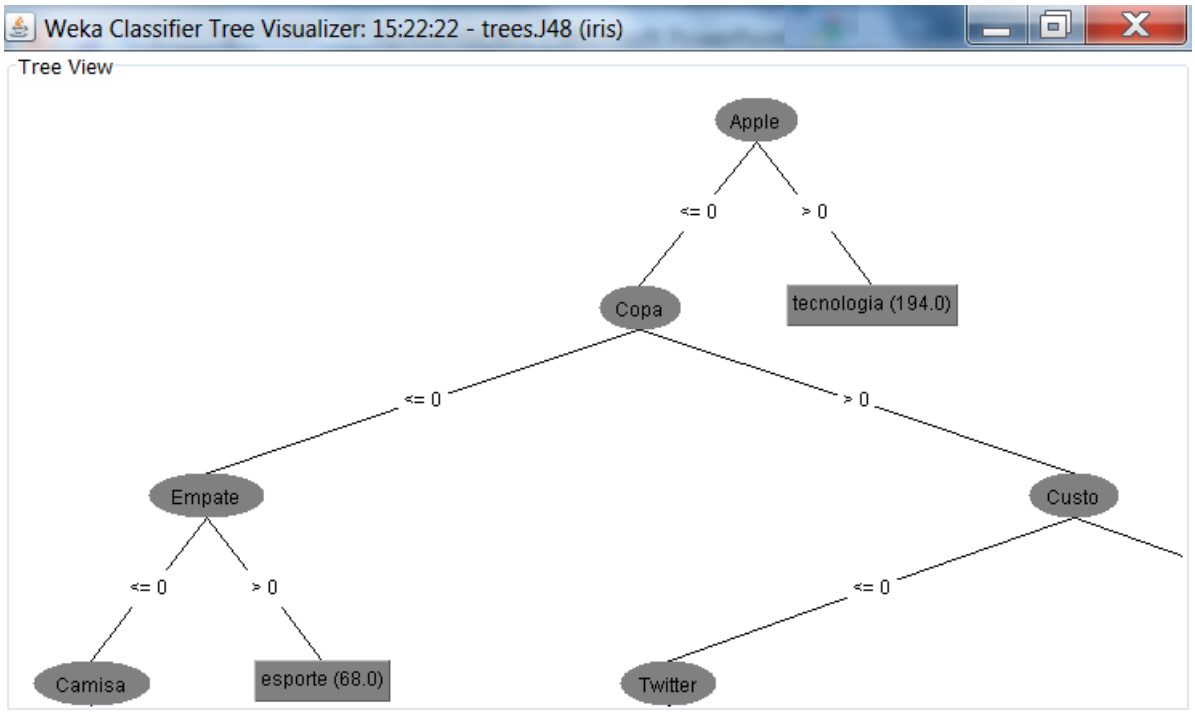

Figura 27 – Parte da árvore de decisão resultante do método J48.

Na Figura 27 está apresentado o resultado do processamento da tabela de exemplos demonstrada na Figura 17 no método de Aprendizado de Máquina J48. A árvore de decisão tem por finalidade ser implementada pela linguagem de programação a fim de construir um classificador. Como a árvore é muito grande é difícil descrever todo o código fonte necessário para gerar o classificador, contudo a Figura 28 traz um trecho do código fonte descrita na linguagem Php.

```php
<?php
function arvore_de_decisao ($valor)
{
    if ($valor['Apple'] > 0)
    {
        return 'tecnologia';
    }
    elseif ($valor['Copa'] > 0)
    {
        if ($valor['custo'] > 0)
        {
            return 'tecnologia';
        }
        else
        {
            return 'esporte';
        }
    }
    elseif ($valor['Empate'] > 0)
    {
        return 'esporte';
    }
    else if ($valor['Camisa'] > 0)
    {
        return 'ciencia';
    }
}
?>
```

Figura 28 – Código em Php da implementação da Árvore de Decisão.

Na Figura 28 está apresentado código fonte descrito em Php que implementa parte da árvore de decisão apresentada na Figura 20. A função "arvore_de_decisao" recebe como entrada um *$valor* (Linha 2) o qual representa um Array contendo todas as palavras enviadas por um novo exemplo. Então o primeiro "*if*" (Linha 4) verifica se no Array de entrada consta o termo "Apple", se sim classifica o texto como sendo de tecnologia (Linha 6), se não valida se o Array contem o termo "Copa" (Linha 8), se sim valida se o Array contem o termo "custo" (Linha 10), se sim classifica o texto como sendo de tecnologia, se não classifica como esporte. Então o sistema verifica se contem o termo "Empate" (Linha 19), se sim o sistema classifica o texto como esporte, se não o sistema testa se o Array contém o termo "Camisa", se sim retorna a categoria como sendo ciências. Desta forma o algoritmo tem a capacidade de testar diversos termos e, assim, classificar textos.

Os métodos e algoritmos neste capítulo descritos tem por objetivo a construção de um classificador de textos. O classificador é responsável por classificar novos os exemplos, a geração desses novos exemplos será descrita no próximo capítulo.

3.2 Classificação de Novos Exemplos

A Classificação de novos exemplos se faz a partir de um site de notícias. Um site de notícias pode ser administrado por uma equipe de pessoas que são responsáveis pelas publicações ou pode ser aberto ao público montando estrutura auto gerenciável. Acompanhando o novo conceito de ferramentas auto gerenciáveis da internet como, por exemplo, as redes sociais. Um dos mecanismos criados para agregar valor a categorização dos conteúdos, foi o desenvolvimento de um classificador através dos métodos de aprendizado de máquina.

Para o desenvolvimento do site foram necessárias as seguintes etapas: layout, diagramação e programação. Para fazer o layout das telas foi utilizado da ferramenta Adobe Fireworks na qual é possível juntar formas, textos e imagens definindo a aparência do site. Nesta etapa é determinada a organização dos conteúdos, os métodos de navegação e o modo de processar cada página. Já a diagramação pode ser feita da Ferramenta Adobe Dreamweaver utilizando as linguagens Html, Javascript e Css, as quais são interpretadas pelos navegadores de internet. Através dessas tecnologias são transformados os detalhes do layout em página de internet.

A partir de então, é possível criar o banco de dados do sistema e as classes de programação que fazem funcionar cada operação. O banco de dados tem duas tabelas principais: cadastro e texto. Relacionadas à tabela de texto tem ainda as tabelas: categoria, status, tipo e complementos. A tabela usuários também é chave segundaria da tabela texto. Após a modelagem e construção do banco de dados, é feita a programação do sistema que administra os usuários e os textos publicados.

Através da programação é possível gerenciar as informações do sistema, salvado e buscando os dados do banco dados. A página inicial do site de notícias desenvolvido pode ser visto através da Figura 29.

Figura 29 – Página inicial do site Jornalismo Interativo.

A Figura 29 apresenta a página inicial o site de notícias desenvolvido como protótipo do trabalho de conclusão. No topo da página a esquerda tem o logotipo e abaixo as categorias ativas do site, na esquerda tem o sistema de login e o local de cadastro para novos usuários. Abaixo tem a lista das notícias separados em dois blocos sendo que o da esquerda é a lista das notícias mais acessadas e o da direita é a lista das últimas notícias publicadas. Cada notícia pode ser visualizada, assim como sua galeria de fotos, pode também ser comentada e divulgada através das ferramentas de integração com redes sociais como: Twitter, Facebook e Orkut.

A dinâmica do site parte do cadastro de usuário onde a pessoa preenche seus dados como endereço, cidade, estado, sexo, data de nascimento e senha de acesso. Ao concluir o cadastro pode-se utilizar da mesma senha para efetuar no login no site. Fazendo o login ou o cadastro o sistema possibilita que cada usuário efetue o post de novos textos, edite e exclua posts anteriores. A página para publicar textos pode ser visualizada na Figura 30.

.

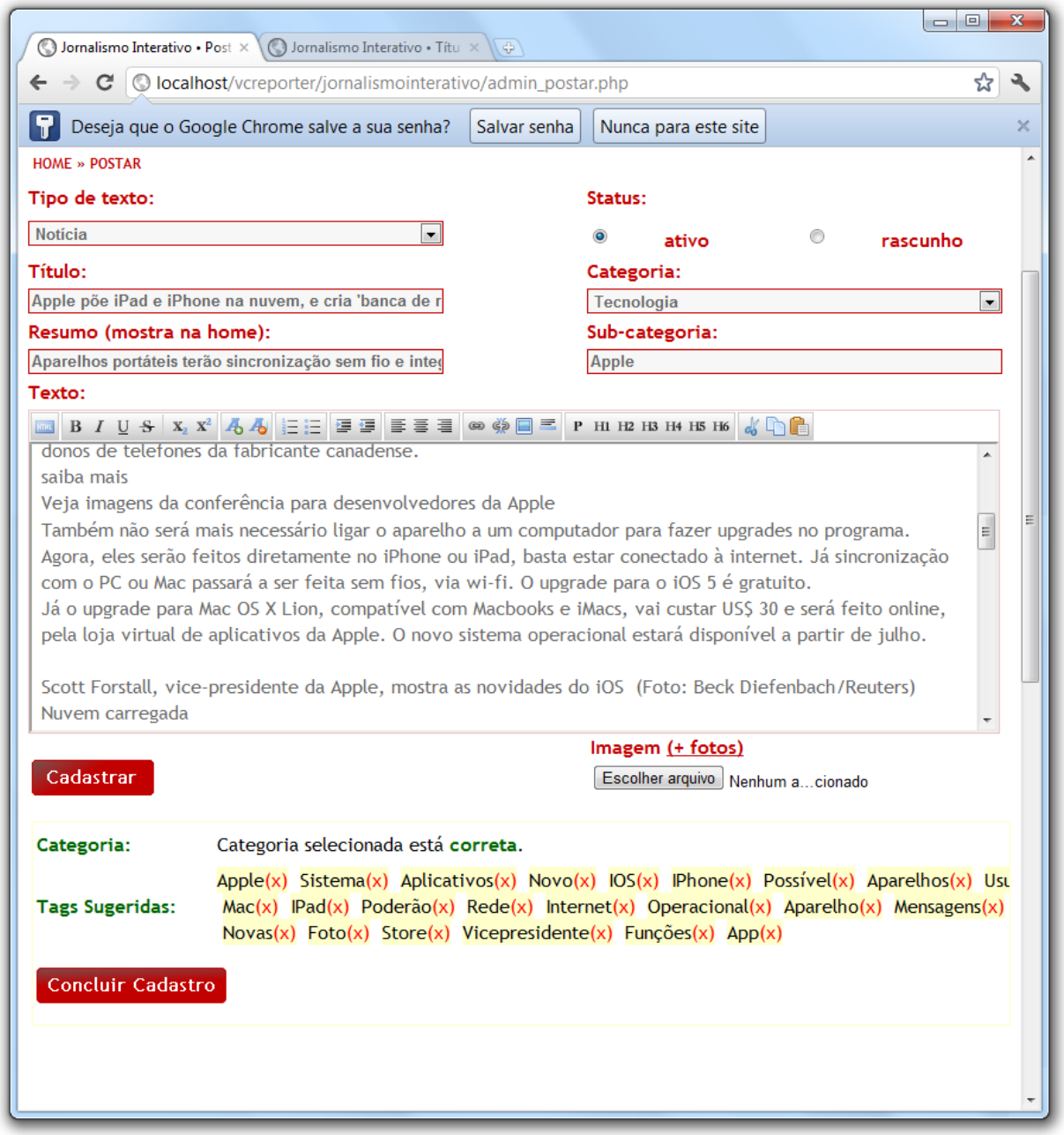

Figura 30 – Página de publicação de textos do site Jornalismo Interativo.

Na Figura 30 está uma imagem do site desenvolvido como protótipo para o trabalho de conclusão, na página responsável por publicar textos. Para publicar devem ser preenchidos os seguintes campos: tipo de texto, status, título, categoria, resumo para a capa, subcategoria e texto. O tipo de texto possui opções: notícia, matéria, opinião ou editorial, artigo, crônica e nova chamada; já o status possui duas opções: ativo para aparecer no site e rascunho para não aparecer. Na categoria pode-se escolher entre tecnologia, ciências e esportes. Além disso, tem os campos de texto: título, resumo para a capa, subcategoria e texto.

Ao preencher os campos do formulário de cadastro de notícias e clicar no bota “Cadastrar” o sistema processa o novo exemplo no classificador. Para isso, o texto passa pelo mesmo algoritmo de pré-processamento apresentando na Figura 15, onde são processadas as etapas: Tokenização, remoção de Stopwords e redução das palavras do plural para o singular através do Stemming. Depois de processado o texto é divido termo a termo dentro de um Array o qual serve de entrada na função “arvore_de_decisao” apresentada na Figura 21. Tal função retorna a classificação definida pelo classificador.

A resposta da classificação é apresentada abaixo do botão “Cadastrar” em dois resultados: categoria e tags sugeridas. A categoria tem como resposta correta quando demonstra a mensagem “Categoria selecionada está correta.” e quando incorreta apresenta a resposta como, por exemplo: “Categoria selecionada está errada. Segundo a análise a categoria correta seria tecnologia.”. As tags sugeridas são palavras chaves ordenadas por palavra mais importante do texto segundo o resultado de termo frequência sobre o pré-processamento.

CONCLUSÃO

A classificação automática de textos é um assunto muito trabalhado pelas comunidades de inteligência artificial. Sendo que, com o crescente desenvolvimento de softwares que utilizam de processamento digital de textos como: text mining, ferramentas tradutoras, ferramentas de auxilio a escrita, entre outras, os estudos nesta área são bastante amplos e possuem uma bibliografia qualificada e abrangente.

Trabalhar com classificação de textos jornalísticos é bastante importante à medida que é um o número de jornais e mídias online crescem no mundo todo. Isso por que é importante para qualificar ainda mais as ferramentas, automatizar o processo de maneira inteligente e auxiliar a tomada de decisão das pessoas. Propósito que auxilia ainda mais pessoas que não são da área jornalística a publicarem textos presando pela organização do site de notícias.

Com essa motivação foi desenvolvido um estudo na área de Processamento da Linguagem Natural e Aprendizado de Máquina a fim de construir um classificador automático de textos e adiciona-lo a ferramenta de publicação de textos. Além da classificação, um resultado do Pré-processamento de textos foi a análise de importância de palavras do documento o que teve por resultado a listagem automática de tags mais importantes.

Para fazer essas ferramentas foram realizados estudos para a coleta de textos e construção de um corpus de palavras separado por categorias, Pré-processamento das palavras utilizando técnicas como Stemming e remoção de Stopwords, cálculo de frequência TF-iDF para a construção da Tabela de Exemplos, ferramenta Weka para processar os exemplos utilizando as técnicas de aprendizado de máquina J48, Naive Bayes e Support Vector Machine e Árvore de Decisão para utilizar o resultado da classificação para implementar em uma linguagem de programação construindo um classificador.

Após a revisão teórica foram colocadas todas as etapas em prática e desenvolvido um classificador para textos jornalísticos divididos em três categorias: Esportes, Ciências e Tecnologia. Então foram utilizadas as técnicas de coleta de textos e Pré-processamento para a construção de um corpus de palavras que serviu como base para a construção de uma Tabela de Exemplos com o cálculo de frequência. Foi utilizada a ferramenta Weka para processar os exemplos nos algoritmos de aprendizado de máquina e analisado os resultados a fim de escolher qual utilizar para construir o classificador. Como os resultados foram muito próximos e a árvore de decisão gerada pelo Método J48 é um resultado visual o qual pode ser

testado e implementado com mais segurança, este foi o método escolhido. Por fim, o classificador foi adicionado a uma ferramenta de publicação do site de notícias desenvolvido como protótipo.

O desenvolvimento do protótipo teve as mesmas etapas planejadas e descritas no primeiro trabalho de conclusão, contudo a expertise sobre os conteúdos foi amplamente evoluída. O cronograma desenvolvido no Ante-Projeto foi bastante equivocado, o desenvolvimento do protótipo foi até metade de maio não sendo possível fazer uma pesquisa de campo para avaliar o protótipo. No Ante-projeto também foi relatada a intenção de validação das informações segundo normas jornalísticas, função a qual também teve de ser deixada para projetos futuros.

Ponto positivo do projeto foram os ótimos resultados obtidos na aprendizagem de máquina. Segundo a ferramenta Weka utilizada para simular o índice de aprendizagem, Figuras 17, 18 e 19, a porcentagem de acerto ficou bastante alta, variando de 80,7% a 88,6%. Simulando a entrada de exemplos no protótipo, o classificador responde com uma média de acerto bastante semelhante aos resultados, contudo, existe uma variável que influencia bastante na classificação: número de palavras do texto. Quanto maior o número de palavras, melhor o classificador se comporta e melhores são os resultados. Isso porque, em textos muito curtos, quase todos os termos tem a mesma importância dificultando a validação do classificador.

Outro ponto positivo foi a publicação do site Jornalismo Interativo (http://www.jornalismointerativo.com.br/) no dia 11 de Junho de 2011 com a implementação do classificador e a seleção automática de tags do texto. Através do site será possível fazer uma análise mais efetiva de qual a porcentagem de acerto do classificador pelas características dos exemplos. E também da aceitação dos usuários para com a ferramenta de classificação.

Ponto negativo do projeto foi o tempo levado para coletar e preparar os dados. Foram realizados muitos testes para baixar sites, os downloads são demorados e os documentos em sua grande maioria eram bagunçados, com o texto muito fragmentado em meio a propagandas e informações do site. A preparação dos dados levou algum tempo até que foi possível selecionar dois portais onde eram melhores os exemplos baixados.

Outro ponto negativo foi à construção da tabela de exemplos com poucas categorias. Para construir a Tabela de Exemplos foi utilizado o cálculo de frequência TF-iDF que faz cálculos de cada termo do texto em relação a todos os termos de todos os textos. Desta forma, a maneira encontrada para efetuar o cálculo, foi salvando informações dos termos em grandes

Array, contudo ao compilar o programa, o sistema operacional bloqueava a execução por atingir um número muito elevado de uso de memória. O algoritmo foi reescrito algumas vezes para otimizar o uso de memória e a consulta de informações até conseguir processar 2100 exemplos divididos em três categorias, enquanto a expectativa inicial era de 60.000 exemplos divididos em seis categorias.

Para projetos futuros ficam melhorias como: melhorar o algoritmo de Stemming promovendo o uso de mais radicais para melhorar a tabela de exemplos; procurar otimizar mais o código de cálculo de frequência do TF-iDF, procurar testar em outras linguagens mais baixo nível como C e Pascal para promover um ganho maior em processamento; ampliar o número de categorias para mais de três; desenvolver ferramentas mais robustas para postar textos na internet com a utilização de técnicas de processamento digital de textos da inteligência artificial.

REFERÊNCIAS BIBLIOGRÁFICAS

ABRAHÃO P.R. Lima, V.L.S. **Um Estudo Preliminar a Metodologias de Análise Semântica da Linguagem Natural**. In: Anais do II Encontro para o Processamento Computacional de Português Escrito e Falado. XIII SBIA, Curitiba, CEFET-PR, 1996.

BARRETO, Jorge Muniz. **Inteligência Artificial: no limiar do século XXI**. 3. ed. Universidade Federal de Santa Catarina, Florianópolis, 2001.

DIAS, M. A. L., Malheiros, M. G.; **Extração Automática de Palavras-chave de Textos da Língua Portuguesa**. Centro Universitário UNIVATES. 2005.

GOLDSCHIDT, R. Passos, E. **Data Mining – Um Guia Prático**. Campus, 2005.

GOMIS, Lorenzo - Armañanzas, Emy y Noci, Javier Díaz. **Periodismo y Argumentación. Géneros de Opinión**, Barcelona, Universidade del Pais Vasco, 1996, pág. 77.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. 2. ed. Porto Alegre: Bookman, 2001.

JESUS, Martin Barbero; **Diáletica Escritura/Leitura in Dos Meios às Mediações.** Comunicação, Cultura e hegemonia. Rio de Janeiro, Editora UFRJ, 1997, pág. 183.

JURASFSKY D, Martin J.H. **Speech and Language Processing – An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**. Upper Saddle River: Prentice Hall. 2000.

LIU, Bing. **Web Data Mining: Exploring hyperlinks, Contents, and Usage Data. With 177 figures**. Chicago: Springer, 2010. xix, pág. 532.

LIVRO DE ESTILO. Disponível em: <http://static.publico.clix.pt/nos/livro_estilo/12-regras-c.html> Acesso em 10/09/2010.

MATSUBARA E. T. C, A. Martins M. C. Monard. **PreText: Uma Ferramenta para Pré-Processamento de Textos Utilizando a Abordagem bag-of-words**, Relatório Técnico ICMC-USP, 2003.

MENDEL J.M.; McLaren, R.W. **Reinforcement-learning Control and Pattern Recognition System**, 1994, pág. 287.

MITCHELL T. M. **Machine Learning**. McGraw-Hill Education (ISE Editions), 1997, Pag. 5 – 9.

OLIVEIRA, Fabio Abreu Dias de. **Processamento da Linguagem Natural: princípios básicos e a implementação de um analisador sintático de sentenças da língua portuguesa**. 2002.

ORENGO, V.M., Huyck, C. R., A **Stemminh Algorithim for The Portuguese Laguage. Proceedings of the SPIRE Conference**. 2001

QUINLAN, J. R. **Induction of decision trees. Machine Learning**. Pág. 81-106. 1986.

REZENDE, Solange Oliveira. **Sistemas inteligentes: fundamentos e aplicações**. 2003. Editora Manole Ltda. Barueri, SP.

RUSSEL, Stuart et al. **Inteligência Artificial**. 2.ed. Traduzido por: Vandenberg D. de Souza. Rio de Janeiro: Elsevier, 2004. 1021 p. Tradução de Artificial Intelligence.

SEBASTIANI, Fabrizio. **Machine Learning in Automated Text Categorization**. Consiglio Nazionale delle Ricerce, 2002. Itália, 2002. pág 47.

THORSTEN, J. **A Probabilistic Analysis of the Roccio Algorithm with TFIDF for Text Categorization**. Universitat Dortmund. 1997.

WEB, Univeridade Waikato. **Weka - machine lerning software in Java**. Disponível no site da Universidade de Waikato, 2009. http://www.cs.waikato.ac.nz/ml/weka.

VAPNIK, C. Cortes and V. Vapnik. **Support Vector Networks**. Machine Learning , 20:273 - 297, 1995.

CHARNIAK, Eugene. **Statistical Techniques for Natural Language Parsing**. AI Magazine 18(4):33–44. 1997.

CHAPMAN, Nigel P., **LR Parsing: Theory and Practice**. Cambridge University Press, 1987. ISBN 0-521-30413-X

AFANTENOS, S.D.; Doura, I.; Kapellou, E.; Karkaletsis, V. **Exploiting Cross Document Relations for Multi-document Evolving Summarization**. In the Proceedings of SETN, pp. 410-41. 2004.

GNU, **Linux Operating System 2010, WGET**, Disponível em: <http://www.gnu.org/software/wget/>. Acesso em 14 jun 2011.

MBAYER, **Martin Bayer 2006, HTML2TEXT**, Disponível em: <http://www.mbayer.de/html2text/readme.shtml>. Acesso em 14 jun 2011.

BRASIL ESCOLA, **IG 2002, ARTIGOS**, Disponível em: <http://www.brasilescola.com/gramatica/artigo.htm>. Acesso em 14 jun 2011.