ESCOLA POLITÉCNICA
DE PERNAMBUCO

TÉCNICA DE APRENDIZAGEM DE MÁQUINA PARA CATEGORIZAÇÃO DE TEXTOS.

Trabalho de Conclusão de Curso

Engenharia da Computação

Ericles Alves de Medeiros
Orientador: Prof. Dr. Adriano Lorena Inácio de Oliveira

Recife, junho de 2004

Ericles Alves de Medeiros

TÉCNICA DE APRENDIZAGEM DE MÁQUINA PARA CATEGORIZAÇÃO DE TEXTOS.

Resumo

Existe um aumento significativo na quantidade de documentos distribuídos eletronicamente, e ao mesmo tempo cresce a necessidade de meios que possam prover de forma rápida e eficiente a organização desses documentos, facilitando assim o acesso aos mesmos. Através da análise e interpretação dos dados que compõem o texto, é possível obter o conhecimento necessário para que as técnicas de aprendizagem de máquina sejam capazes de tomar algumas decisões quanto à classificação dos documentos em categorias pré-definidas. Porém, como os textos se apresentam em uma forma desestruturada de dados, se faz necessário um tratamento bastante cuidadoso a fim de transformar os documentos numa estrutura compreensível pelas técnicas de aprendizagem de máquina, podendo assim retirar as informações de maior relevância nos documentos. Este trabalho apresenta todo processamento executado com os documentos utilizados na categorização, bem como um estudo comparativo da técnica de aprendizagem Máquina de Vetor Suporte (SVM) sobre diferentes condições de treinamento, como alteração dos kernels utilizados, variação de alguns parâmetros de treinamento e utilização de diferentes modelos de atribuição dos pesos. Foi utilizado neste trabalho o pacote de rotinas TMSK [20] para auxiliar nas etapas de pré-processamento dos documentos e também na classificação linear dos documentos, porém o classificador automático que possibilitou uma melhor análise dos experimentos, implementando o algoritmo SVM, foi o SVM^{light} [19]. Os treinamentos executados neste trabalho tiveram como medidas de desempenho analisadas as taxas de precisão, cobertura e acurácia. Obtendo os melhores resultados na utilização da técnica SVM com o kernel RBF.

Abstract

There is a significant increase in the number of electronically delivered documents. At the same time it makes necessary ways to support quick and efficient organization of these documents, making easier its access. It is possible to obtain the necessary knowledge to make learning machine technique able to take decisions about the classification of text data in pre-defined categories. This necessary knowledge can be obtained through the text data analysis and interpretation. However, as these texts are unstructured, it becomes necessary a very careful approach to change them into a comprehensible structure for the learning machine technique. With the structured text created we can get all the relevant informations from them. This work presents all the procedure used with the documents in the categorization, as well as a comparative study of the learning technique Support Vector Machine (SVM) over different training conditions. Examples of the training conditions used are changes on the used kernels, variation on some training parameters and use of different kinds of weight attributions. We used in this study the pack TMSK [19] to help the data procedure and in the linear classification of the documents. But the automatic classifier wich provided the best results using the SVM algorithm to analyze the experiments was SVM light [18]. The training executed in this work used as performance parameters the precision tax and memory. Getting the best results using the SVM technique with the kernel RBF.

Sumário

Índice de Figuras

Índice de Tabela

Tabela de Símbolos e Siglas

AI – Artificial Inteligence (Inteligência Artificial)
RNA – Redes Neurais Artificiais
SVM – Support Vector Machine (Máquina de Vetor Suporte)
TMSK – Text Miner Software Kit (Pacote de Sistema de Mineração de Texto)
XML – Extensible Markup Language (Linguagem de Marcação Extensível)
TF – Term Frequency (Freqüência do Termo)
IDF – Inverse Document Frequency (Freqüência Inversa do Documento)
P_v – Exemplos Positivos Verdadeiros
P_f – Exemplos Positivos Falsos
N_v – Exemplo Negativos Verdadeiros
N_f – Exemplos Negativos Falsos
MCP – Modelo McCulloch e Pitts
SRM – Structural Risk Minimization (Minimização do Risco Estrutural)

Agradecimentos

Agradeço profundamente ao meu fiel amigo Deus, ao qual confio todos os dias a minha vida e devo tudo que tenho e sou.

Essa minha mais nova conquista, agradeço muito a:

Minha família, em especial a minha mãe Nereusa, pelo esforço e confiança em todos os momentos. A minha avó Rosa pelo seu jeito acolhedor e carinhoso.

Minha segunda família, minha noiva Fabiana, por toda dedicação e paciência nunca vista igual, minha sogra Bernadete, com seu jeito carinhoso e incentivador, e minha cunhada Aninha, por tanta bondade e alegria. Agradeço a você Aninha pelo computador "emprestado", pois sem ele o caminho ficaria mais difícil.

Todos os meus colegas da empresa Segsat, em especial aos do meu setor de desenvolvimento, que tanto me agüentam.

Meus colegas da Escola Politécnica, que compartilharam vários momentos tristes e alegres de minha vida. Porém, não posso deixar de citar alguns que me ajudaram bastante nesta fase alcançada, como os companheiros Ronaldo, George, Miguel, Tiago Batista, Daniel e Thyago, que me apoiaram nas fases críticas do trabalho e sempre com uma sugestão amiga e incentivadora.

Senhora Edenira Barbosa, mãe do companheiro Thyago, por toda dedicação quando íamos até altas horas de estudo. A Tia Dodora e Graça pelo incentivo e torcida. Agradeço também à companheira Amanda pela força dada no desenvolver desse trabalho.

Professores e colaboradores do curso de Engenharia da Computação da POLI, pela ótima formação oferecida a nós alunos. Em especial ao professor Carlos Alexandre, pois numa fase difícil do curso, sua voz experiente me guiou em busca desse fruto hoje colhido.

Meu orientador, Adriano Lorena Inácio de Oliveira, por toda compreensão e orientação bem exercida.

Por trás de toda decisão precisa na minha vida, existe sempre alguém em um plano mais alto a me guiar. E mesmo que não possamos mais nos ver ou abraçar, viveremos para sempre esse nosso eterno amor.

Dedico este meu trabalho ao meu pai, Elias Alves de Medeiros, que faleceu durante o período de minha faculdade, deixando bastante saudade e uma lição de vida.

Te amo Pai !

Capítulo 1

Introdução

Existe uma crescente quantidade de informação textual no formato digital. Textos que compõem o dia-a-dia de empresas e de pessoas estão sendo armazenados eletronicamente, assim como novas páginas contendo textos são lançadas diariamente na web. Ao mesmo tempo em que existe esse crescimento, as dificuldades na organização dessas informações distribuídas eletronicamente também aumentam. Por isso métodos que possam prover organização, recuperação, filtragem e controle desses dados digitais estão sendo bastante explorados. Uma técnica chave na organização de documentos é a categorização de texto, a qual se baseia em características (palavras) extraídas no texto para associação de um documento, pertencente a um determinado domínio, em uma ou mais classes pré-definidas. Como a quantidade de palavras relevantes no documento é grande e cada uma ocupa uma dimensão no espaço de representação dos dados, gera-se um espaço de alta dimensionalidade para representação das características.

Aplicações em categorização de texto datam dos anos 60 e se expandiram de outras áreas como Recuperação de Informação e Filtragem de Informação, tornando-se uma área de pesquisa própria nos anos 80 [29]. Nos anos 90, o aprendizado de máquina começou a se popularizar e fazer parte do processo de categorização, apresentando-se de uma forma tão cuidadosa quanto à categorização dirigida por especialistas, e ao mesmo tempo sendo mais viável pela rapidez e ausência do especialista para realização da tarefa de categorização. Técnicas de Inteligência Artificial (AI–*Artificial Inteligence*) estão sendo bastante utilizadas para automatizar o processo de categorização, entre elas Redes Neurais Artificiais (RNAs), que são caracterizadas por se apresentarem como uma forma de computação não algorítmica e que são semelhantes, em algum nível, à estrutura do cérebro humano [1].

Ferramentas de aprendizado de máquina vêm sendo bastante exploradas, de forma que são treinadas para oferecer suporte à classificação baseada nos dados do treinamento. Um método novo de aprendizagem de máquina e bastante utilizado na construção de classificadores automáticos de texto é a máquina de vetor de suporte (*Support Vector Machine*-SVM) [22][23] [24], proposto por V. Vapnik [25][26]. A grande utilização dessa técnica se deve à sua habilidade de trabalhar com um espaço de características de alta dimensionalidade presente em cada texto de forma bem robusta. SVM tem obtido sucesso não apenas na categorização de documentos, como também em um grande número de aplicações, como identificação de partículas e fazes, bioinformárica e banco de dados de marketing [2].

O objetivo da categorização de textos é a classificação de documentos dentro de um número de categorias pré-fixadas [23], podendo cada documento pertencer a múltiplas categorias ou nenhuma, assim organizando os documentos por categorias. A fim de alcançar os objetivos mencionados na categorização de texto, são usualmente utilizadas as seguintes fases [29]:

1. Coleta de documentos: nesta fase se obtêm os dados para treinamento do método de aprendizagem de máquina escolhido.

2. Pré-processamento: nesta fase é preparada cuidadosamente a estruturação das informações, que se encontram originalmente como uma coleção de documentos desestruturados, a fim da obtenção de informações do texto.

3. Fase de Treinamento: são apresentados alguns documentos previamente categorizados à técnica de aprendizado de máquina, a fim de se extrair o conhecimento necessário para categorizar novos documentos.

4. Classificação: documentos são organizados nas categorias pré-existentes com base no aprendizado adquirido com documentos anteriores.

1.1 Motivação

A categorização de textos tem sido bastante explorada na automatização de aplicações nas áreas de indexação automática para sistemas de recuperação de informação, organização de documentos e filtragem de textos. Algumas aplicações práticas nas áreas mencionadas são [29]: (1) organização de mensagens de e-mail, (2) filtro de mensagens e notícias, evitando *Spam*, e (3) organização e recomendação de documentos. Inúmeros sistemas de busca na internet utilizam técnicas baseadas em palavra-chave, o que gera muitas informações sem relevância retornadas ao usuário. Isto é devido à falta de uma análise mais detalhada do conteúdo dos documentos. A categorização de textos tem se tornado uma alternativa nas buscas pela rede, uma vez que os documentos são organizados cuidadosamente em categorias pré-estabelecidas de acordo com uma análise mais detalhada do conteúdo que os compõe.

Grande parte dos textos técnicos, científicos, notícias e outros textos disponibilizados na web se encontram na língua inglesa, sendo necessário levar em consideração a língua em que os mesmos se encontram para que a análise de seu conteúdo possa orientar a organização dos documentos. A utilização de especialistas para realizar categorizações manuais é um processo custoso e lento, o que motiva a investigação de técnicas de categorização automática de documentos utilizando técnicas de aprendizagem de máquina. Os resultados obtidos pela classficação automática de textos têm se mostrado bastante satisfatórios, mas mesmo quando não apresentam uma precisão alta, sistemas de categorização automática podem ser usados para realizar pré-filtragens dos textos, auxiliando o trabalho dos especialistas.

Uma preparação dos documentos bem elaborada e cuidadosa na fase de pré-processamento, assim como ajustes realizados em alguns parâmetros do algoritmo de aprendizado utilizado para extração de informação dos documentos, pode ter grande influência nos resultados provenientes da aplicação da técnica de aprendizagem. Dessa forma se encontram abordadas de forma clara as etapas seguidas na preparação de documentos e experimentos realizados.

Neste trabalho é realizada a categorização automática de documentos no padrão XML, utilizando a técnica de aprendizagem SVM sobre diferentes condições de treinamento do algoritmo. Cada documento é classificado como pertencente ou não pertencente à categoria em análise, organizando assim os documentos nas suas respectivas categorias. Na Figura 1 é visto um exemplo de categorização de um novo documento pertencente à categoria Alcool.

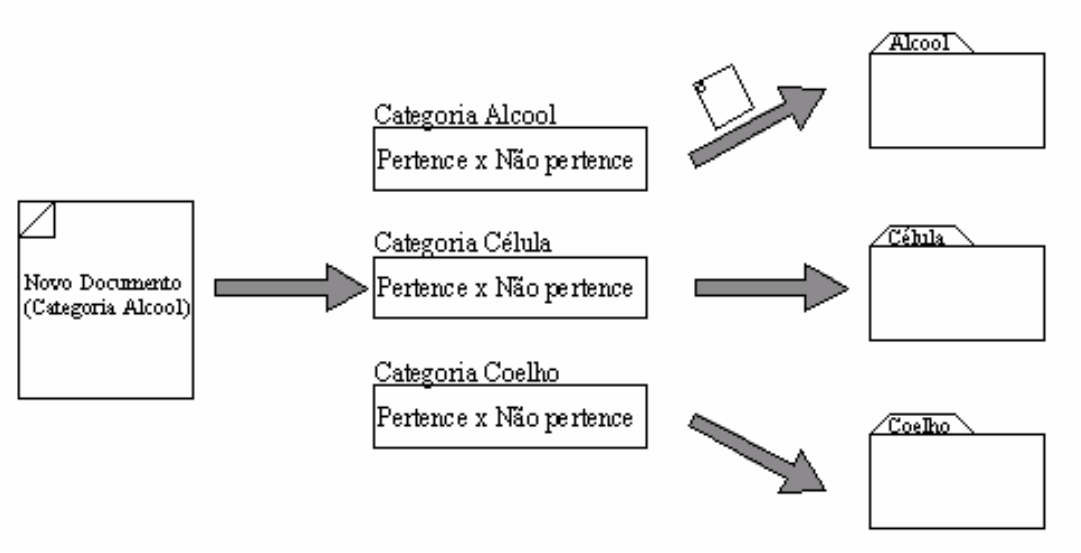

Figura 1. Exemplo de Categorização de Texto

1.2 Objetivos do Trabalho

O objetivo deste trabalho é fazer um estudo experimental da categorização de documentos utilizando aprendizagem de máquina, baseando-se na utilização do método Support Vector Machine (SVM) para categorização de textos no formato digital, como também validar a utilização do aprendizado de máquina na categorização de documentos. A aprendizagem de máquina com a técnica SVM possui alguns parâmetros que podem influir nos resultados, como o tipo do kernel, parâmetros utilizados por cada kernel e o modelo de atribuição dos pesos nas palavras, neste trabalho foi realizado um estudo da influência destes parâmetros na categorização dos documentos. Outro objetivo deste trabalho é analisar os problemas encontrados na utilização de dados textuais pela aprendizagem de máquina, abordando o processamento necessário a ser realizado nos documentos, a fim de transformá-los em um formato compatível para extração de conhecimentos pela técnica de aprendizagem utilizada. As operações aplicadas nos documentos e posterior aplicação da técnica SVM mostram como resultado uma documentação com dados estatísticos dos testes e descrição detalhada do funcionamento das operações realizadas nos documentos, com a finalidade de trabalhos posteriores serem gerados.

1.3 Trabalhos Relacionados

Algumas abordagens têm sido propostas para a categorização automática de documentos a partir de técnicas de aprendizagem de máquina. Nesta seção serão citados alguns trabalhos relacionados, como:

- C. Oliveira e P. Castro [15], propõem a aplicação de Árvores de decisão para a categorização múltipla de textos. O processo de aprendizado é efetuado sobre todas as categorias dos textos, gerando um único classificador capaz de associar cada novo documento a uma ou mais categorias pré-definidas. São usadas para os experimentos as

coleções *Reuters – 21578* (um conjunto de 21.578 artigos da agência de notícias *Reuters*, devidamente classificados).

- J.Thorsten [23], realiza uma análise das propriedades particulares do aprendizado de máquina com dados textuais, relacionando algumas destas ao bom desempenho obtido com a técnica de aprendizagem de máquina SVM. O autor realiza também alguns experimentos comparativos do desempenho da técnica, utilizando kernels RBF e polinomial com a coleção de dados *Reuters – 21578.*

- F.Sebastiani[18], analisa comparativamente classificadores obtidos de diferentes técnicas de aprendizagem de máquina sobre cinco versões da coleção *Reuters*.

Diferentemente dos trabalhos relacionados, foi gerado uma documentação bem detalhada das etapas necessárias para se aplicar aprendizagem de máquina em textos, como também foram separados padrões da coleção de dados que se apresentavam em maior quantidade, gerando três categorias predominantes nos documentos, para que assim fosse verificado o comportamento do método de aprendizagem SVM sobre diferentes condições de treinamento do classificador.

1.4 Estrutura do Trabalho

Este trabalho é composto inicialmente no capítulo 2 por uma introdução da seqüência de operações realizadas para tratamento de documentos digitais e extração de características, a fim de alcançar uma estrutura de dados compreensível pela técnica de aprendizagem de máquina utilizada. Ainda no capítulo 2 o leitor poderá se familiarizar com a ferramenta utilizada nas etapas do processamento, sendo abordado também o formato dos dados utilizados e gerados nas etapas operações.

No capítulo 3 é descrita a técnica de classificação automática de texto SVM, assim como uma implementação da mesma, feita por J.Thorsten [22]. Neste capítulo ainda são abordados os kernels com seus respectivos parâmetros, utilizados no modelo SVM. O kernel é a parte do núcleo dos classificadores, responsável por mapear as entradas no espaço de saída.

Uma vez descrita a técnica de classificação, parte-se no capítulo 4 para a descrição dos experimentos realizados com os parâmetros selecionados, apresentando ao leitor a descrição do manuseio dos arquivos para atingir uma estrutura de dados compatível com os classificadores, bem como a análise dos resultados obtidos ,tendo-se por fim uma comparação das técnicas abordadas no trabalho.

O capítulo 5 proporciona ao leitor a visão de algumas conclusões obtidas com a elaboração do trabalho, assim como trabalhos futuros que podem ser explorados nessa linha de pesquisa.

Capítulo 2

Processamento de Documentos Digitais para Categorização

Existe uma grande quantidade de textos em formato digital, apresentando-se geralmente como uma coleção de dados com pouca ou nenhuma estruturação e geralmente contendo milhares de palavras e termos, o que dificulta o manuseio desses arquivos de texto para o processo de obtenção de informação dos documentos. Tipicamente as técnicas de extração de características do texto utilizam informações estruturadas, que são cuidadosamente preparadas por processos ou podem ser colecionados baseando-se em um formato previamente estabelecido. Geralmente o processo de extração de informações do texto gera pares de atributos e valores que representam o conteúdo dos documentos [29].

Nesta parte do trabalho será apresentado o formato dos dados considerados neste trabalho, assim como as etapas seguidas na estruturação dos documentos e obtenção de informações do texto, permitindo ao leitor a visualização das transformações obtidas com os documentos ao término de cada etapa. Por fim, descrevemos como se obtém uma estrutura de dados compreensível pelas técnicas de aprendizado de máquina utilizadas no trabalho.

2.1 Processo de Mineração de Texto (*Text Mining*)

Mineração de Texto *(Text mining),* também conhecido como *Text data mining* [7], é o processo de extrair conhecimento a partir de documentos contendo textos não estruturados, utilizando esse conhecimento adquirido para melhor organizar as informações para uma referência futura. Alguns materiais de mineração de texto como os produzidos pela empresa IBM, consideram a mineração de texto como a aplicação da mineração de dados para textos não estruturados [9]. A mineração de texto assim como a mineração de dados baseia-se em amostras de exemplos passados, mas a composição desses exemplos é bastante diferente nos dois casos. No entanto, a similaridade dos métodos de aprendizagem nos processos de mineração de dados e de texto justifica a transformação dos textos em uma representação numérica similar à utilizada em mineração de dados[29], pois esses métodos trabalham com uma estrutura de dados seguindo uma representação numérica. Na Tabela 1 pode-se visualizar algumas comparações entre os processos de mineração de dados e mineração de texto.

Tabela 1. Comparação entre Mineração de dados e Mineração de Textos [29].

	Mineração de Dados	**Mineração de Texto**
Objeto de investigação	Dados categóricos e numéricos	Textos
Estrutura do objeto	Bases de dados Relacionais	Textos em formato livre
objetivo	Prever resultados de situações Futuras	Recuperar informações relevantes, purificar o significado, categorizar o resultado.
Métodos	Conhecimento de máquina: Redes Neurais, Árvores de decisão, Algoritmos genéticos e outros.	Indexação, processamento especial de redes neurais, lingüística e ontologias.

O processo de Mineração de texto possui duas fases principais e seqüentes: a extração de informação e a mineração de dados. A primeira fase tem papel bastante importante no processo de mineração, pois permite o tratamento da representação de dados e tem como objetivo extrair conceitos, estatísticas e palavras relevantes de um conjunto textual, visando proporcionar uma estrutura mínima. Na segunda fase são aplicadas diretrizes e algoritmos de mineração de dados destinados a gerar regras, classificações ou agrupamentos (Figura 2).

O completo entendimento da linguagem natural dos textos não pode ser atingido pela mineração de texto, mas esta se foca na obtenção de pequena quantidade de informação como título, autor e referências. Porém a mineração de texto deve preocupar-se com alguns problemas que podem ocorrer no processamento da linguagem natural, tais como: problemas de normalização das palavras, como variação das palavras, números e flexões verbais.

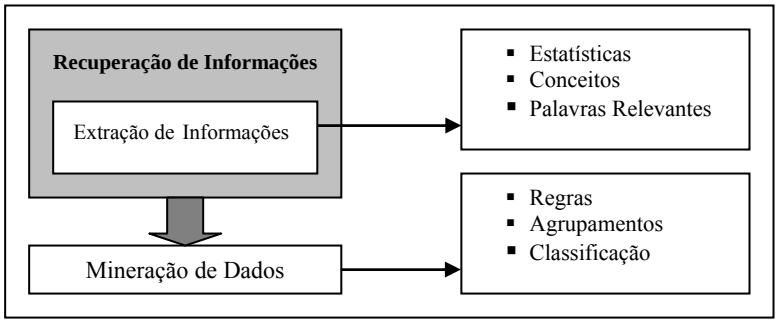

Figura 2. Técnica de Mineração de Texto.

2.2 Etapas do Processo de Mineração de Texto

Para minerar o texto é preciso transformá-lo em uma forma estruturada de dados que possa ser utilizada pelas técnicas de mineração de dados [29]. A preparação dos textos é o passo inicial no processo de descoberta de conhecimento em textos. Esta etapa envolve a seleção das bases de texto que constituirão os dados de interesse e o trabalho de seleção de uma parte do texto que melhor expresse o conteúdo do mesmo, ou seja, toda informação presente no texto que for irrelevante para sua categorização deve ser desprezada. Após a preparação dos textos eles devem ser transformados em uma forma de representação de dados que seja compreensível pelas técnicas de mineração de texto para extração do conhecimento. Por fim, o conhecimento é avaliado para que os resultados sejam validados, essa avaliação pode ser feita através da análise

de algumas métricas oferecidas nos resultados e com a ajuda do conhecimento de especialistas para a consolidação do conhecimento.

Abordaremos nesta parte do trabalho, como as etapas da mineração de texto são realizadas. Na Figura 3, inicialmente é mostrado como o pré-processamento dos dados é realizado para preparar o conjunto de dados para as fases posteriores de processamento dos dados e análise dos resultados obtidos.

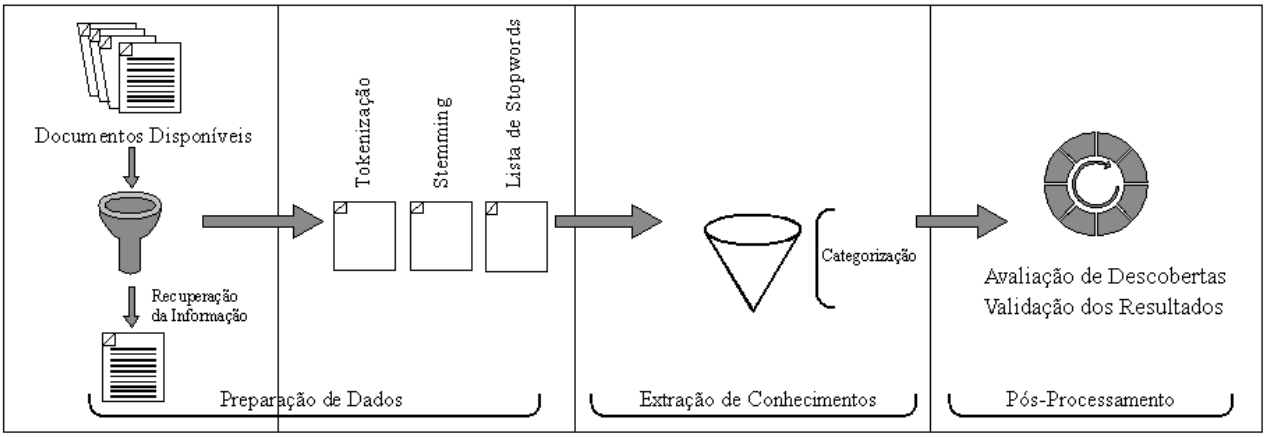

Figura 3. Etapas da Mineração de Texto.

Na Figura acima pode ser visto o esquema básico do processo de mineração de texto com os passos que podem compor cada etapa. Na etapa de preparação dos dados, os documentos que vão participar da categorização são coletados e tratados, a fim de atingir uma estruturação que permita a extração de conhecimento. A etapa de extração de conhecimento é onde as tarefas cabíveis ao objetivo da aplicação de mineração de texto são executadas, no caso deste trabalho a tarefa executada é a categorização dos documentos. A última etapa corresponde ao pós-processamento, que é onde os resultados são analisados e validados.

Neste trabalho para a realização da etapa de preparação dos dados, no processo de mineração de texto, foi utilizado o *Text Miner Software Kit* (TMSK) [20], que é um pacote de software para mineração de textos disponibilizado gratuitamente para os leitores do livro *Text Mining* [7]. Uma especificação mais detalhada do funcionamento e funcionalidades do sistema TMSK, pode ser visto na Seção 2.3.

2.2.1 Coleta de Documentos

O primeiro passo na mineração de texto é coletar os dados que serão utilizados, ou seja, aqueles documentos que são relevantes para o treinamento [29]. Na tarefa de categorização os documentos mais relevantes são aqueles que apresentam as palavras com maior relevância nas categorias, ou seja, as palavras de maior peso para categorizar. Estes documentos relevantes podem ser fornecidos ou podem ser parte da descrição do problema. Algumas vezes os documentos podem ser obtidos através de bancos de dados, mas nesses casos deve-se analisar a credibilidade dos documentos fontes.

Outro recurso de fornecimento dos documentos que pode ser considerado também é a *World Wide Web*. Coleções de páginas de um tópico particular podem ser construídas. Entretanto, o principal problema deste método é que os dados podem ser de qualidade dúbia. Isto implica ser necessária extensiva limpeza dos dados antes do uso.

Algumas vezes o conjunto de documentos apresenta-se bastante grande e técnicas podem ser utilizadas para selecionar os documentos de maior relevância no conjunto, a técnica utilizada

irá depender da aplicação, pode-se citar como exemplo a análise de relevância do documento em relação ao tempo, ou seja, um documento mais recente tendo relevância maior.

Para pesquisas e desenvolvimento de técnicas de mineração de texto, alguns dados genéricos podem ser necessários, estes dados são geralmente chamados de *Corpus*. Como é importante se ter uma base com bastantes textos para efeito do estudo, muitas organizações e iniciativas estão em andamento para coordenar atividades e prover mecanismos para agrupamento desses textos na base de dados. Duas das principais iniciativas são: *Linguistic Data Consortium* (LDC) na universidade da Pennsylvania, Estados Unidos da América e o *International Computer Archieve of Modern and Medieval English* (ICAME) em Bergen, Noruega [29].

Os documentos utilizados nos experimentos do presente trabalho foram escolhidos do corpus total de 348.566 documentos da coleção OHSUMED [11], que é de uma coleção de documentos pertencente à base de dados digital de informações médicas MEDLINE [10]. Os documentos da coleção OHSUMED consistem de dados recolhidos de 270 revistas científicas médicas ao longo de cinco anos (1987-1991), por William Hersh e colegas para experimentos[8] Eles apresentam uma estruturação própria dos dados como pode ser visto na Figura 4. Esta estruturação das informações contidas nos documentos é feita através dos campos descritos abaixo:

- .I - Identificador seqüencial
- .U - Identificador MEDLINE
- .M - Tópicos abordados no texto
- .T - Título
- .P - Tipo de Publicação
- .w - Resumo
- .A - Autor
- .S – Fonte

```
.I 54711
.U
88000001
.S
Alcohol Alcohol 8801; 22(2):103-12
.M
Acetaldehyde/*ME; Buffers; Catalysis;
.T
The binding of acetaldehyde to the active site of
ribonuclease: alterations in catalytic activity and effects of phosphate
.P
JOURNAL ARTICLE.
.W
Ribonuclease A was reacted with
[1-13C,1,2-14C]acetaldehyde and sodium
cyanoborohydride in the presence or absence
of 0.2 M phosphate. After several hours of
.....
.A
Mauch TJ; Tuma DJ; Sorrell MF.
```

Figura 4. Documento da Coleção OHSUMED.

2.2.2 Padronização dos Documentos

É bastante comum encontrar uma grande variedade de formatos diferentes nos documentos digitais, que depende do processo de geração desses documentos. Alguns documentos podem ter sido gerados por processadores de palavras com formatos próprios ou até mesmo por um simples editor de texto e salvos como um texto ASCII. Como o objetivo é executar um processamento comum a todos os documentos coletados, é de extrema importância convertê-los para um formato padrão que possibilite as operações de extração de características dos textos. A principal vantagem de padronizar os documentos é que as ferramentas de mineração podem ser aplicadas sem levar em consideração a origem dos mesmos.

Muitas das comunidades de processamento de texto adotam o formato XML (*Extensible Markup Language*) como padrão para a conversão dos documentos a serem processados [29]. O padrão XML estabelece uma linguagem que possibilita ao usuário a identificação das estruturas do texto através da criação de *tags* (etiquetas ou rótulos). Porém, XML não especifica mais nada a respeito do significado das estruturas. A principal razão para identificar as estruturas constituintes dos documentos é selecionar as partes que possuem relevância no processo de geração de características do texto.

Alguns dos campos presentes na base OHSUMED não apresentam relevância no processo de categorização dos documentos, com base na análise de seus conteúdos, podendo ser descartados na representação de dados. Para os experimentos deste trabalho os campos de interesse nos documentos são: **.M**, **.T** e **.W**. O formato dos documentos aceitos pela ferramenta TMSK é o XML. Dessa forma foi necessário um programa Java para transformar os textos da base de dados OHSUMED para o formato XML. Este programa foi desenvolvido neste trabalho de conclusão de curso e mostrado no Apêndice A.

Na Figura 5 mostramos um exemplo de documento XML gerado após a aplicação do programa Java ao documento da Figura 4. Note que este padrão destaca melhor as estruturas de interesse no texto para extração das características que servirão na categorização, como o texto definido pelas tags <TITLE></TITLE>,<TEXT></TEXT> e <TOPICS></TOPICS> .

```
<DOC>
<TITLE>
The binding of acetaldehyde to the active site of ribonuclease:
alterations in catalytic activity and effects of phosphate.
</TITLE>
<AUTHOR>
Mauch TJ; Tuma DJ; Sorrell MF.
</AUTHOR>
<TOPICS>
<TOPIC>
Acetaldehyde
</TOPIC>
<TOPIC>
Buffers
</TOPIC>
<TOPIC>
Catalysis
</TOPIC>
</TOPICS>
<TEXT>
Ribonuclease A was reacted with [1-13C,1,2-14C]acetaldehyde
and sodium cyanoborohydride in the presence or absence of
0.2 M phosphate. After several hours of
</TEXT>
</DOC>
```

Figura 5. Documento no Padrão XML

2.2.3 Preparação dos Dados

Utilizar arquivos de texto apresenta alguns desafios a serem tratados, começando pela falta de estruturação, repetição de palavras com mesmo sentido e palavras irrelevantes para categorização presentes no texto. Outro fator que não pode ser esquecido é a quantidade de palavras relevantes que aparecem nos documentos, pois estas são quem determinam a dimensão no espaço de representação dos documentos. Por isso deve-se tratar com bastante cuidado os dados textuais para que o uso indevido destes não comprometa as atividades de categorização.

2.2.3.1 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 como tabulação, espaço e nova linha. Para obter as melhores características presentes no texto, os processos de tokenização são sempre otimizados para avaliação dos textos utilizados, pois o processo de tokenização é dependente da linguagem, sendo os princípios gerais os mesmos, porém os detalhes são diferentes. O foco principal do trabalho são documentos na língua inglesa devido à disponibilidade dos documentos categorizados, no entanto, para outras linguagens o processo de tokenização segue os mesmos princípios gerais. Na Figura 6 está representado o processo geral de identificação e extração das palavras do texto através da operação *Tokenizer*, considerando as etapas seguintes para se obter relevância nos dados obtidos como representação.

No pré-processamento executado neste trabalho com as bases de texto no formato XML, o processo de tokenização foi executado pela ferramenta TMSK [20], levando em consideração os *tokens* delimitadores e caracteres delimitadores, como “\n”, “ \t”, “ \r”, “.”, “;”, “:”, “!”, “?”, “(”, “)”, “<”, “>”, “[”, “]”, “+”, “\”.

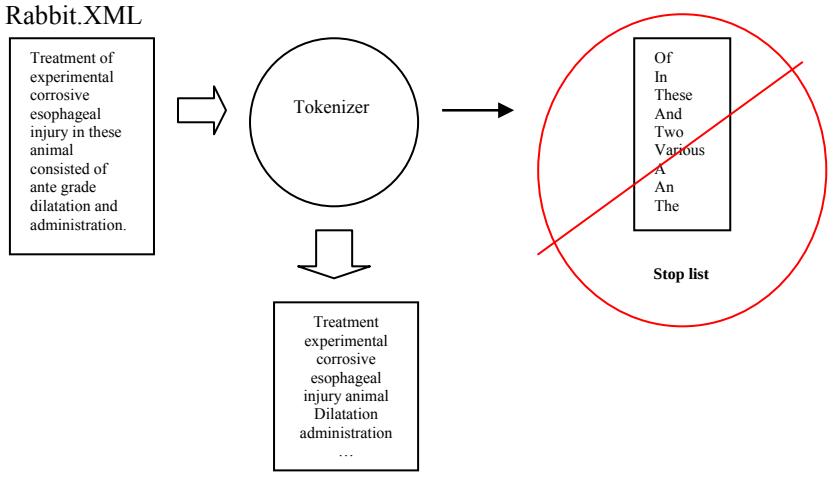

Figura 6. Processo de Extração das Palavras.

2.2.3.2 Stopwords

Outra tarefa na preparação 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 constitui conhecimento no texto. O resultado é uma lista com as palavras a serem descartadas (conhecido também como *Stoplist*). Este conjunto de palavras é chamado de *Stopwords*.

Stopwords são palavras que não apresentam um conteúdo semântico de significância 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, eles, elas*), que não fornecem nenhuma informação que venha a representar conteúdo dos textos. Na construção de uma lista de *stopwords* são acrescentadas palavras como preposições, pronomes, artigos e também podem ser adicionadas palavras que apresentam uma incidência muito alta em uma coleção de documentos e que, portanto não apresentam influência na categorização dos textos.

A análise dos dados textuais neste trabalho utiliza a ferramenta TMSK, que para executar a rotina *mkdict*, responsável pelo processo de criação do dicionário de palavras de relevância nos documentos, utiliza uma lista de *stopwords* passada como parâmetro no arquivo de propriedades da ferramenta, para assim desconsiderar as palavras que não apresentam significância no contexto dos documentos nas etapas posteriores. Pode ser visto na Figura 7 um exemplo de lista de *stopwords*, onde as palavras são dispostas uma por linha, para assim serem excluídas na formação do dicionário de palavras.

Figura 7. Exemplo de Dicionário de *Stopwords*.

2.2.3.3 Stemming

Esse processo é realizado considerando cada palavra de forma isolada. O objetivo é remover o sufixo e prefixo dos termos que possam vir a representar variação verbal ou plural, para que assim, sejam gerados apenas os radicais de acordo com as regras gramaticais da linguagem utilizada. *Stemming* auxilia bastante o processo de filtragem e classificação de documentos, pois reduz o número de termos diferentes nas representações dos documentos e impede a repetição de palavras com mesmo radical que possam vir a atrapalhar a definição dos pesos para os termos do texto. Algoritmos de *Stemming* empregam regras lingüísticas e são dependentes do idioma. Estes algoritmos não costumam usar informações de contexto para determinar o sentido de cada palavra, a maioria das palavras é admitida como apresentando um significado único.

No processamento executado nos textos utilizados neste trabalho, o processo de stemming acontece internamente na rotina *mkdict* da ferramenta TMSK [20]. Variações das palavras que não apresentam o mesmo radical da palavra raiz, são declaradas no dicionário de *stems* no

arquivo de propriedades da ferramenta. Pode-se ver na Figura 8 um *stem* dicionário simples, onde cada linha apresenta duas palavras separadas por um espaço. Em cada linha, a primeira palavra representa uma variação da segunda palavra raiz. Deste modo, a primeira palavra é substituída pela segunda no processo de *stemming*. Analisando a construção de um dicionário de *stemming* para a língua portuguesa, poderia ser inserida como exemplo a linha contendo "criado criar", onde a ocorrência da primeira no documento, como sendo uma variação verbal, implicaria na substituição pela segunda palavra "criar".

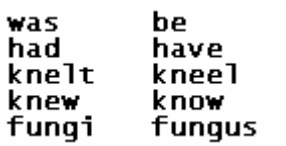

Figura 8. Exemplo de Dicionário de *Stemmer*.

2.2.3.4 Dicionário de Características

Considerando o problema abordado neste trabalho, que tem como objetivo a organização de documentos em categorias pré-definidas, é necessário agrupar as características que representam maior relevância nos documentos em um conjunto denominado *dicionário*. No dicionário estão presentes as palavras que têm relevância nos documentos pela freqüência de aparição nos textos. Estas palavras tornam-se uma base para a formação de uma estrutura compreensível pelas técnicas de classificação. As palavras presentes no dicionário associadas à sua freqüência de aparição no texto fornecem informação para construção de uma estrutura compreensível pelas técnicas de aprendizagem de máquina.

A construção de dicionários a partir dos documentos pode ser feita através de uma visão global, considerando-se as palavras que aparecem com maior freqüência em todos os documentos da base de dados, o que gera uma representação dos textos com alta dimensão de características, pois utilizou as palavras relevantes de todos os documentos como forma de representação. Algumas transformações podem ser executadas nos dicionários para redução de seu tamanho. Dependendo do método de aprendizado utilizado na categorização, isto pode ajudá-lo a obter melhor desempenho na categorização [29]. Podemos citar algumas dessas transformações como:

1. Construção de dicionário a partir de uma visão local, ou seja, considerar apenas as palavras de relevância nos documentos pertencentes à categoria que se deseja categorizar novos documentos.

2. Retirar do dicionário as palavras que constam na lista de *stopwords*, por não apresentarem relevância para a categorização.

3. Determinar número mínimo de freqüência para que palavras presentes no texto sejam incluídas no dicionário de palavras.

4. Eliminação de palavras como plural de uma outra palavra ou variações verbais, através da lista de *stems*

Com a ferramenta TMSK utilizada neste trabalho podemos gerar o dicionário de palavras da nossa base de dados tanto de uma perspectiva geral, para todos os documentos, como também para uma análise local, levando em consideração apenas os documentos pertencentes a uma categoria. A rotina responsável pela geração do dicionário elimina as palavras presentes na lista de *stopwords* e *stems* passadas no arquivo de configuração da ferramenta. O formato do dicionário gerado pela ferramenta pode ser visto na Figura 9, apresentando uma série de palavras com uma por linha, as palavras compostas por mais de uma e separadas por hífen, são separadas pelo caractere ‘: ’ no dicionário.

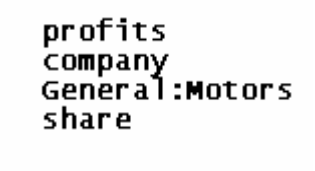

Figura 9. Exemplo de Dicionário de Características.

2.2.4 Representação dos Dados

Após a construção do dicionário das palavras que apresentam relevância nos documentos, partimos então para a representação dos dados através de um modelo compreensível pelas técnicas de categorização utilizadas.

2.2.4.1 Modelo do Espaço Vetorial

As técnicas de categorização abordadas neste trabalho seguem o modelo do espaço vetorial, desenvolvido por Gerald Salton e colaboradores [17]. Neste modelo cada documento é representado por um vetor numérico de termos, em que cada termo possui um peso associado que indica a sua importância no documento. Assim, cada documento é representado por um vetor constituído por elementos do dicionário e o peso associado. Os termos do dicionário que não estão no documento apresentam peso zero, ou seja, não precisam ser representados no vetor.

Existem diversas formas para se calcular o peso de cada termo, o que será apresentado na seção seguinte, porém a maior parte das técnicas de categorização se baseia na freqüência com que o termo aparece em cada documento para atribuição dos pesos [29].

Cada elemento do vetor é considerado uma coordenada, assim o documento quando colocado no espaço euclidiano, assume tantas dimensões quantas sejam as palavras que constituem sua representação vetorial. A posição do documento em uma dada dimensão depende do peso da palavra associada àquela dimensão. Documentos com os mesmos termos estão localizados numa mesma região do espaço, ou seja, querem apresentar conteúdos similares [17].

Podemos ver na Figura 10 a representação de três documentos no modelo espaço vetorial.

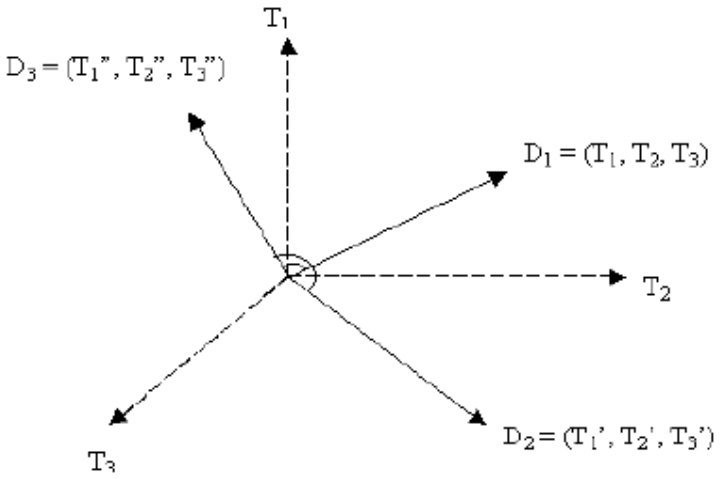

Figura 10. Representação de Documentos no Modelo do espaço Vetorial [3].

2.2.4.2 Indexação

Uma das principais dificuldades em trabalhar com texto como já foi citado anteriormente, é a grande quantidade de palavras. Isto gera dicionários bastante extensos e certa dificuldade na classificação devido à grande dimensão espacial dos vetores. O processo de indexação é responsável por converter os textos em um formato que proporcione buscas rápidas a elementos do texto, eliminado o processo seqüencial e lento da exploração por elementos do texto. Esse processo gera como saída um índice, que pode ser uma estrutura de dados que proporcione o acesso rápido às palavras armazenadas no documento, funcionando como o índice de um livro que possibilitam ao leitor a visualização rápida das páginas que discutem determinados tópicos.

2.2.4.3 Atribuição de Pesos (*Weighting*)

No modelo do espaço vetorial abordado anteriormente pudemos ver que os documentos são representados como vetores de termos e consequentemente uma coleção de documentos será representada por uma matriz X com dimensão igual a: número de termos x número de documentos. Cada elemento dessa matriz corresponderá à ocorrência de um termo do dicionário em um documento da coleção, acompanhado com seu peso no documento.

O peso agregado a cada termo do vetor pode ser determinado de diversas formas, porém todas são baseadas em duas observações sobre o texto:

- Por uma observação local do documento, nota-se que quanto mais vezes um termo ocorre, mais relevante para o tópico do documento ele se torna.
- Por uma visão geral da coleção de documentos, pode-se perceber que quanto mais o termo ocorre dentre os documentos da coleção, menos importante ele se torna para diferenciar os documentos.

Podemos verificar abaixo alguns métodos de determinar o peso de um termo baseando-se no seu número de ocorrências, na freqüência de ocorrência no documento e levando-se também em consideração o cálculo do peso em relação à coleção de documentos.

Modelo Booleano

Este modelo é bastante simples, baseia-se na atribuição de pesos 0 ou 1 para representar respectivamente a ausência ou presença do termo no documento, Onde f_{ik} corresponde à freqüência do termo no documento.

$$a_{ik} = \begin{cases} 1, \ se \ f_{ik} > 0 \\ 0, \ caso \ contrário \end{cases} \tag{2.1}$$

Modelo da Freqüência da Palavra

Este modelo também é bastante simples e considera que a freqüência do termo, quantidade de aparições do termo, no documento determina sua importância no documento.

$$a_{ik} = f_{ik} \tag{2.2}$$

Modelo TFIDF (*Term Frequency / Inverse Document Frequency*)

Este método leva em consideração a freqüência do termo em relação a todos os documentos da coleção, o peso é associado ao termo na proporção da freqüência do termo no documento e na proporção inversa da quantidade de documentos em que o termo aparece.

$$a_{ik} = f_{ik} * \log\left(\frac{N}{n_i}\right) \tag{2.3}$$

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 freqüência do termo no documento e o resultado a_{ik} é o peso atribuído ao termo.

O modelo TFIDF considera a freqüência de cada termo em todos os documentos, como medida inversa da sua capacidade de representar especificamente cada documento [21], ou seja, em 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 que outro termo aparece, esse termo que aparece em menos documentos terá um maior peso de decisão na categorização.

Neste trabalho, além dos modelos booleano e TF, foi utilizado também o modelo de atribuição de pesos TFIDF, pois devido à existência de documentos com diferentes tamanhos na base de dados utilizada nos experimentos, alguns pesos poderiam ser atribuídos de forma tendenciosa pelo tamanho dos documentos se apenas for levada em consideração a freqüência da palavra no documento, ou seja, se uma palavra aparecer muitas vezes em um documento não indica que tem relevância para categorizar aquele documento, pois sua freqüência pode ser justificada pelo tamanho do texto.

2.2.4.4 Representação Vetorial

No modelo de espaço vetorial como foi mostrado anteriormente cada documento é representado por um vetor de termos, onde cada termo corresponde a uma característica (palavra) de relevância do dicionário presente no texto, associada ao seu peso no documento por um caractere separador.

As características no dicionário estão dispostas uma por linha, sendo identificadas no vetor pelo número da linha correspondente no dicionário. Cada vetor pode ou não possuir um rótulo de identificação como pertencente a uma categoria pré-definida, sendo necessário que todo vetor que represente um documento da coleção de treinamento esteja rotulado, pois com o rótulo os classificadores interpretam de quais categorias os exemplos de treinamento pertencem, e assim podendo classificar novos documentos. Como neste trabalho a categorização executada nos documentos acontece de forma dual, duas classes para representar os exemplos pertencentes e não pertencentes a cada categoria, os rótulos dos vetores assumem valor 1 para os documentos que pertencem à categoria e valor zero aos que não pertencem. Características do dicionário que não estejam presentes no documento não aparecem como componente no vetor, pois isto apenas aumentaria a dimensão do vetor de características do documento e não teria importância alguma no processo de categorização.

Pode-se visualizar na Figura 11 um exemplo de vetores de características gerados pela Ferramenta TMSK, com a representação de um documento por linha e caractere separador "@", seguindo a seguinte estrutura de representação:

Rótulo característica@peso característica@peso ..... característica@peso

Podemos interpretar a representação na Figura 11 da seguinte forma:

- A primeira linha corresponde ao primeiro documento com rótulo 1 indicando como pertencente à categoria em questão, a característica 2 do dicionário de palavras ocorre 3 vezes no texto e a característica 4 ocorre 6 vezes. Como as características 1 e 3 não têm ocorrência no documento, elas não aparecem nesta linha.

- Na segunda linha tem-se o segundo documento com rótulo 0 indicando como não pertencente à categoria em análise e com a palavra 1 do dicionário de palavras ocorrendo 3 vezes, as demais palavras do dicionário não aparecem neste documento.

- Como o rótulo da terceira linha é 1, o terceiro documento pertence à categoria em análise e tem a palavra 1 do dicionário de características aparecendo 2 vezes no documento, a palavra 3 do dicionário aparece 3 vezes e a palavra 4 aparece 2 vezes. As palavras do dicionário que não aparecem no documento não têm seus índices apresentados nos vetores.

```
1 2@3 4@6
0 1@3
1 1@2 3@3 4@2
```

Figura 11. Representação Vetorial dos Documentos.

Na Figura 12 é mostrado um exemplo de documento que pode ser representado pelo vetor da primeira linha na Figura 11, onde é pertence à categoria em análise (categoria alcohol) e as características 2 e 4 do dicionário são as palavras *alcohol* e *alcoholics*, que aparecem 3 e 6 vezes respectivamente nos documentos.

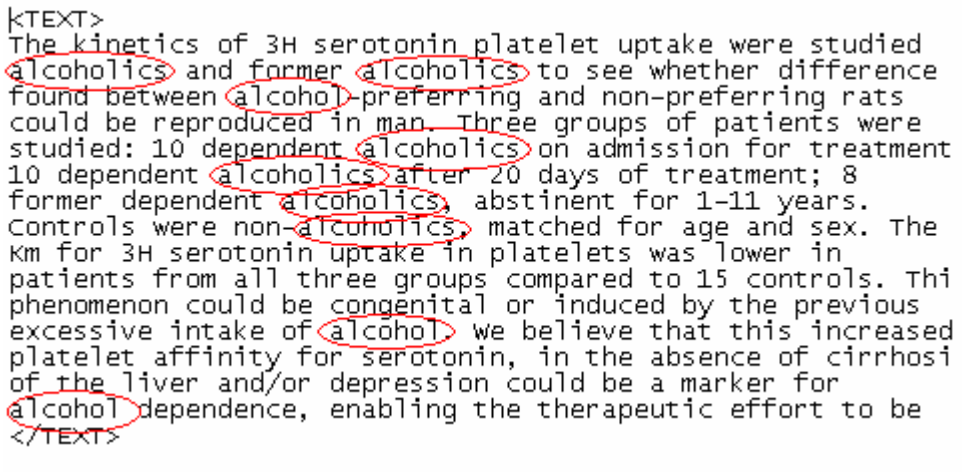

Figura 12. Exemplo de documento representado no vetor.

2.2.5 Extração de Conhecimento

Na fase de extração de conhecimento do texto está envolvida a aplicação de algoritmos que implementam os métodos de classificação. Estes métodos adquirem a indução da classificação automática de novos textos através da aprendizagem de máquina utilizando uma base de treinamento. Um conjunto de documentos previamente categorizados é usado para construir o classificador, sendo que uma parte dos documentos é utilizada para treinar o classificador e outra parte é utilizada para testar a precisão do classificador.

Abaixo estão apresentadas algumas categorias de métodos de classificação automática de textos bastante utilizados e algumas propriedades inerentes a cada método [18].

- **Árvores de Decisão:** Este método de classificação utiliza uma árvore com nós internos representando termos, ramos que partem dos termos representando testes nos pesos que um termo deve ter e nós-folha representando as categorias. A classificação ocorre percorrendo o caminho na árvore para o nó-folha apropriado, ou seja, levando a uma categoria.

- **Baseado em Exemplos:** Estes modelos baseiam-se nos julgamentos utilizados pelos especialistas para a classificação dos documentos da base de treinamento, este modelo não constrói um classificador à frente no tempo, mas usam as amostras rotuladas para classificar novas amostras. Um exemplo de classificador deste modelo é o k-vizinho mais próximo (KNN) [27], que se baseia em encontrar os k documentos mais similares ao novo documento a ser categorizado, podendo essa similaridade entre os documentos ser medida pela quantidade de palavras comum entre os documentos.

- **Probabilísticos:** Neste método cada exemplo de treino pode aumentar ou reduzir a probabilidade estimada que uma hipótese esteja correta. O conhecimento prévio, como uma probabilidade inicial dada a cada hipótese, pode ser combinado com os dados observados durante o treinamento para determinar a probabilidade dada a uma hipótese.

- **Regras de Decisão:** Neste método o classificador é construído pelo aprendizado de regras encontradas para separar completamente as classes. Ele consistirá de fórmulas condicionais para compor suas regras lógicas. A premissa da regra é composta de letras significando ausência ou presença de uma palavra de significância em um documento,

enquanto a cláusula denota que a classificação desse documento para uma determinada categoria leva a premissa a ser aceita por outros documentos.

- **Redes Neurais Artificiais:** As RNAs mais comuns para classificação são MLPs (*Multilayer perceptron*) e RBFNs (*Radial Basis Functions Networks*), que são do tipo feed-forward. Normalmente estas redes possuem três camadas. Os nodos da camada de entrada representam os termos e os nodos da camada de saída representam o conjunto de categorias. Os pesos são indicados pelas conexões dos nodos e representam uma relação de dependência condicional.

- ***Support Vector Machines*** (SVM): Trata-se de um método recente proposto por V. Vapnik [25][26] e usado por J. Thorsten para categorização de textos [23]. Este método divide o espaço de termos em hiperplanos ou superfícies de decisão, separando as amostras do treinamento em positivas e negativas. A superfície que provê maior separação entre as amostras é selecionada, o processo visa maximizar a margem entre superfície de decisão e os dados específicos de cada classe no conjunto de treinamento.

Os experimentos abordados neste trabalho utilizam o modelo de aprendizagem de máquina SVM para a categorização de documentos. Dessa forma o leitor pode encontrar no Capítulo 3 uma discussão mais detalhada desse modelo de categorização, como também uma aplicação do método SVM, produzida por J. Thorsten [23], para categorização de texto.

2.2.6 Avaliação de Conhecimento

A interação existente entre os elementos que participam do processo de categorização de texto permite detectar possíveis erros em alguma etapa do processo ou a possibilidade da aplicação de métodos que provejam melhores resultados na categorização. Por isso é bastante importante a atenção nas etapas de processamento das bases de textos, pois erros cometidos podem levar etapas a serem retomadas no processo de categorização.

Neste trabalho, o método de avaliação do conhecimento utiliza os documentos que foram agrupados no conjunto de teste para estimar a qualidade das predições executadas com o modelo de classificação gerado, ou seja, aprendido pela técnica de aprendizagem de máquina. A maioria das estimativas de qualidade das predições pode ser gerada a partir da comparação dos valores reais dos exemplos com os valores obtidos na predição, obtendo assim para uma classificação binária a análise dos números de classificações corretas e incorretas para cada classe. Na Tabela 2 é apresentado um modelo de classificação bi-valorada com as possibilidades existentes de classificação para essas duas classes. As possibilidades existentes são:

1. P_v- Positivo verdadeiro
2. P_f- Positivo falso
3. N_v- Negativo verdadeiro
4. N_f- Negativo falso

Tabela 2. Possibilidades de Classificação Bi Valorada.

Classificados como	**Predição como positiva**	**Predição como Negativa**
Exemplos Positivos	**P_v** - exemplos positivos classificados como positivos.	**P_f** - exemplos negativos classificados como positivos.
Exemplos negativos	**N_f** – exemplos positivos classificados como negativos	**N_v** - exemplos negativos classificados como negativos.

O desempenho pode ser estimado por diversas medidas, mas a medida padrão para a classificação é a taxa de erro da Equação (2.4) e o erro padrão da Equação (2.5).

$$\text{Taxa de Erro } (erate) = \frac{Numero\ De\ Erros}{Numero\ De\ Documentos} \tag{2.4}$$

$$\text{Erro Padrão } (SE) = \sqrt{\frac{erate * (1 - erate)}{Numero\ De\ Documentos}} \tag{2.5}$$

As medidas acima são utilizadas para estimar o desempenho das predições em geral, mas para a maioria das aplicações de texto, assim como categorização, é desejável uma análise mais detalhada dos erros [29]. Para aplicações de Recuperação de Informação existe um alto número de dados negativos, e um classificador pode conseguir uma acurácia muito elevada (Taxa de Erro muito baixa) simplesmente por classificar todos os dados como negativos. As medidas de precisão (*Precision*), cobertura (Recall) e acurácia (Accuracy) são interessantes medidas da qualidade de decisões binárias no problema de categorização de documentos. Suas definições podem ser vistas nas Equações (2.6) utilizando as informações apresentadas na Tabela 2.

$$\text{Precision} = \frac{Pv}{Pv + Pf}$$

$$\text{Recall} = \frac{Pv}{Pv + Nf} \tag{2.6}$$

$$\text{Accuracy} = \frac{Pv + Nv}{Pv + Pf + Nv + Nf}$$

O desempenho do classificador construído pode ser avaliado pelo cálculo das três medidas como segue:

- A porcentagem de todos os documentos pertencentes à classe em questão que conseguiram ser recuperados é a medida de cobertura (*recall*).

- A porcentagem dos documentos que foram corretamente rotulados como pertencentes à classe é a medida de precisão (*precision*).

- A porcentagem dos documentos que foram corretamente classificados corresponde à medida de acurácia (*accuracy*).

Existem situações em que o classificador construído apenas prediz em relação a uma classe e os valores para algumas medidas de desempenho tornam-se altos, embora não representem uma boa estimativa. O caso de aplicações de recuperação de informação tem geralmente muitos dados negativos, o que pode fazer com que um classificador consiga uma alta medida de acurácia simplesmente predizer todos os documentos como negativos. A medida *F-measure* que pode ser visualizada na Equação (2.7) é utilizada para contornar esse problema, sendo esta uma medida harmônica da precisão (*precicion*) e cobertura (recall). Esta medida é utilizada para medir a exatidão do classificador quando um único número é preferido [29], nos casos em que o classificador deixa de generalizar e passa a classificar toda entrada fornecida como pertencente à categoria.

$$\text{F-measure} = \frac{2}{(1/precision) - (1/recall)} \tag{2.7}$$

Como as coleções de documentos utilizadas para treinamento e teste dos classificadores são geralmente grandes, a precisão elevada é mais avaliada, para estes casos de elevada precisão as decisões positivas são geralmente corretas, podendo não funcionar para todas as decisões positivas, o que será medido pela medida de cobertura. Por exemplo, se um classificador identifica spam e-mail com alta precisão e baixa cobertura, isto pode levar a não identificação de todos spans, mesmo que quando haja a identificação dos *spans* seja de forma correta.

2.3 Representação dos Dados com TMSK

O *Text Miner Software Kit* (TMSK) [20] é um pacote de software para mineração de textos. Este pacote de software é composto por algumas rotinas para processamento e predição de textos baseados no formato XML (Figura 5). Estas rotinas podem ser de acesso do usuário final ou sub-rotinas acessadas internamente por outras rotinas para realização de tarefas. As rotinas encontram-se já compiladas na linguagem de programação Java, podendo ser acessadas pelo prompt de comandos. A Tabela 3 mostra as tarefas realizadas pela ferramenta TMSK separadas por categoria da tarefa.

As tarefas são executadas por diferentes rotinas, sendo que algumas dessas rotinas acessadas diretamente pelo usuário e outras são acessadas internamente por outras rotinas. Na

Tabela 4 são listadas as rotinas acessadas pelos usuários da ferramenta assim como as tarefas executadas pelas mesmas.

Tabela 3. Tarefas Realizadas pelo TMSK.

Categoria	Tarefa
Texto para Vetor	Tokenização
	Stemming
	Detecção de Final da Sentença
	Criação de Dicionário
	Geração de Vetor
Predição	Naive Bayes
	Modelos Lineares
Recuperação de Informação	Query Matcher
Algoritmo de agrupamento (não supervisionado)	K-means Clustering
Extração de Informação	Identificação de Entidades Nomeadas

Tabela 4. Rotinas Acessadas pelos Usuários no TMSK.

Categoria	Tarefa	Rotina TMSK
Texto para Vetor	Criação de Dicionário	*mkdict*
	Geração de Vetor	*vectorize*
Predição	Naive Bayes	*nbayes, testnbayes*
	Modelo Linear	*Linear, testline*
Recuperação de Informação	Query Matcher	*matcher*
Algoritmo de agrupamento (não supervisionado)	K-means Clustering	*kmeans*
Extração de Informação	Identificação de Entidades Nomeadas	*tagNames*

2.3.1 Rotinas TMSK

O pacote TMSK contém rotinas para processamento dos textos baseados no formato XML, assim como rotinas para predição. As rotinas para processamento dos textos podem ser utilizadas na preparação dos dados para as rotinas de predição da própria ferramenta, ou na utilização dos dados por outras ferramentas de classificação, podendo requerer algumas adaptações dos arquivos gerados para a interpretação correta por ferramentas de classificação.

Pode-se ver abaixo uma breve descrição das rotinas pertencentes à ferramenta TMSK, que foram utilizadas no trabalho, acessadas pelo usuário através da execução por linha de comando:

- *mkdict*: Rotina responsável por gerar um dicionário a partir de um conjunto de documentos. Pode ser usada para gerar um dicionário global ou um dicionário local, que é específico para uma categoria passada na chamada à execução da rotina por linha de comando. O tamanho do dicionário desejado pode também ser especificado na chamada à

execução da rotina. O conjunto de documentos no padrão XML que formará o dicionário de características, é fornecido como o parâmetro *infile* no arquivo de propriedades.

- *vectorize*: Esta rotina converte um conjunto de documentos em vetores baseando-se no dicionário gerado com os documentos. Estes vetores podem ser rotulados como pertencentes ou não a uma categoria. Os vetores gerados por essa ferramenta têm sempre como peso de cada termo a contagem da freqüência do termo no documento. A conversão para outros tipos de características é feita por outras rotinas que utilizam os vetores gerados como entrada.

- *Linear*: Esta rotina constrói um classificador linear binário a partir de um conjunto de vetores etiquetados obtidos pela vetorização dos documentos de treinamento. Há diversos parâmetros que o usuário pode ajustar para variar o classificador linear obtido, como o tipo de peso atribuído aos termos (*feature-type*) que pode ser binário, freqüência ou TFIDF. Há também parâmetros como *decision-threshold* que pode ser ajustado para controlar as medidas de cobertura (*recall*) e precisão (*precision*).

- *testline*: Esta rotina aplica o classificador binário gerado pela rotina *linear* para novos vetores e separa os documentos correspondentes aos vetores, como classificações positivas e negativas, ou seja, pertencentes ou não à categoria em questão.

Os vários parâmetros citados que ajustam o comportamento das rotinas são especificados no arquivo de propriedades da ferramenta *tmsk.properties*.

2.3.2 Especificação dos Dados de Entrada

O TMSK analisa os documentos no formato XML e faz algumas considerações dos documentos dados como entrada:

- Cada documento é distinguido individualmente dos demais na coleção por uma tag especificada pelo parâmetro *doctag* no arquivo de propriedades da ferramenta. Por exemplo, nos documentos utilizados neste trabalho as tags <DOC></DOC> distinguem o começo e fim de um documento da coleção.

- Apenas seções do texto no documento é que são de interesse para a categorização do documento. O usuário especifica as tags das seções que interessam do texto através do parâmetro *bodytags* no arquivo de propriedades da ferramenta. Nos documentos utilizados neste trabalho as seções que apresentam relevância para a categorização são marcadas pelas tags <TEXT> e <TITLE> e são configuradas da seguinte forma no arquivo de propriedades:

  bodytags=TEXT TITLE

- O usuário especifica a tag usada para identificar as etiquetas nos documentos. Por exemplo, as etiquetas dos documentos deste trabalho são determinadas pela tag <TOPICS> e como um documento pode pertencer a mais de uma categoria, pode haver tags internas <TOPIC > que determina cada categoria individualmente.

Como o formato dos documentos da base de dados OHSUMED [11], a qual foi utilizada neste trabalho, não correspondia ao formato XML utilizado pelas rotinas do TMSK, foi necessária a implementação de um programa que executasse uma conversão do formato dos documentos para um padrão XML com estruturas que atendessem as necessidades da aplicação.Assim como foi mencionado na seção 2.2.2.

2.3.3 Gerando Dados

Algumas rotinas do sistema TMSK após executarem sua funcionalidade, geram arquivos que serão utilizados como entradas por outras rotinas do próprio sistema. Estes arquivos podem também ser editados manualmente, dependendo da aplicação dos dados pelo usuário.

Dicionário de Palavras

A rotina *mkdict* gera dicionários a partir de documentos XML com um tamanho determinado pelo usuário e como foi visto na seção 2.2.3.1, se o nome da categoria é especificado na chamada à rotina, o dicionário gerado será local e caso contrário será gerado um dicionário global para qualquer categoria. Um parâmetro importante passado para essa rotina através do arquivo de propriedades é a freqüência mínima (*minimum-frequency*) de aparição de uma palavra no texto para que a mesma seja colocada no dicionário.

Pode-se verificar abaixo a sintaxe da chamada à rotina de geração de dicionários, tanto global como local:

>> java mkdict size dictionaryfile (dicionário global)
>> java mkdict size category dictionaryfile (dicionário Local)

Na geração do dicionário local as palavras apresentadas são aquelas de relevância nos documentos pertencentes à categoria analisada, e no dicionário global as palavras que são apresentadas são aquelas de relevância em todos os documentos da coleção de dados.

A lista de *stopwords* é usada para filtrar o dicionário de palavras. Estas palavras da lista são removidas do dicionário após o processo de geração do dicionário. A remoção destas palavras diminui o tamanho do dicionário e aumenta a precisão na escolha de palavras com relevância nos documentos.

A lista de *stopwords* fornecida na ferramenta TMSK através do arquivo stop.wds é relativamente pequena para a língua inglesa, por isso foram adicionadas manualmente mais palavras que não apresentavam relevância no processo de categorização, as palavras adicionadas tiveram como base a observação dos dicionários gerados com a lista de *stopwords* da ferramenta.

Criação de Vetores

O TMSK processa documentos em um formato conhecido por vetor esparso. Este formato é alcançado convertendo primeiro os documentos para um formato de tabela, onde cada linha corresponde a um documento e cada coluna indica uma palavra do dicionário gerado. Cada célula na tabela representa individualmente o número de vezes que a palavra aparece no documento. A partir da representação por tabela parte-se para a representação do vetor escasso, onde células que contenham valor zero não são representadas, pois não influenciam na categorização.

Na Figura 13 é mostrado um exemplo de uma representação dos dados por vetor escasso a partir de uma representação por tabela. Na primeira linha da tabela, por exemplo, o primeiro e o

terceiro elemento têm número de aparição no documento igual a zero. Então não precisam ser representados os índices 1 e 3 na primeira linha do vetor, mas como o segundo e o quarto elemento da primeira linha possuem quantidade de aparições no documento diferente de zero, então serão representados com suas respectivas quantidades 15 e 3 na representação por vetor escasso. Note a representação da figura 13 (b).

Tabela

0	15	0	3
12	0	0	0
8	0	5	2

(a)

Vetor Escasso

(2,15) (4,3)
(1,12)
(1,8) (3,5) (4,2)

(b)

Figura 13. Exemplo de conversão do Formato de tabela (a) para Vetor Escasso (b).

A rotina *vectorize* pode gerar vetores rotulados ou não a partir dos documentos. Porém, para o treinamento dos classificadores automáticos de texto é necessário que os documentos contenham rótulos para que estas apareçam como etiquetas no inicio de cada representação do documento como um vetor.
O TMSK possui em seu arquivo de propriedades um parâmetro chamado *feature-type* que é responsável por definir o modelo de atribuição dos pesos no vetor, mas esse não é um parâmetro para a rotina *vectorize*, a qual sempre gera vetores com atribuição de pesos baseada no contador de freqüência de cada termo (Figura 13). Como vimos na seção 2.2.4.3, o modelo de atribuição de pesos TFIDF foi utilizado neste trabalho para evitar uma atribuição tendenciosa dos pesos. Porém, como os vetores gerados pelo TMSK utilizam a quantidade da freqüência das palavras no texto, foi necessária a implementação de uma classe que tratasse os pesos de cada termo no vetor gerado pelo TMSK, transformando-os no modelo TFIDF. A classe implementada na linguagem Java pode ser vista no *Apêndice B*.

Com os pesos atribuídos a cada termo do vetor seguindo o modelo TFIDF, pode-se passar a utilizar o vetor gerado com outras ferramentas de classificação que apresentem técnicas de aprendizado diferente.

No próximo capítulo serão descritas as técnicas de classificação de texto utilizadas no trabalho, assim como algumas implementações de classificadores automáticos baseados nas técnicas descritas.

2.3.4 Arquivo de Propriedades

As rotinas presentes no pacote de software TMSK possuem alguns parâmetros em comum, como também opções que não são alteradas frequentemente nas rotinas. Todos esses parâmetros e opções das rotinas são declarados no arquivo de propriedades *tmsk.properties*. Este arquivo permite que as rotinas, se comuniquem e compartilhem informações sobre as estruturas em análise. Um pequeno trecho de um simples arquivo de propriedades é visto na Figura 14, com os comentários começando pelo caractere “#”.

```
# this tag identities individual documents. case sensitive.
doctag=DOC

# these are the tags for the text to be used. case sensitive.
bodytags=TEXT TITLE

# labeltag is the tag for the categories.
# for mkdict/vectorize, the actual category name is specified on
commandline.
labeltag=TOPICS

# input can be a zip file or an xml file. the zip file can contain
# xml files. there can be nested zip files (as long as eventually there
# are xml files).
infile=entradaTotal.xml

# input dictionary for vectorize, etc.
dictionary=dict.dt

# stopwords to ignore for dictionary creation.
# words are one per line. case-insensitive.
stopwords=stop.wds
```

Figura 14. Arquivo de Propriedades TMSK.

O arquivo de propriedades do sistema TMSK possui muito mais parâmetros do que os vistos na Figura 14, mas para efeito prático do trabalho apenas foram utilizados os parâmetros descritos na Tabela 5.

Tabela 5. Parâmetros TMSK utilizados no Trabalho.

Nome do parâmetro	Breve descrição	Exemplo no trabalho
word-delimiters	Caracteres separadores das palavras.	*word-delimiters*=" \n\t\r,.;:!?()<>[]+\"\'-"
doctag	Tag XML para identificação dos documentos individualmente.	*doctag*=DOC
bodytags	Tag XML para identificar as partes a serem analisadas nos documentos.	*bodytags*=TEXT TITLE
labeltag	Tag XML que identifica a categoria dos documentos.	*labeltag*=TOPICS
infile	Nome do arquivo de entrada no formato XML.	*infile*=rabbit.xml
dictionary	Nome do arquivo de dicionário gerado e utilizado para vetorização.	*dictionary*=dictRabbit.dt
stopwords	Nome do arquivo de *stopwords* utilizado na geração do dicionário.	*stopwords*=stop.wds
vectorfile	Nome do arquivo de vetor escasso gerado, que será utilizado pelos classificadores.	*vectorfile*=treinoRabbit.vec
minimum-frequency	Freqüência mínima das palavras no arquivo para que componha o dicionário.	*minimum-frequency*=3
feature-type	Tipo da atribuição de peso, usada internamente pela rotina *linear*.	*feature-type*=tfidf
lambda	Controla o tamanho do espaço de busca na classificação linear.	*Lambda*=0.001

learning-rate	Taxa de aprendizado do algoritmo de treinamento.	*Learning-rate*=0.25
linear-iterations	Controla o número de iterações no treinamento.	*Linear-iterarions*=400
decision-threshold	Controla a negociação entre as taxas de precisão e cobertura.	*Decision-threshold*=0.1

Capítulo 3

Técnicas de Classificação de Texto

Neste capítulo são apresentados conceitos referentes aos métodos de classificação automática de texto, como também são abordados os parâmetros e os classificadores utilizados.

3.1 Aprendizado de Máquina

A construção de máquinas capazes de aprender a partir de experiências passadas tem gerado grande discussão técnica e filosófica. O aspecto técnico dessa discussão demonstra que máquinas podem apresentar significante nível de habilidade no aprendizado, embora os limites para essa habilidade ainda estejam longe de ser definidos [3].

Existem muitos problemas que não podem ser resolvidos por técnicas de programação convencional. Assim por exemplo, não se sabe como escrever um programa convencional de computador que classifique um texto na categoria a qual ele pertence, ou mesmo reconheça um caractere escrito à mão. A construção do aprendizado humano nesses casos mencionados, para reconhecer uma letra ou um texto de uma categoria, é feita através da apresentação prévia de elementos (caracteres e textos) individuais para posterior reconhecimento. Baseando-se nessa forma de resolução do problema por máquinas constrói-se a metodologia de aprendizado.

Em aprendizagem de máquina, problemas são resolvidos usando uma estratégia de fazer com que o computador tente construir uma função que mapeie os pares entrada/saída dos exemplos fornecidos. Este caso particular da metodologia de aprendizado, que usa os pares entrada/saída no aprendizado, é chamado de aprendizado supervisionado. Existem outros tipos de aprendizado como o não-supervisionado, onde a saída desejada para cada entrada não é fornecida. Neste trabalho apenas foi utilizado técnica de aprendizagem de máquina supervisionada.

Os pares de entrada/saída refletem o relacionamento funcional que mapeia as entradas às saídas, embora às vezes as saídas sejam corrompidas por ruídos. No caso da classificação esta função de mapeamento é referenciada como *função de decisão*.

O responsável pela aquisição do conhecimento através dos exemplos fornecidos é o algoritmo de aprendizado. A saída dada pelo algoritmo de aprendizagem é tida como a solução do problema de aprendizagem. Um problema de aprendizado com saídas binárias é referenciado

como um problema de *classificação binária* (duas classes), enquanto para um valor real de saídas o problema torna-se conhecido como uma *regressão*. Outro tipo de variação nos modelos de aprendizado é o caminho pelo qual os dados de treinamento são gerados e como são apresentados ao classificador, como exemplo pode-se citar o aprendizado em *batch*, no qual todos os dados são apresentados ao classificador no início da aprendizagem, e o aprendizado *on-line* no qual o classificador recebe um exemplo por vez.

Neste trabalho a categorização de documentos segue o método de aprendizado supervisionado, onde cada documento, fornecido em *batch*, é rotulado com a categoria à qual pertence. Através da comparação entre as saídas desejadas e as fornecidas pelo algoritmo de treinamento é possível avaliar o desempenho do modelo de classificação.

3.2 *Overfitting*

Geralmente um algoritmo de aprendizagem é treinado usando um conjunto de dados para treinamento, e através das saídas conhecidas desses dados, o conhecimento do algoritmo é construído e deve ser usado para novos dados apresentados. O ideal é que o algoritmo construído tenha a capacidade de *generalizar* o conhecimento adquirido para novas situações que não foram apresentadas durante o treinamento. No entanto, um dos principais problemas nos algoritmos de aprendizagem de máquina é a especialização ou *overfitting* de suas regras, onde as regras do algoritmo não generalizam bem e passam a decorar situações dadas como entradas. Pos isso, as regras geradas pelos algoritmos devem ter a forma mais geral possível.

Uma técnica bastante aplicada pelos algoritmos de aprendizagem na tentativa de se evitar *overfitting* e de averiguar a robustez dos resultados gerados, é a validação cruzada, que consiste na divisão do conjunto total de padrões em N grupos com tamanhos aproximadamente iguais, assim realizando o treinamento N vezes, sendo a cada treinamento um dos grupos deixado para teste e os outros N-1 para treinamento [6], dessa forma aumentando o número de situações diferentes apresentadas ao classificador. Com os erros de classificação obtidos nesses N treinamentos, é realizado o cálculo da média dos erros na classificação. Uma outra técnica, derivada da validação cruzada, é o calculo da taxa *Leave-One-Out*, que funciona testando cada padrão um a um com o treinamento dos demais que formam a base de dados. Ao término do treinamento, todos os padrões passaram pelas fases de treinamento e teste, gerando erros que servirão para se calcular a média de erros. Como as ferramentas utilizadas no trabalho já aplicam a técnica de validação *Leave-One-Out* no treinamento dos dados, e esta se apresenta gerando um número maior de treinamentos com os padrões de entrada, não foi utilizada a técnica de validação cruzada.

3.3 Classificação Linear

Costuma-se referenciar o aprendizado de máquina que utiliza hipóteses na formação de combinações lineares das variáveis de entrada, como aprendizado linear de máquina. Na sua forma básica os classificadores SVM aprendem regras binárias lineares, que constroem hiperplanos ótimos para maximizar a margem de separação das classes [3], como mostrado na Figura 15. As regras aprendidas são descritas por uma função linear h(x), que utiliza um vetor de pesos W e uma diagonal inicial b, como segue na Equação (3.1), para mapear as entradas no espaço de hipóteses da função. Dependendo de qual lado do hiperplano o vetor de atributos X incidir, será classificado nas classes +1 ou -1 .

$$h(x) = sign(W * X + b) = \begin{cases} +1, se\ W * X + b > 0 \\ -1, caso\ contrário \end{cases} \quad (3.1)$$

A interpretação geométrica destas hipóteses propostas na equação acima, é que o espaço de entrada X é dividido em duas partes pelo hiperplano definido na equação $(W * X) + b = 0$, como é mostrado na Figura 15. Um hiperplano divide o espaço em dois subespaços de dimensão *n*-1, os quais correspondem a duas diferentes classes. Na Figura 15, o hiperplano é definido pela linha preta com o vetor de pesos perpendicular a ele, enquanto o valor de *b* move o hiperplano numa direção paralela a sua.

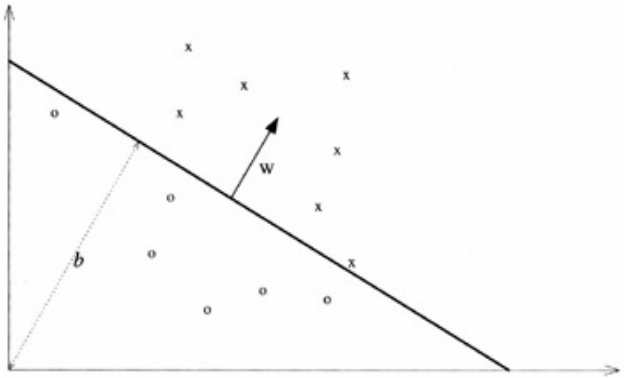

Figura 15. Hiperplano de separação (W, b) no treinamento bidimensional.

O problema da classificação dual pode ser resolvido pela definição de um vetor de pesos W_i e da diagonal inicial b_i para cada classe, onde cada vez que um novo padrão for fornecido para a classificação, ambas as medidas são ajustadas e o ponto X é atribuído à classe 1 se $W_1 X + b_1 \geq W_{-1} X + b_{-1}$, e para a classe -1 em outros casos. Mas para problemas de classificação multi-classes, a generalização do aprendizado de máquina para as *m*-classes é direto, ou seja, para cada classe são associados um vetor de pesos W_i e uma tupla (W_i, b_i) [3]. Neste trabalho utiliza-se apenas a classificação dual, com documentos sendo categorizados como pertencentes, ou não a cada categoria.

No geral, problemas com complexidade maior requerem um espaço de hipóteses maior do que o espaço construído por funções lineares. Representações por kernels oferecem uma solução por projetar os dados em um espaço de características de alta dimensão. O problema da escolha de uma arquitetura adequada nas aplicações com redes neurais, aqui é trocado pela escolha de um kernel mais adequado para a aplicação.

3.3.1 Classificador Linear no TMSK

A classificação de documentos através da rotina *Linear* do sistema TMSK, baseia-se no método linear de classificação, que já são clássicos pelo seu poder de predição. Matematicamente este método é uma função linear de marcação, a equação geral desta função pode ser vista na Equação 3.2. Nesta Equação, *D* representa um documento e W_j é o peso para a *j*-ésima palavra do dicionário, *b* é uma constante, e X_j é um ou zero, dependendo da presença ou ausência da j-ésima palavra no documento. Geometricamente a rotina produz uma linha ou hyperplano de separação, embora com uma linha não se possam cobrir superfícies complexas de formadas pelos padrões de entrada, sendo talvez necessário métodos capazes de executarem a separação com curvas.

$$\text{marcação(D)} = \sum_j W_j X_j + b \qquad (3.2)$$

Cada documento é rotulado como pertencente ou não à categoria em análise de acordo com a aplicação da equação 3.2, e através dos pesos das palavras pertencentes ao dicionário da categoria, pesos estes que são aprendidos de acordo com uma aplicação numérica de análise do documento (*Weighting*).

Uma vantagem nesse método de marcação linear, é que ele trabalha eficientemente com a escassez de dados nas representações vetoriais. Isto é bastante importante para as aplicações de mineração de texto. Um dos maiores avanços nos métodos lineares para texto é a habilidade de trabalhar com dicionários enormes e encontrar pesos para todas as palavras do dicionário.

A fim de se manusear grande quantidade de dados, é necessário utilizar um algoritmo que possa tirar benefícios da estrutura de representação dos documentos por vetores. O algoritmo de aprendizagem utilizado pelo classificador linear do sistema TMSK pode ser visto na Figura 16. Neste algoritmo η_1 é a taxa de aprendizagem, n é o número de características do vetor, K representa o número de iterações, para estabelecer o critério de parada, λ é um parâmetro que controla o tamanho do espaço de pesos, c é igual a $1/\lambda n$, (x^n, y^n) representa o n-ésimo vetor do documento com sua respectiva saída y^n, e w_j representa o peso do j-ésimo termo no documento. Um valor alto para o parâmetro λ significa procurar num espaço menor de pesos, e isto dificulta o ajuste dos dados no treinamento. Por isso para um bom desempenho do algoritmo é necessário escolher o parâmetro λ apropriadamente. Na rotina de classificação linear da ferramenta TMSK, o parâmetro é ajustado para o valor $\lambda=10^{-3}$, pois este valor geralmente é bom para os conjuntos de dados [29].

Input: training data (x^1, y^1)..... (x^n, y^n)
Parameters: K, c, $\eta_{1,...,}\eta_n$
Output: weight vector w_j (j = 1... m), b
Initialize: $\alpha_I = 0$ (i = 1... n): $w_j = 0$ (j = 1 ...m): $b = 0$
for k = 1 to K **do**
  for i = 1 to n **do**
    $p = (w \cdot x_i + b)y_i$
    $d_i = \max(\min(2c - \alpha_i, \eta_i((c - \alpha_i)/c - p)). - \alpha_i)$
    $w = w + d_i x^i y^i$
    $b = b + d_i y^i$
    $\alpha_i = \alpha_i + d_i$
  endfor
endfor

Figura 16. Algoritmo de Aprendizagem TMSK.

3.4 Redes Neurais Artificiais (RNAs)

As Redes Neurais Artificiais (RNAs) ressurgiram no final da década de 80 e trata-se de uma forma computacional não algorítmica, que apresenta um modelo matemático inspirado na estrutura neural do cérebro humano[1]. RNAs são constituídas por unidades de processamento chamadas *nodos* que simulam os neurônios biológicos, estes nodos são distribuídos em uma ou mais camadas e interligados por conexões, das quais são associados pesos para armazenar o

conhecimento do modelo. O comportamento inteligente de uma Rede Neural Artificial é inerente às interações entre as unidades de processamentos da rede e não a complexidade de cada unidade de processamento.

A solução de problemas através da utilização de RNAs tem sido bastante atrativa , pois devido ao paralelismo natural inerente à arquitetura existe a possibilidade de um desempenho superior aos modelos convencionais. As RNAs possuem alta capacidade de aprendizado e generalização, sendo capazes de extrair informações não apresentadas de forma explícita através dos exemplos e de realizar auto-organização.

3.4.1 Inspiração na Biologia

O sistema nervoso é formado por um conjunto extremamente complexo de células, os neurônios, que têm a função de determinar o comportamento e funcionamento do raciocínio humano. Estas unidades constituintes do sistema nervoso são formadas pelos dendritos, que são um conjunto de terminais de entrada, pelo corpo central, e pelos axônios que são longos terminais de saída (Figura 17). A sinapse é a região onde dois neurônios entram em contato e através da qual os impulsos nervosos são transmitidos entre estes dois neurônios. Quando os impulsos são recebidos por um neurônio, estes são processados e ao atingir certo limiar de excitação o neurônio dispara substância neurotransmissora, que pode inibir ou excitar a geração de pulsos pelo outro neurônio constituinte da sinapse [16].

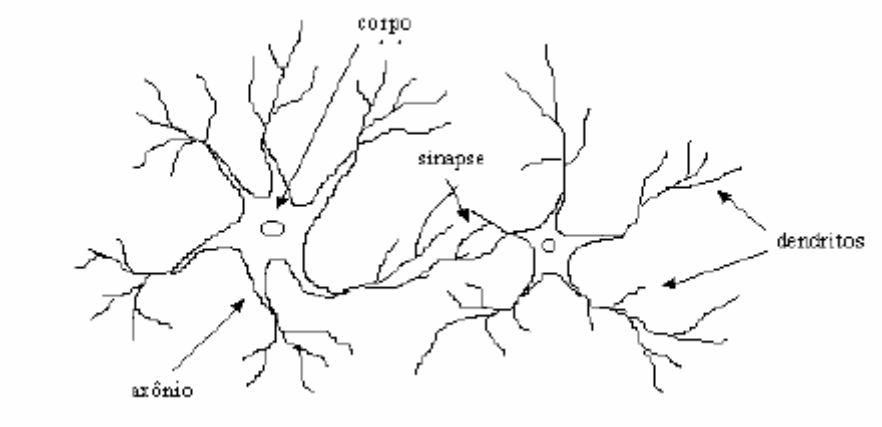

Figura 17. Estruturas de um Neurônio Biológico.

As RNAs tentam reproduzir o comportamento básico do neurônio e sua dinâmica, possuindo assim algumas características comuns entre o sistema biológico e artificial, como :

- Baseados em unidades de computação paralela e distribuída.
- Redundância e modularização das conexões.
- Comunicação das unidades formadoras por meio de conexões sinápticas.
- Possuem detectores de características.

3.4.2 Características Gerais

A estrutura da unidade de processamento das redes neurais foi proposta por McCulloch e Pitts. Sendo modelada como uma estrutura formada por terminais de entrada, X_1, X_2,...., X_p, que funcionam como os dendritos nas unidades biológicas, e um terminal de saída Y que emula as

sinapses .As entradas estão associadas a pesos W_1, W_2, ..., W_p, que indicam a influência de cada entrada na saída da unidade de processamento.Como foi visto na seção anterior o neurônio biológico executa um disparo ao ser atingido certo limiar de excitação, já com o modelo de neurônio artificial proposto por McCulloch e Pitts, neurônio MCP, a ativação é obtida através de uma função de ativação, que por meio da soma ponderada dos sinais ativa ou não a saída.Na descrição original do modelo MCP, a função de ativação das unidades de processamento é dada por uma função de limiar , descrita na Equação (3.3), que pode associar à saída os valores 1 ou 0 que corresponderão respectivamente à ativação ou não do neurônio.

$$\sum_{i=1}^{p} X_p W_p \geq \theta \qquad (3.3)$$

Onde θ é o limiar (*threshold*) de ativação do neurônio.

A Figura 18 representa o modelo de neurônio MCP com f(a) $=W_1X_1+W_2X_2+ \ldots .. +W_pX_p$, e a saída Y sendo dada por :

$$Y=\begin{cases}1, se\ f(a) \geq \theta \\ 0, se\ f(a) < \theta\end{cases}$$

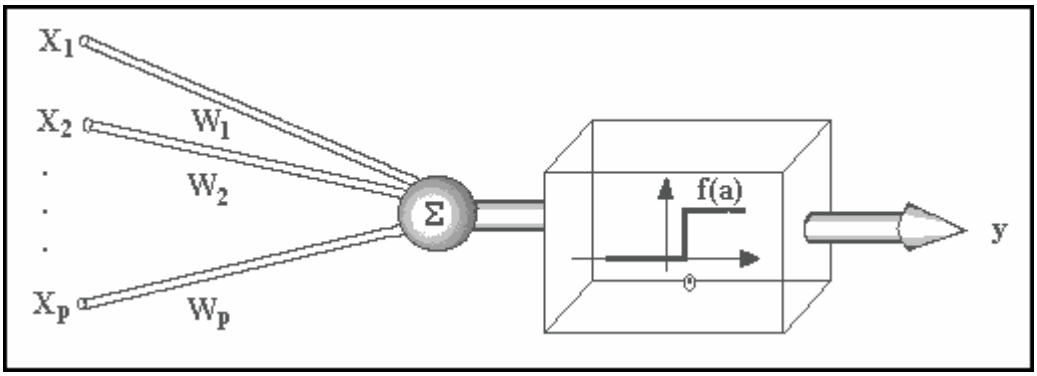

Figura 18. Modelo de Neurônio MCP.

A partir do modelo proposto por McCulloch e Pitts foram derivados vários outros modelos com diferentes funções de ativação, não permitindo apenas saídas com valores 0 ou 1. Um exemplo de função de ativação deste tipo é a função rampa, a qual é um caso particular da função linear, que permite saídas diferentes de 0 e 1 em um intervalo de domínio [-z, +z] da função (Figura 19).

$$f(x) = \begin{cases}1, se\ x \geq 1/2 \\ x, se -1/2 < x < 1/2 \\ 0, se\ x < 1/2\end{cases}$$

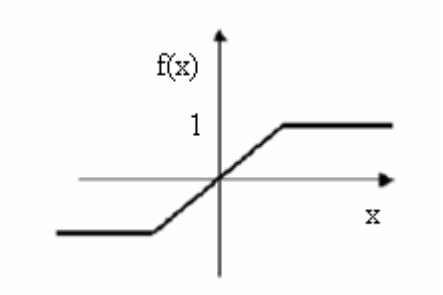

Figura 19. Função Rampa.

Além da função de ativação, outro parâmetro importante na configuração das RNAs é sua arquitetura, pois esta restringe o tipo de problema tratado pela rede [1]. Redes com uma camada única de nodos do modelo MCP, por exemplo, só resolvem problemas linearmente separáveis. Alguns parâmetros são utilizados para a definição da arquitetura, como: número de camadas da rede, número de neurônios em cada camada, tipo de conexão entre os nodos e topologia da rede. Na Figura 20 pode-se ver o exemplo de uma rede neural com uma arquitetura de múltiplas camadas, não apresentando ciclos, ou seja, nenhum neurônio de uma camada *i* fornece saída como entrada para um neurônio de uma camada de índice menor ou igual a *i*.

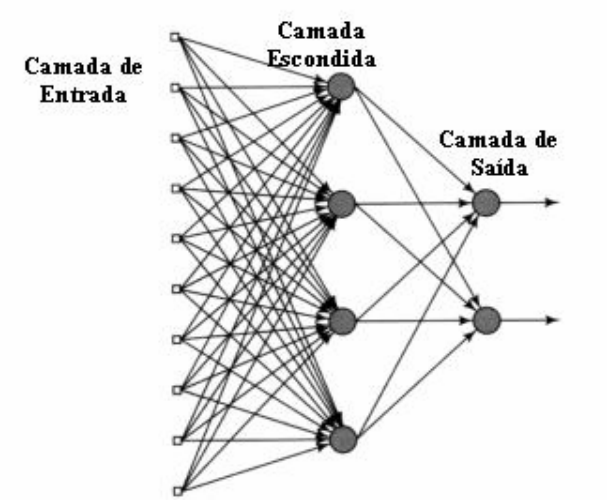

Figura 20. Arquitetura de múltiplas camadas e acíclica.

3.5 Máquinas de Vetor de Suporte – Support Vector Machines (SVM)

Máquina de vetor de suporte é uma técnica recente de classificação e regressão desenvolvida por Vapnik, baseada no princípio da minimização de risco estrutural (*Structural Risk Minimization*-SRM) [26], que será abordada na seção seguinte. A técnica SVM deriva da técnica de aprendizado por RNA, na sua forma computacional não algorítmica para obtenção do conhecimento. No entanto, a técnica SVM utiliza o espaço de hipóteses das funções lineares em um espaço de características de alta dimensão, treinada com um algoritmo de aprendizagem que deriva da teoria de aprendizado estatístico. SVM tem conseguido uma notável precisão para problemas importantes [3][28], aplicações que variam desde identificação de partículas, categorização de documentos, bioinformática e banco de dados de marketing [2].

No método SVM, uma variável de predição é denominada atributo, e este atributo quando empregado na construção de hiperplanos é chamado de característica. O conjunto de

características selecionadas para descrever um documento na classificação é chamado de vetor, e os vetores que se encontram próximos dos hiperplanos construídos para separar as categorias são chamados de *vetores de suporte* [19], como pode ser visto na Figura 21.

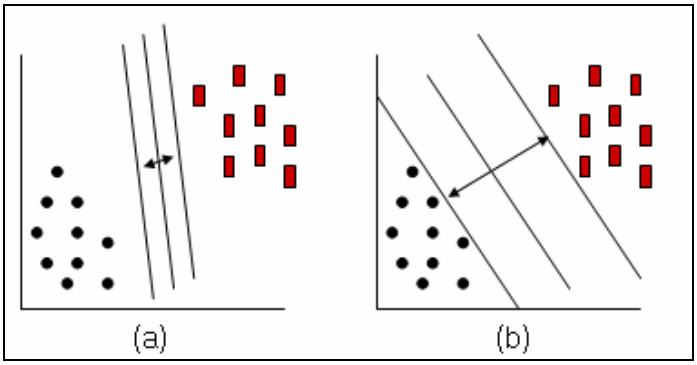

Figura 21. Hiperplano com margem de separação estreita (a) e margem de separação larga (b).

O método SVM é utilizado para reconhecimento de padrões sobre o espaço vetorial, a idéia deste método consiste em construir uma superfície de decisão (hiperplano) que melhor separe os padrões de treinamento das diferentes classes, ou seja, formar o plano que maximize a margem de separação das categorias para o processo de categorização de documentos. Na Figura 21 (a) pode ser visto um hiperplano com uma margem de separação estreita, e na Figura 21 (b) o hiperplano com uma margem de separação mais larga, que deverá ocasionar uma maior generalização para os padrões dados como entrada no processo de classificação.

3.5.1 Minimização do Risco Estrutural (SRM)

O método de classificação SVM é baseado no princípio da minimização do risco estrutural (*Structural Risk Minimization*-SRM), o qual indica que um algoritmo de aprendizagem de máquina deve tentar diminuir o risco estrutural em vez de diminuir o risco empírico para obter bom desempenho na generalização [3][28]. O risco empírico é o erro no conjunto de treinamento, enquanto o risco estrutural considera tanto o erro no conjunto de treinamento quanto a complexidade da classe de funções usadas para ajustar os dados.

Na categorização de textos existe um conjunto de n observações, onde cada observação corresponde a um vetor X_i e sua verdade associada Y_i. O vetor X_i, que representa cada documento da coleção, é composto de palavras e seus respectivos pesos. O rótulo Y_i, que é associado a cada documento, recebe valor 1 para exemplos pertencentes à categoria em análise, e -1 para exemplos não pertencentes à categoria em questão. Na Figura 22 abaixo, pode ser visto um exemplo de n observações, tendo para primeira observação um documento representado pelo vetor X_1 e com uma verdade associada Y_1 igual a um.

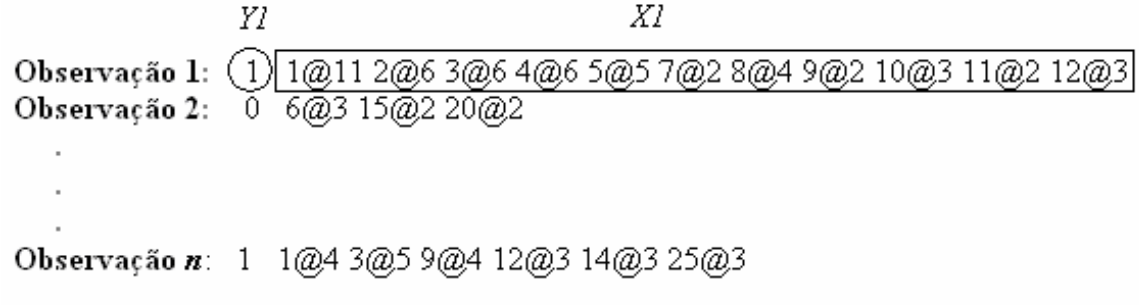

Figura 22. Observações Vetoriais.

Supõe-se que exista uma máquina cuja tarefa é aprender o mapeamento $X_i \rightarrow Y_i$, esta máquina adquire o aprendizado através de um conjunto de possíveis mapeamentos $X_i \rightarrow f(X, \alpha)$. Admitindo-se ser uma máquina determinística para cada X dado e um α escolhido, a máquina sempre produzirá como resultado $f(X, \alpha)$. A expectativa de erro para o teste da máquina treinada é dada pela Equação (3.4).

$$R(\alpha) = \int \frac{1}{2} | Y - f(X, \alpha) | dP(X_i, \alpha) \tag{3.4}$$

O risco empírico, já abordado nesta seção, é o erro medido sobre o conjunto fixo e finito de treinamento, como definido na Equação (3.5). Pode-se ver que o valor para R_{emp} (α) é fixo para um α escolhido e um conjunto de treinamento específico. Nesta Equação (3.5) a quantidade $\frac{1}{2} | Yi - f(Xi, a) |$ é chamada de perdas, podendo assumir valores 0 e 1.

$$R_{emp}(\alpha) = \frac{1}{2n} \sum_{i=1}^{n} | Yi - f(Xi, \alpha) | \tag{3.5}$$

Segundo V. Vapnik [26] SRM consiste em encontrar o subconjunto de funções que minimize o limite da expectativa de erro. E se escolhido o valor n para as perdas, sendo $0<n<1$ e com probabilidade $1-n$, obtém-se a relação mostrada na Equação (3.6). Onde h é um inteiro não negativo chamado *Vapnik Chervonenkis Dimension*, que representa medida de capacidade.

$$R(\alpha) \leq \mathrm{Re}\, mp(\alpha) + \sqrt{\left(\frac{h(\log(2l/h)+1) - \log(n/4)}{l} \right)} \tag{3.6}$$

SRM pode eficazmente evitar o problema de *overfitting* com o mecanismo de maximização da margem do plano de separação construído e a seleção apropriada do parâmetro de penalidade C. Pois este é um parâmetro de regularização, que é adicionado ao erro no conjunto de treinamento para que seja executada a negociação entre a taxa de precisão e cobertura, tentando assim impedir valores bastante altos da medida de precisão por motivo de especificação do classificador. Um parâmetro C elevado corresponde a atribuir uma penalidade mais elevada aos erros da classificação. A solução de um hiperplano de separação ideal é obtida através de uma negociação entre a maior margem de separação obtida e o menor número de erros, sendo controlados pelo parâmetro C.

3.5.2 SVM com kernel

O método SVM na categorização mapeia os vetores de treinamento X_i em um espaço de dimensão elevada, podendo ser infinito, através da função φ. Então o método encontra um plano de separação linear com a máxima margem de separação neste espaço com muitas dimensões. Um kernel (x, y) é um produto interno em algum espaço de características de alta dimensionalidade, $K(x, y) = \varphi^T(x)\, \varphi(y)$. Existem vários kernels propostos na literatura [3][4][5]. Neste trabalho, são utilizadas a função de base radial (RBF) e a função polinomial como kernels.

Em um kernel RBF o limite de decisão no espaço de entradas é dado pela função $K(X_i, X_j) = \exp(-\gamma \| X_i - X_j \|^2)$, onde $\| X_i - X_j \|$ é a distância euclidiana entre X_i e X_j. SVMs com kernels RBF possuem dois parâmetros, o primeiro é conhecido como parâmetro de penalidade de

erro do termo e identificado pela letra *C*, o qual negocia a taxa de precisão e cobertura na classificação, o outro parâmetro é a largura dos kernels RBF, identificado por γ.

No kernel Polinomial o limite de decisão no espaço de entrada é definido por uma curva polinomial de grau *d*, representada pela função $K(Xi, Xj) = (Xi^T \ X_j + 1)^d$. Este tipo de kernel possui dois parâmetros, o primeiro é o parâmetro C, utilizado também no kernel RBF e classificação linear, e o segundo é o parâmetro *d* que representa o expoente da função de decisão.

Esses parâmetros utilizados nos kernels possuem grande influência na classificação e generalização em SVMs. Os valores destes parâmetros podem ser selecionados para efeito de verificação de suas influências na categorização dos documentos, como pode ser observado nos experimentos deste trabalho.

3.5.3 SVM na Categorização de Textos

A categorização de textos consiste na associação dos documentos a categorias pré-definidas, e para facilitar o aprendizado cada categoria é tratada como um problema separado, admitindo-se uma classificação binária [24].

Segundo J. Thorsten [23] o bom desempenho da técnica SVM na categorização de textos é justificado por algumas propriedades dos documentos, como as citadas abaixo:

- Espaço dimensional de entrada bastante grande: ao se trabalhar com textos nos classificadores, é notável a grande quantidade de características presente. SVM utiliza *overfitting protection,* não dependendo da quantidade de características. Os classificadores textuais baseados em SVM têm o poder de controlar este grande número de características presentes nos textos.

- Poucas características irrelevantes: testes realizados com o conjunto da base de dados *Reuters* por J. Thorsten, demonstram que os classificadores que utilizam grande parte da informação extraída do texto obtêm desempenho melhor do que os classificadores que refinam bastante as informações extraídas.

- Vetores de documentos são esparsos: cada vetor que representa um documento da base contém poucas entradas diferentes de zero.

Um ponto importante no trabalho de J. Thorsten [23] quanto à utilização do método SVM na categorização de documentos é a afirmação de que a utilização da atribuição de pesos nos vetores seguindo o modelo TFIDF pode melhorar o desempenho. Para efeito de verificação desta afirmação, os experimentos utilizaram variações nos modelos de atribuição dos pesos.

3.5.4 SVM^{light} – Uma Implementação da Técnica SVM

SVM^{light} é uma implementação da técnica de SVM, desenvolvida por J. Thorsten [22] na linguagem de programação *C*. O algoritmo utilizado tem requisitos de escalonamento de memória e pode lidar com problemas que contenham milhares de vetores de suporte. O programa SVM^{light} é disponível para as plataformas Solaris, Windows, Linux e Cygwin, podendo ser livremente usado em pesquisas científicas. As principais características do programa em sua versão atual 6.01 podem ser vistas abaixo:

- Otimização rápida do algoritmo

    - A seleção dos conjuntos de trabalho é baseada no método *steepest feasible descente*, que tenta diminuir a complexidade da função usada pelo classificador e assim atender a minimização do risco estrutural (SRM) e não apenas a diminuição do risco empírico.

    - Escolha de métodos baseado em regras derivadas do modelo matemático para a solução dos problemas indutivos.

- Soluciona problemas de classificação e regressão.

- Calcula a estimativa da taxa de erro *XiAlpha, precision* e *recall*.

- Manipula milhares de vetor de suporte.

- Calcula eficientemente a estimativa *Leave-One-Out* da taxa de erro, da precisão e da cobertura.

- Usa vetores esparsos na sua representação.

- Suporta funções padrão de kernel linear e permite que sejam definidas outras funções de kernel.

O SVM^{light} consiste basicamente de dois módulos: *svm_learn* e *svm_classify*.

svm_learn

Este é o módulo de treinamento do programa, onde padrões de treinamento são utilizados para extração de conhecimento. A forma geral para chamar este módulo com seus respectivos parâmetros é:

```
svm_learn [options] example_file model_file
```

Onde:

- `Options`: são as opções de configuração do programa, que podem alterar a forma do aprendizado (classificação ou regressão) do mesmo, tipos de kernel, estimativas de desempenho dente outros fatores.

- `example_file:` arquivo de entrada que contém os exemplos de treinamento do programa, a primeira linha pode conter comentários que são ignorados ao começar com "#" . Cada uma das linhas seguintes representa um exemplo de treinamento no seguinte formato:

`<line> .=. <target> <feature>:<value>..... <feature>:<value>`
Onde:

`<target>.=.+1|-1|0`(Para exemplos positivos ou negativos da classe em questão)
`<feature> .=.<integer>` (índice no arquivo de dicionário das características)
`<value> .=. <float>` (Peso associado à característica: TF, TFIDF, ou outro)

A linha abaixo, por exemplo, indica um documento do arquivo de entrada rotulado como negativo e com as características 3, 35 e 43 presentes, associadas respectivamente aos pesos 0.327, 0.583 e 1.002.

-1 3:0.327 35:0.583 43:1.002

- `model_file:` arquivo de saída do módulo de treinamento, onde é armazenado o modelo de classificação, obtido através do treinamento com exemplos contidos em `example_file.`

A Figura 23 mostra uma chamada ao módulo svm_learn para o treinamento de alguns documentos utilizados nos experimentos, que se encontram no arquivo *treinoRabbit.dat*, utilizando a função de base radial como opção de kernel, através do parâmetro "–t 2" que é definido na chamada módulo na figura abaixo, e tendo como saída o modelo de classificação armazenado no arquivo *modelo*.

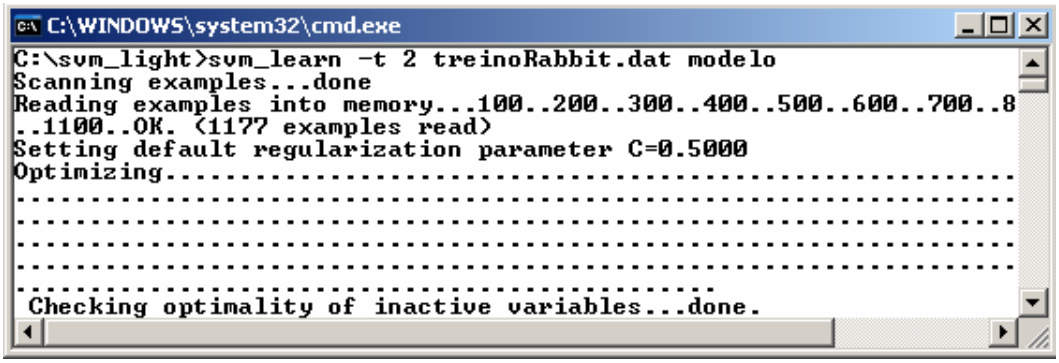

Figura 23. Tela referente à chamada ao módulo *svm_learn*.

svm_classify

Neste módulo é executada a classificação dos documentos com base no modelo de classificação construído no módulo *svm_learn.* A chamada deste módulo com seus respectivos parâmetros para a realização de predições é feita da seguinte forma:

```
svm_classify [options] example_file model_file output_file
```

Onde:

- `Options`: representa as opções de configuração na execução da classificação, que podem ser configurações como: ajuda, nível de exibição de mensagens e escolha do formado de saída.

- `example_file model_file:` arquivo que contem os exemplos de teste no mesmo formato dos exemplos de treinamento no módulo *svm_learn*.

- `output_file:`nome do arquivo de saída onde serão armazenados os valores da função de decisão tomada para cada exemplo classifcado.Neste arquivo os valores são dispostos um por linha na mesma ordem dos correspondentes exemplos de teste, o sinal de cada valor indica a predição da classe tratada como negativa ou positiva.

Na Figura 24 é mostrada a categorização de documentos contidos na base de teste *testeRabbit,* construída a partir da base OHSUMED para testes no presente trabalho, e o armazenamento dos valores da predição no arquivo *predicoes,* através da chamada ao módulo *svm_classify.* Pode ser visto também, pela figura abaixo, a obtenção de uma acurácia de 85.25% na categorização dos 400 documentos presentes.

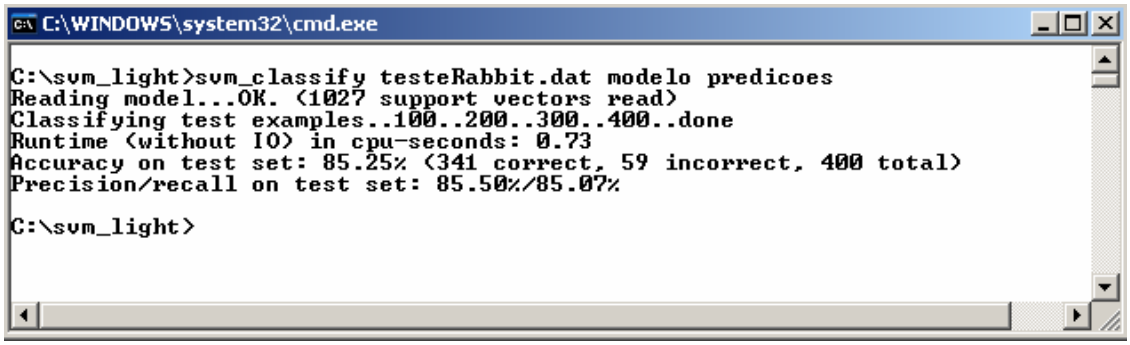

Figura 24. Tela referente à chamada ao módulo *svm_classify*.

Capítulo 4

Experimentos e Resultados

Este capítulo tem como objetivo descrever como foram elaborados os experimentos e os dados utilizados na realização dos mesmos. Apresentando algumas etapas seguidas, com a ferramenta TMSK, para a realização do processamento nos documentos utilizados nos experimentos. Além disso, é apresentada uma análise dos resultados obtidos com a aplicação da técnica de aprendizagem de máquina SVM e o classificador Linear da ferramenta TMSK, comparando os resultados obtidos sob diferentes condições de treinamento, como a mudança de kernel no classificador, seleção de parâmetros dos kernels e modificação na forma de atribuição de peso nos termos da base de dados.

4.1 Obtenção e distribuição dos dados

Os dados utilizados nos experimentos deste trabalho foram obtidos da base de dados OSHUMED, onde constam 348.566 artigos de 270 revistas científicas médicas sobre diversos temas médicos, como já foi visto na seção 2.2.1. Devido ao grande número de documentos pertencentes a várias categorias na base de dados, apenas alguns destes foram selecionados para os experimentos, com base na observação das categorias mais predominates na coleção, e esta seleção ocorreu da seguinte forma:

1. Através da observação do campo ".M", o qual identifica os tópicos abordados nos documentos da base OHSUMED, foi verificada a predominância de documentos pertencentes a três categorias , são elas : *Alcohol, Rabbit* e *cells*. A escolha destas três categorias para os experimentos deveu-se ao fato delas possuírem um espaço de amostras maior para os treinamentos e testes pelo classificador.

2. A base de dados OHSUMED foi colocada no padrão XML por uma classe desenvolvida em Java neste trabalho (cujo código fonte está mostrado no Apêndice A), assim possibilitando a aplicação de técnicas de mineração de texto com a ferramenta TMSK.

3. Para cada categoria foi construído um dicionário local dos documentos que constituem a base de dados, utilizando a rotina *mkdict* da ferramenta TMSK. Na construção desse

dicionário ocorre internamente a retirada das palavras pertencentes à lista de *stopwords*. Na Tabela 6 pode ser visto a quantidade de palavras que formam o dicionário de cada característica.

Tabela 6. Dimensão dos vetores de entrada.

Categoria	Dimensão do Vetor de Entrada (características)
Alcohol	788
Cells	801
Rabbit	812

4. Com o dicionário para cada categoria já construído, passou-se para a fase de vetorização dos documentos em relação a cada uma das três categorias. A rotina *vectorize* da ferramenta TMSK executa sua tarefa com o auxílio de outras rotinas internas, como a de tokenização dos textos, e por meio dos dicionários de cada categoria para indexar os as palavras que formam os documentos.

5. Com os documentos rotulados em relação às três categorias, foi possível formar arquivos de treinamento e teste com igual quantidade de padrões pertencentes e não pertencentes às categorias. As distribuições das quantidades de padrões de treinamento e teste para cada categoria, podem ser visualizadas na Tabela 7.

Tabela 7. Características dos conjuntos de treinamento e de teste.

Categoria	Treinamento (Quant. Padrões)	Teste (Quant. Padrões)
Alcohol	1372	401
Cells	2000	600
Rabbit	1206	402

6. Como a atribuição de pesos nos vetores gerados pela ferramenta TMSK seguem o modelo TF, foi construída uma classe em Java para transformar o peso dos termos no modelo TFIDF (descrita no *Apêndice B*), e assim possibilitar expandir os experimentos com tipos de atribuições de peso diferentes, na ferramenta SVM^{light}. Além dessa funcionalidade, a classe executa algumas formatações nos delimitadores e rótulos dos vetores gerados pelo TMSK, para que assim, seja obtido um formato compreensível pelo SVM^{light}.

Ao término das etapas acima citadas, estavam prontos os arquivos de treinamento e teste com uma estruturação dos dados compreensível pelos algoritmos de aprendizado de máquina, bem como o dicionário de características para cada categoria. Finalmente pode-se partir para a etapa dos experimentos com diferentes modelos de atribuição de pesos, e com os classificadores SVM^{light} e *linear* do sistema TMSK impostos a diferentes condições de aprendizado, como será demonstrado nas seções seguintes.

4.2 Experimentos com TMSK

Os elementos dados como entradas para o classificador, são vetores rotulados como pertencentes ou não pertencentes a cada categoria. Mas os novos documentos fornecidos para serem categorizados não precisam ser etiquetados, a não ser que se deseje obter as medidas e desempenho da categorização, como é o nosso caso. Através da forma de classificação dual, cada documento é categorizado como pertencente ou não em relação à categoria em análise.

4.2.1 Metodologia

No sistema TMSK, temos a opção de alterar alguns parâmetros do funcionamento interno da rotina de classificação, como a forma de atribuição interna dos pesos e o parâmetro de *decision-threshold c*. Dessa forma foram realizados experimentos com a base de dados das três categorias construídas, alterando a forma de atribuição de pesos em: Binary, TF e TFIDF, e com o parâmetro de *decision-threshold c* assumindo os valores 0,1 e -0,25, com o último sendo negativo para verificar sua influência no controle das medidas de precisão e cobertura. A alteração do modelo de atribuição de pesos ocorre internamente na rotina de classificação linear, através do parâmetro feature-type que pertence à ferramenta TMSK. Para cada treinamento executado, a combinação dos parâmetros e arquivos de entrada era alterada manualmente no arquivo de propriedades do sistema.

O algoritmo de aprendizagem do classificador linear, pertencente à ferramenta TMSK, realizou os treinamentos com o número de 400 iterações e com uma taxa de aprendizado de 0,25, que são os valores padrão dos parâmetros.

4.2.2 Resultados

Nas Tabelas 8, 9 e 10 que são apresentadas abaixo, foram avaliadas as médias de algumas medidas de desempenho, como F-measure, que calcula a exatidão das predições, precisão e cobertura, obtidas através de várias execuções feitas no processo de validação Leave-One-Out com os documentos do conjunto de teste, pertencentes às três categorias analisadas. Havendo a variação do parâmetro de *decision-threshold c* e modelo de atribuição dos pesos.

Tabela 8. Resultados Categoria Alcohol.

Tipo Peso	*c*	F-measure	Precisão	Cobertura
Binário	0	87,02%	88,60%	85,50%
	1	80,59%	100,00%	67,50%
	-0,25	85,22%	83,98%	86,50%
TF	0	88,42%	93,33%	84,00%
	1	85,14%	99,33%	74,50%
	-0,25	87,65%	88,32%	87,00%
TFIDF	0	84,92%	85,35%	84,50%
	1	83,28%	96,07%	73,50%
	-0,25	85,01%	83,57%	86,50%

Tabela 9. Resultados Categoria Rabbit .

Tipo Peso	*c*	F-measure	Precisão	Cobertura
Binário	0	83,04%	82,03%	84,07%
	1	75,37%	93,38%	63,18%
	-0,25	81,23%	77,72%	85,07%
TF	0	81,05%	78,24%	84,07%
	1	75,70%	87,58%	66,66%
	-0,25	80,73%	74,89%	87,56%
TFIDF	0	75,22%	69,78%	81,59%
	1	67,96%	77,21%	60,69%
	-0,25	74,83%	67,74%	83,58%

Tabela 10. Resultados Categoria Cells.

Tipo Peso	*c*	F-measure	Precisão	Cobertura
Binário	0	85,76%	86,94%	84,61%
	1	80,45%	94,17	70,23
	-0,25	85,15%	83,12%	87,29%
TF	0	82,63%	83,33%	81,93%
	1	79,61%	94,90%	68,56%
	-0,25	82,02%	80,19%	83,94%
TFIDF	0	78,72%	78,85%	78,59%
	1	77,47%	83,92%	71,57%
	-0,25	78,60%	77,96%	79,26%

Pelos resultados obtidos nos experimentos, pôde ser observada uma grande influência do parâmetro *decision-threshold c* nas medidas de desempenho do processo de categorização. A utilização de valores negativos neste parâmetro, como foi utilizado -0,25, favorece a medida de cobertura, embora os melhores resultados de Acurácia foram obtidos com o valor do parâmetro igual a zero. Quanto ao modelo de atribuição de pesos nos experimentos, levando-se em consideração que a forma de classificação foi dual, os melhores resultados foram conseguidos com o modelo de atribuição de pesos binário, com exceção da categoria *Alcohol*.

4.3 Experimentos com SVM^{light}

A ferramenta SVM^{light} além de realizar uma classificação linear, através do mapeamento de seu espaço de entradas para outro espaço de características de alta dimensão, trabalha também com a utilização de kernels para definir o espaço de características implícito, na qual a máquina de aprendizado linear opera. Os vetores de representação dos documentos são dados como entrada ao classificador, então um modelo de classificação é construído com vetores de suporte que auxiliarão na separação das categorias.

4.3.1 Metodologia

Na ferramenta SVM^{light} existe a possibilidade da utilização de kernels para construção de um modelo de decisão, dessa forma foram realizados experimentos com as três categorias analisadas e alterando-se os parâmetros de penalidade, comum a todo modelo construído, o tipo de kernel e os parâmetros específicos para cada kernel. Como os vetores gerados pelo sistema TMSK se encontravam no modelo de atribuição de pesos por freqüência TF, através da implementação mostrada no Apêndice A, foi possível atribuir pesos aos termos dos vetores seguindo o modelo TFIDF, e assim ampliar os experimentos realizados com a técnica SVM.

Os experimentos aqui descritos utilizaram a classificação linear básica, classificação com kernel RBF e com kernel polinomial, para os três conjuntos de categorias. Como os vetores gerados pela ferramenta TMSK se encontram no modelo de atribuição de pesos por freqüência e as transformações para outros modelos pela ferramenta ocorrem de forma interna, foi utilizado o modelo TF dos vetores gerados e a obtenção de vetores como o modelo de atribuição de pesos por TFIDF, a partir da classe implementada no trabalho Apêndice B. O parâmetro de penalidade de erro *C*, comum à classificação na forma mais básica ou com kernel, teve seu valor ajustado para os seguintes valores: 1, 10 e 100. O kernel RBF teve seu valor γ ajustado para os valores 0,1; 0,01 e 0,001, enquanto o kernel polinomial sofreu algumas variações no valor do expoente *d* de sua função de decisão, assumindo os valores 1, 3 e 9.

4.3.2 Resultados

Nas tabelas apresentadas nesta seção, constam algumas comparações com maior relevância dos resultados obtidos, avaliando as medidas de desempenho utilizadas, como acurácia, precisão e cobertura, para as diferentes condições de treinamento que foram impostas. O resultado completo dos experimentos realizados pode ser analisado no Apêndice C.

Tabela 11. Categoria Alcohol-Linear (TFIDF)

C	Acurácia	Precisão	Cobertura
0.005	89,53%	97,59%	81,00%
1	83,29%	84,10%	82,00%
10	83,79%	84,26%	83,00%
100	83,79%	84,26%	83,00%

Tabela 12. Categoria Alcohol-kernel RBF (TFIDF)

C	G	Acurácia	Precisão	Cobertura
1	0.1	53,12%	51,60%	96,50%
1	0.01	88.53%	87,75%	89,50%
1	0.001	87,78%	98,09%	77,00%
10	0.1	53,12%	51,60%	96,50%
10	0.01	88,78%	87,08%	91,00%
10	0.001	91,27%	97,69%	84,50%
100	0.1	53,12%	51,60%	96,50%
100	0.01	88,03%	85,85%	91,00%
100	0.001	87,28%	91,16%	82,50%

Tabela 13. Categoria Alcohol-kernel Polinomial (TFIDF)

C	D	Acurácia	Precisão	Cobertura
1	1	83,29%	84,10%	82,00%
1	3	75,81%	83,66%	64,00%
1	9	49,88%	49,88%	100,00%
10	1	83,79%	84,26%	83,00%
10	3	75,81%	83,66%	64,00%
10	9	49,88%	49,88%	100,00%
100	1	83,79%	84,26%	83,00%
100	3	75,81%	83,66%	64,00%
100	9	49,88%	49,88%	100,00%

Pode-se notar que para valores muito altos do parâmetro G, que corresponde à largura do kernel RBF, e muito altos do parâmetro *d,* que corresponde ao expoente do kernel polinomial, não foi possível realizar uma generalização, fazendo com que todos os padrões de entrada fossem classificados como pertencentes à categoria e consequentemente atingindo uma taxa de precisão e acurácia em torno de 50% e a taxa de cobertura aproximadamente 100%.

O kernel RBF apresentou medidas de desempenho superiores às outras formas de categorização testadas, com um alto grau de precisão nas predições executas. Na tabela 12, mostrada acima, foram atingidas as melhores medidas de desempenho dos experimentos, com uma taxa de 91,27% para a precisão dos documentos da *base alcohol,* utilizando um kernel RBF, seguindo um modelo de atribuição de pesos TFIDF e com os valores de seus parâmetros mostrados. Isto significa que dos 401 documentos presentes na base de teste da categoria, 366 destes foram categorizados corretamente como pertencentes ou não à categoria.

As condições de treinamento que proporcionaram melhores resultados para cada uma das três categorias analisadas, podem ser verificadas na Tabela 14.

Tabela 14. Melhores Resultados SVM^{light}

Categoria	Kernel	Modelo de pesos	Parâmetros	Acurácia	Precisão	Cobertura
Alcohol	RBF	TFIDF	C=10; G=0,001	91,27%	97,69%	84,50%
Rabbit	RBF	TF	C=10; G=0,001	85,57%	84,88%	86,57%
Cell	RBF	TFIDF	C=10; G=0,001	89,67%	93,41%	85,28%

Os melhores resultados obtidos para as três categorias, demonstram um melhor desempenho na categorização não apenas com a utilização do kernel RBF, como também nos ajustes dos parâmetros de penalidade e gama para C=10 e G=0,001, respectivamente.

Nas Figuras 25, 26 e 27 são mostrados alguns exemplos de categorizações textuais executadas corretamente e incorretamente com as três categorias pré-definidas, utilizando as configurações de treinamento que obtiveram os melhores resultados para cada categoria, como foi visto na Tabela 14.

Na Figura 25, o texto que foi classificado corretamente como pertencente à categoria *Alcohol*, contém palavras destacadas como pertencentes ao dicionário de características da categoria, e estas representam bastante relevância para identificação do documento como pertencentes ou não à categoria. No entanto o texto da figura que foi classificado erroneamente como pertencente à categoria, apresenta uma palavra bastante relevante para a identificação da categoria *Alcohol*, e seguindo um modelo de relevância da palavra no documento por sua freqüência, pode ter levado a encarar o texto como pertencente à categoria.

Classificação Correta na Categoria Alcohol	Classificação Incorreta na Categoria Alcohol
<TEXT> This paper presents data from a general population survey o three areas of Britain which manifest considerable differences in official rates of problem drinking, yet show similar patterns and levels of alcohol consumption.Consideration of various sources of bias (non-response,forgetting, selective under-reporting, and interviewer) suggest that they do not differentially influence. </TEXT>	<TEXT> A case of vulvar hematoma resulting from cunnilingus occurred in an alcoholic woman. Incision and drainage with debridement of devitalized tissues, systemic antibiotics covering both anaerobes and aerobes. </TEXT>

Figura 25. Exemplos de Categorizações realizadas na Classe Alcohol.

A Figura 26 mostra mais um exemplo de documento que foi categorizado corretamente em relação à categoria *Rabbit*, considerando não apenas a relação das palavras do dicionário que aparecem com mais freqüência no documento, mas também aquelas palavras que menos ocorrem nos outros documentos da coleção. Ao mesmo tempo é mostrada na figura 26 mais um caso em que a classificação ocorreu de forma errada, havendo uma palavra de relevância para identificação da categoria.

Classificação Correta na Categoria Rabbit	Classificação Incorreta na Categoria Rabbit
<TEXT> Treatment of experimental corrosive esophageal injury in these animal consisted of ante grade dilatation and administration of two lathyrogenic drugs, colchicine and penicillamine, in various combinations. This study demonstrated that the rabbit can serve as a suitable animal model for study of corrosive injury of the esophagus, that penicillamine given alone affected wound healing with less severe stricture after such an injury. </TEXT>	<TEXT> The monoclonal antibody (MoAb) MM4 reacts with human multiple myeloma (MM) cell lines and bone marrow from patients with plasma cell dyscrasias but not with normal peripheral blood or bone marrow cells. Treatment with MM4 and rabbit complement (C') was cytotoxic to the plasma cell-derived cell lines GM 1312, RPMI 8226, and ARH-77, as demonstrated by chromium release microcytotoxicity and trypan blue exclusion assays. </TEXT>

Figura 26. Exemplos de Categorizações realizadas na Classe Rabbit.

Pela Figura 27, fica visível que as palavras que constituem o dicionário de características da categoria *Cells*, aparecem geralmente como elemento dos textos pertencentes à categoria em maior número do que nos documentos não pertencentes, mas nestes últimos quando ocorrem se o classificador ainda não tiver uma boa generalização para os casos apresentados, o texto será classificado de forma errada.

Classificação Correta na Categoria Cells	Classificação Incorreta na Categoria Cells
<TEXT> The characteristics of the human dermal mast cell population with respect to formalin fixation sensitivity, toluidine blue staining and alcian blue/safranin staining were studied. Thirty-seven specimens of normal human skin were bisected. One half was fixed in 10% neutral buffered formalin and the other in Carnoy's fixative. Sections were cut and stained with either toluidine blue or alcian blue/safranin. Significantly more mast cells were visualized with alcian blue/safranin than with toluidine blue. </TEXT>	<TEXT> With the use of antisera against bovine retinal S-antigen and bovine opsin the authors demonstrate that in cerebellar medulloblastomas certain tumor cells display immunocytochemical properties characteristic of retinal photoreceptors and pinealocytes. S-antigen-like and opsin-like immunoreactions occur in nine of 28 medulloblastomas investigated. </TEXT>

Figura 27. Exemplos de Categorizações realizadas na Classe Cells.

Capítulo 5

Conclusões e Trabalhos Futuros

A utilização de técnicas de aprendizagem de máquina na categorização de textos não é uma tarefa trivial, pois envolve uma série de decisões inerentes às técnicas de aprendizagem utilizadas, bem como a escolha de métodos que melhor representem as informações, ou seja, os métodos de pré-processamento. Um grande problema apresentado na categorização de documentos é alta dimensionalidade dos textos, fazendo-se necessário um pré-processamento custoso, a fim de alcançar uma representação dos dados adequada para o aprendizado de máquina, e algoritmos de aprendizado que apresentem soluções para lidar com um espaço de entradas tão grande.

O objetivo deste trabalho foi realizar um estudo comparativo da utilização de técnicas de aprendizagem de máquina combinadas ao uso de diferentes *kernels*, modelos de atribuição de peso e parâmetros de treinamentos na categorização de documentos.

A técnica SVM foi aplicada através da ferramenta SVM^{light}, que trabalha com a representação de vetores esparsos. Ao se utilizar esse formato de arquivo, as representações de vetores dos documentos ocupam um espaço bem menor que a maneira convencional, à custa de um *overhead* gerado pela utilização de índice de características para cada termo não nulo. Como a quantidade de termos não nulos é bem menor que os termos nulos no vetor, o custo gerado pelo uso dos índices é irrelevante, compensando o uso dessa notação. Quanto à utilização da ferramenta TMSK, foi de grande importância nas etapas de pré-processamento dos documentos, porém seu classificador linear não apresenta uma técnica tão eficiente para validação dos resultados na fase de treinamento.

Os resultados obtidos nos experimentos foram bastante satisfatórios, e apresentaram um ótimo desempenho na categorização de documentos, as taxas de acurácia obtidas na categorização dos documentos mantiveram-se altas, indicando uma boa generalização dos modelos de classificação gerados. O uso de diferentes kernels e ajustes nos parâmetros tiveram bastante influência nas medidas de desempenho, chegando a alcançar 91,27% de precisão na classificação bi valorado para a categoria *Alcohol*, com a utilização do kernel RBF. A técnica SVM associada ao uso do kernel RBF e combinada com os parâmetros C=10 e G=0,001, obteve os melhores resultados para as três categorias analisadas.

Com a realização deste trabalho foram visualizadas algumas oportunidades de trabalhos futuros, associados aos estudos complementares nessa área de extração de conhecimento, como:

- Análise de outros algoritmos para extração de características do texto.
- Avaliação do desempenho do algoritmo de aprendizagem, com outros métodos de tratamentos dos dados.

- Desenvolvimento de sistemas de organização de documentos baseados em aprendizado de máquina.

O presente estudo demonstrou a viabilidade do uso da técnica de aprendizado de máquina SVM na categorização de documentos. O estudo aqui desenvolvido também proporcionou ao autor do presente trabalho a aquisição de novos conhecimentos na área de mineração de texto, categorização de documentos e aprendizagem de máquinas, despertando interesse pelo desenvolvimento de aplicações que possam utilizar a categorização de documentos no processo de segurança eletrônica.

Sistemas de gerenciamento eletrônico de documentos não realizam simplesmente o gerenciamento de arquivos, mas também ações voltadas ao controle de níveis de segurança. O interesse desenvolvido com a elaboração deste trabalho é a implementação da categorização de documentos no suporte ao reconhecimento das ações de segurança a serem tomadas por um sistema de gerenciamento eletrônico.

Bibliografia

[1] BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B. Redes Neurais Artificiais Teoria e Aplicações. Livros Técnicos e Científicos Editora, Rio de Janeiro, 2000.

[2] CAMPBEL, C. *kernel Methods: a Survey of current techniques.* Neurocomputing, v.48, p.63-84, 2002.

[3] CRISTIANINI, N.; SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[4] CRISTIANINI, N.; SHAWE-TAYLOR, J. *Kernel Methods for Pattern Analysis*. Cambridge University Press. 2004.

[5] DAVID, V.; SANCHEZ, A. Advanced Support Vector Machines and Kernel Methods. Neurocomputing, v.55, p.5–20, 2003.

[6] DUDA, R. O., HART, P. E. e STORK, D. G. *Pattern Classification*. Wiley- Interscience, second edition, 2000.

[7] HEARST, M.A. *Text Data Mining: Issues, Techniques, and the Relationship to Information Access*. Presentation notes for UW/MS Workshop on Data Mining, July 1997.

[8] HERSH, W.R.; BUCKLEY, C.; LEONE, T.J.; HICKAM, D.H. OHSUMED: *An Interactive Retrieval Evaluation and New Large Test Collection for Research*. Proceedings of the 17th Annual ACM SIGIR Conference, 1994.

[9] IBM.*IBM Intelligent Miner for Text*. IBM Business Intelligence Solutions CD.1998.

[10] MEDLINE. Disponível em: http://medline.cos.com/, visitado em : 15/05/2006.

[11] OHSUMED Test Collection. Disponível em: <ftp://medir.ohsu.edu/pub/ohsumed>, visitado em: 19/04/2006.

[12] OLIVEIRA, A. L. I.; MEDEIROS, E. A.; ROCHA, T. A. B. V.; BEZERRA, M. E. R.; VERAS, R. C. A. *Study on the Influence of Parameter θ- on Performance of RBF Neural Networks Trained with the Dynamic Decay Adjustment Algorithm*. Fifth International Conference on Hybrid Intelligent Systems, 2005.BAIRD, H.S. Document Image Defect Models. Structured Document Image Analysis, p. 546-556, 1992.

[13] OLIVEIRA, A. L. I.; Melo, B.J.M.; Meira, S.R.L. *"Integrated Method for Constructive Training of Radial Basis Functions Networks"*. IEE Electronics Letters, v.41, n.7, p.429-430, 2005. http://dx.doi.org/10.1049/el:20057296

[14] OLIVEIRA, A. L. I.; NETO, F. B. L.; MEIRA, S. R. L. *Improving RBF-DDA Performance on Optical Character Recognition through Parameter Selection*. Proceedings of the 17th International Conference on v. 4, p. 625–628, 2004.

[15] OLIVEIRA, C.; CASTRO, P. Categorização Múltipla com Árvores de Decisão e Regras. Instituto Militar de Engenharia, Departamento de Engenharia de Sistemas (Relatório Técnico), 2000.

[16] PAULINO, W. R. Biologia Atual. Editora Ática, São Paulo, 1998.

[17] SALTON, G.; WONG, A. e YANG C.S. *A Vector Space Model for Automatic Indexing*. Communications of the ACM, New York, v.18, p.147-150, 1975.

[18] SEBASTIANI, F. *Machine Learning in Automated Text Categorization*. Technical Report IEI B4-31-12-99, Instituto di Elaborazione della Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy. 1999.

[19] SVM – Support Vector Machines. Disponível em: <http://www.dtreg.com/svm.htm>, acessado em 26/04/2006.

[20] Text Mining Software. Disponível em: <http://www.data-miner.com/>,visitado em 12/12/2005.

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

[22] THORSTEN, J. *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning*. B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.Disponível em: <http:// svmlight.joachims.org/>, visitado em 10/10/2005.

[23] THORSTEN, J. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. Proceedings of the European Conference on Machine Learning, Springer, 1998.

[24] THORSTEN, J. *Transductive Inference for Text Classification using Support Vector Machines*. Proceedings of the International Conference on Machine Learning (ICML), 1999.

[25] VAPNIK, V. *Support-vector networks*. Machine Learning, v.20, p.273-297, November 1995.

[26] VAPNIK, V. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[27] Yang, Y. *Expert network: effective and efficient learning from human decisions in text categorization and retrieval*. In Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval , p.13-22, Dublin, Ireland, 1994.

[28] WEBB, A. *Statistical Pattern Recognition*. John Wiley & Sons, second edition, 2002.

[29] WEISS, S.M.; ZHANG, T.; DAMERAU, F. *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer Editora, Edição 1, 2005.

Apêndice A

Conversão OHSUMED-XML

Apresentamos aqui o código Java para converter os documentos que se encontravam na base de dados OHSUMED para o padrão XML, de acordo com as estruturas necessárias para interpretação pela ferramenta TMSK.

```
public class ohsuXML {
  public static void main(String[] args) {

        String line;
        String str;
        String title;
        String body;
        String byline;
        String subject;
        String subjec;
        String source;
        StringBuffer text = new StringBuffer();
        BufferedReader in = null;
        PrintWriter pw = null;
        FileReader inpf;
        try {
                inpf = new FileReader(new File("c:\\ohsumed.88"));
                in = new BufferedReader(inpf);
                pw = new PrintWriter(new FileWriter("c:\\saida.xml"));
        } catch (FileNotFoundException e1) {
                System.out.println("[Arquivo Nao Encontrado]");
        } catch (IOException e) {
                System.out.println("[Problemas no Arquivo Gerado]");
        }
```

```
pw.println("<?xml version=\"1.0\" encoding=\"ISO-8859-1\" standalone=\"yes\"?>");
pw.println("<CORPUS>");

System.out.println("Processando...");
str = "E"; // setado para inicializaação no laço
while (str != null){
body = "";
byline = "";
subject = "";
source = "";
title = "";
text.setLength(0);
try {
        str = in.readLine(); // Lê o primeiro .I (documento)
        while ((str = in.readLine()) != null) {
                if (str.startsWith(".I")){
                        break; //comeca um novo documento
                }
                if (str.startsWith(".U")) {
                        continue;
                }
                else if (str.startsWith(".S")){
                        source = in.readLine(); // origem do documento
                        continue;
                }
                else if (str.startsWith(".A")){
                        byline = in.readLine(); // Autores
                        continue;
                }
                else if (str.startsWith(".T")){
                        title = in.readLine(); //titulo
                        continue;
                }
                else if (str.startsWith(".M")){
                        subject = in.readLine(); //topicos do documento
                        if (subject.endsWith(".")){
                                subject = subject.substring(0,subject.length() - 1);
                        }
                        subject = subject + ";";
                        continue;
                }
                else if (str.startsWith(".W")){  // texto
```

```
                        while (!body.endsWith(".")) {
                                body = body + in.readLine();
                        }
                        continue;
                }
                else {
                        continue;
                }
        }//fim do while
} catch (IOException e) {
        System.out.println("[Problemas na Execucao das Linhas]");
}
if (body.length() == 0) continue;

// salvando os dados separados no formato XML

        text.append(body);
        int bodlen = body.length();
        pw.println("<DOC>");
        pw.println("<TITLE>");
        pw.println(title);
        pw.println("</TITLE>");
        pw.println("<AUTHOR>");
        pw.println(byline);
        pw.println("</AUTHOR>");
        pw.println("<TOPICS>");
        StringTokenizer tksu = new StringTokenizer(subject,";");
        while (tksu.hasMoreTokens()){
                subjec = tksu.nextToken();
                 if (subjec.indexOf("/") > 0){
                        subjec = subjec.substring(0,subjec.indexOf("/"));
                 }
                subjec = subjec.trim();
                  pw.println("<TOPIC>");
                  pw.println(subjec);
                 pw.println("</TOPIC>");
        }//fim do while
        pw.println("</TOPICS>");
        pw.println("<TEXT>");
        int nech = 0;
        int sch = 0;
        int ech = 60;
```

```
        while(ech < bodlen) {
                nech = ech;
                for (int itx = ech; itx > sch; itx--){
                        if (!(text.charAt(itx)== ' ')) {
                                nech = itx;
                        } else break;
                }//laco for
                ech = nech - 1;
                 line = text.substring(sch,ech);
                pw.println(line);
                 sch = nech;
                ech = sch + 60;
        }//fim do while
         pw.println("</TEXT>");
          pw.println("</DOC>");
        pw.println("</CORPUS>");
        try {
                in.close();
                 pw.close();
        } catch (IOException e2) {
        System.out.println("[Problemas no Fechamento da Porta]");
        }
    }
}
```

Apêndice B

Atribuição de Pesos TFIDF

Apresenta-se aqui o código da classe Java implementada para receber os vetores construídos pela ferramenta TMSK e transformar num formato compreensível pelo programa SVM^{light} , atribuindo pesos ao termos dos vetores através do modelo TFIDF.

```
public class TFIDF {
        private static TFIDF aInstancia=new TFIDF()
        public static TFIDF getInstancia(){
                if(aInstancia==null){
                    aInstancia=new TFIDF();
                }
                return aInstancia;
        }
        public static void main(String[] args) throws IOException{
                double numdocs=0.0;
                BufferedReader in=null;
                PrintWriter out=null;
                String etiqueta;
                String linha="";
                String termo;
                String carac;
                String novodoc;
                double tf;
                double tfidf;
                double aparicoesDoc;
                int freqDoc;
                try{
                FileReader arquivo=new FileReader(new File("c:\\testeRabbit.vec"));
                in=new BufferedReader(arquivo);
                out=new PrintWriter(new FileWriter("c:\\testeRabbitTFIDF"));
                numdocs=(double)TFIDF.getInstancia().calculaNumDocs();
```

```
        }catch(Exception e){
            System.out.println("Arquivo não encontrado");
        }
        int contaLinhas=0;
        while((linha=in.readLine())!=null){
                contaLinhas++;
                String novalinha=" ";
                StringTokenizer stk=new StringTokenizer(linha," ");
                etiqueta=stk.nextToken();
                System.out.println("Linha:"+contaLinhas);
                while(stk.hasMoreTokens()){
                        termo=stk.nextToken();
                        carac=termo.substring(0,termo.indexOf("@"));
                        tf=Double.parseDouble(termo.substring(termo.indexOf("@")+1));
                        aparicoesDoc=TFIDF.getInstancia().verificaDocs(carac);
                        tfidf=tf*((Math.log(numdocs/aparicoesDoc))/2.30258509299405);

                        novalinha=novalinha+carac+":"+tfidf+" ";
                }//fim do while

                if(etiqueta.equals("1")){
                        novodoc="+1"+novalinha;
                }else{
                        novodoc="-1"+novalinha;
                }
                out.println(novodoc);
                out.flush();
        }//fim do while
}

public int calculaNumDocs() throws IOException{
        FileReader arquivo=new FileReader(new File("c:\\testeRabbit.vec"));
        BufferedReader in=new BufferedReader(arquivo);
        int numDoc=0;

        while(in.readLine()!=null){
                numDoc++;
        }
        return numDoc;
}// fim do metodo

public int verificaDocs(String caracter){
```

```
		BufferedReader in =null;
		try{
		FileReader arquivo=new FileReader(new File("c:\\testeRabbit.vec"));
		in=new BufferedReader(arquivo);
		}catch(Exception e){
			System.out.println("problema nas buscas");
		}
		String etiqueta;
		String termo;
		int contdocs=0;
		int cont=0;
		String linha=null;
		try {
			while((linha=in.readLine())!=null){
				StringTokenizer stkaux=new StringTokenizer(linha," ");
				etiqueta=stkaux.nextToken();
				while(stkaux.hasMoreTokens()){
					String elemento=stkaux.nextToken();
					String caracteristica=elemento.substring(0,elemento.indexOf("@"));
					if(caracteristica.equals(caracter)){
						contdocs++;
					}
				}
				cont++;
			}
		} catch (IOException e) {
			System.out.println("Erro na busca pelo termo");
		}
		return contdocs;
	}

}
```

Apêndice C

Resultado Completo dos Experimentos

Aqui são apresentados os resultados dos experimentos realizados com a ferramenta SVM^{light} , utilizando as bases de dados das três categorias formadas, com a finalidade de proporcionar uma melhor avaliação do desempenho obtido, por meio de comparações de algumas medidas sobre perspectivas diferentes de treinamento.

Base Alcohol

Kernel Linear-TF

C	Acurácia	Precisão	Cobertura
0,0086	90,27%	99,39%	81,00%
1	87,53%	90,32%	84,00%
10	87,03%	88,14%	85,50%
100	87,03%	88,14%	85,50%

Kernel Linear-TFIDF

C	Acurácia	Precisão	Cobertura
0,005	89,53%	97,59%	81,00%
1	83,29%	84,10%	82,00%
10	83,79%	84,26%	83,00%
100	83,79%	84,26%	83,00%

Kernel RBF-TF

C	G	Acurácia	Precisão	Cobertura
1	0,1	56,11%	53,35%	95,50%
1	0,01	90,52%	97,65%	83,00%
1	0,001	86,53%	100,00%	73,00%
10	0,1	56,86%	53,78%	96,00%
10	0,01	90,02%	95,98%	83,50%
10	0,001	91,27%	99,40%	83,00%
100	0,1	56,86%	53,78%	96,00%
100	0,01	89,78%	94,92%	84,00%
100	0,001	90,77%	95,03%	86,00%

Kernel RBF-TFIDF

C	G	Acurácia	Precisão	Cobertura
1	0,1	53,12%	51,60%	96,50%
1	0,01	88,53%	87,75%	89,50%
1	0,001	87,78%	98,09%	77,00%
10	0,1	53,12%	51,60%	96,50%
10	0,01	88,78%	87,08%	91,00%
10	0,001	91,27%	97,69%	84,50%
100	0,1	53,12%	51,60%	96,50%
100	0,01	88,03%	85,85%	91,00%
100	0,001	87,28%	91,16%	82,50%

Kernel Polinomial-TF

C	d	Acurácia	Precisão	Cobertura
1	1	87,53%	90,32%	84,00%
1	3	80,80%	83,24%	77,00%
1	9	50,12%		
10	1	87,03%	88,14%	85,50%
10	3	80,80%	83,24%	77,00%
10	9	50,12%		
100	1	87,03%	88,14%	85,50%
100	3	80,80%	83,24%	77,00%
100	9	50,12%		

Kernel Polinomial-TFIDF

C	d	Acurácia	Precisão	Cobertura
1	1	83,29%	84,10%	82,00%
1	3	75,81%	83,66%	64,00%
1	9	49,88%	49,88%	100,00%
10	1	83,79%	84,26%	83,00%
10	3	75,81%	83,66%	64,00%
10	9	49,88%	49,88%	100,00%
100	1	83,79%	84,26%	83,00%
100	3	75,81%	83,66%	64,00%
100	9	49,88%	49,88%	100,00%

Base Rabbit

Kernel Linear-TF

C	Acurácia	Precisão	Cobertura
0,0084	84,33%	85,57%	82,59%
1	80,35%	77,98%	84,58%
10	79,35%	77,06%	83,58%
100	79,35%	77,06%	83,58%

Kernel Linear-TFIDF

C	Acurácia	Precisão	Cobertura
0,0044	76,62%	75,60%	78,61%
1	74,63%	71,43%	82,09%
10	74,63%	71,43%	82,09%
100	74,63%	71,43%	82,09%

Kernel RBF-TF

C	G	Acurácia	Precisão	Cobertura
1	0,1	55,22%	52,83%	97,51%
1	0,01	72,89%	67,16%	89,55%
1	0,001	76,62%	74,88%	80,10%
10	0,1	55,47%	53,01%	96,52%
10	0,01	74,13%	68,03%	91,04%
10	0,001	85,57%	84,88%	86,57%
100	0,1	55,47%	53,01%	96,52%
100	0,01	73,63%	67,53%	91,04%
100	0,001	80,10%	77,63%	84,58%

Kernel RBF-TFIDF

C	G	Acurácia	Precisão	Cobertura
1	0,1	53,98%	52,08%	99,50%
1	0,01	63,43%	58,77%	90,05%
1	0,001	73,13%	71,04%	78,11%
10	0,1	55,22%	52,77%	99,50%
10	0,01	67,91%	62,08%	92,04%
10	0,001	78,61%	74,89%	86,07%
100	0,1	55,22%	52,77%	99,50%
100	0,01	67,91%	62,08%	92,04%
100	0,001	77,86%	73,73%	86,57%

Kernel Polinomial-TF

	d	Acurácia	Precisão	Cobertura
1	1	80,35%	77,98%	84,58%
1	3	63,68%	66,57%	54,73%
1	9	50,00%		
10	1	79,35%	77,06%	83,58%
10	3	63,68%	66,57%	54,73%
10	9	50,00%		
100	1	79,35%	77,06%	83,58%
100	3	63,68%	66,57%	54,73%
100	9	50,00%		

Kernel Polinomial-TFIDF

C	d	Acurácia	Precisão	Cobertura
1	1	74,63%	71,43%	82,09%
1	3	62,19%	71,30%	40,80%
1	9	50,00%		
10	1	74,63%	71,43%	82,09%
10	3	62,19%	71,30%	40,80%
10	9	50,00%		
100	1	74,63%	71,43%	82,09%
100	3	62,19%	71,30%	40,80%
100	9	50,00%		

Base Cell

Kernel Linear-TF

C	Acurácia	Precisão	Cobertura
0,0089	88,17%	95,60%	79,93%
1	80,83%	82,17%	78,60%
10	78,17%	78,38%	77,59%
100	79,33%	79,86%	78,26%

Kernel Linear-TFIDF

C	Acurácia	Precisão	Cobertura
0,0052	89,50%	96,83%	81,61%
1	79,17%	80,00%	77,59%
10	78,33%	78,45%	77,93%
100	78,33%	78,45%	77,93%

Kernel RBF-TF

C	G	Acurácia	Precisão	Cobertura
1	0,1	63,83%	58,37%	95,65%
1	0,01	89,33%	91,81%	86,29%
1	0,001	87,50%	97,86%	76,59%
10	0,1	65,50%	59,58%	95,65%
10	0,01	87,00%	86,71%	87,29%
10	0,001	88,67%	94,25%	82,27%
100	0,1	65,50%	59,58%	95,65%
100	0,01	86,50%	85,39%	87,96%
100	0,001	86,17%	89,13%	82,27%

Kernel RBF-TFIDF

C	G	Acurácia	Precisão	Cobertura
1	0,1	61,17%	56,42%	96,99%
1	0,01	86,67%	84,76%	89,30%
1	0,001	88,67%	96,39%	80,27%
10	0,1	62,00%	57,00%	96,66%
10	0,01	84,67%	81,08%	90,30%
10	0,001	89,67%	93,41%	85,28%
100	0,1	62,00%	57,00%	96,66%
100	0,01	85,33%	81,68%	90,97%
100	0,001	83,83%	85,07%	81,94%

Kernel Polinomial-TF

C	d	Acurácia	Precisão	Cobertura
1	1	80,83%	82,17%	78,60%
1	3	82,33%	87,55%	75,25%
1	9	50,00%		
10	1	78,17%	78,38%	77,59%
10	3	82,33%	87,55%	75,25%
10	9	50,00%		
100	1	79,17%	79,79%	77,93%
100	3	82,33%	87,55%	75,25%
100	9	50,00%		

Kernel Polinomial-TFIDF

C	d	Acurácia	Precisão	Cobertura
1	1	79,17%	80,00%	77,59%
1	3	75,00%	83,41%	62,21%
1	9	50,00%		
10	1	78,33%	78,45%	77,93%
10	3	75,00%	83,41%	62,21%
10	9	50,00%		
100	1	78,33%	78,45%	77,93%
100	3	75,00%	83,41%	62,21%
100	9	50,00%		