UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

SELEÇÃO DE CARACTERÍSTICAS PARA CLASSIFICAÇÃO DE TEXTO

TRABALHO DE GRADUAÇÃO

Aluno: Hially Rodrigues de Sá (hrs@cin.ufpe.br)
Orientadores: Flávia de Almeida Barros (fab@cin.ufpe.br) e
Ricardo B. C. Prudêncio (rbcp@cin.ufpe.br)

Janeiro de 2008

Resumo

A crescente disponibilização de documentos de texto na Internet vem fazendo com que a tarefa de classificação de texto ganhe cada vez mais utilidade. A classificação de textos pode ser aplicada em uma grande variedade de contextos como, por exemplo: indexação automática de textos, identificação de autores de textos, filtragem de emails, classificação hierárquica de páginas da Internet e geração automática de metadados.

Contudo, o imenso volume de textos on-line traz o problema da alta dimensionalidade de atributos (palavras), caracterizando uma grande quantidade de termos redundantes e/ou irrelevantes, que muitas vezes comprometem o desempenho dos sistemas de classificação. Normalmente, para remediar esse problema aplicam-se técnicas de seleção de características como uma etapa inicial do processo de classificação, afim de diminuir o ruído dos termos irrelevantes assim como reduzir o custo computacional.

Este trabalho de graduação traz uma apresentação geral sobre classificação de texto, e aprofunda-se nos métodos de seleção de termos (focando naqueles que utilizam aprendizagem de máquina). Foi implementada uma pequena aplicação que possibilita a análise comparativa de algumas técnicas, assim como evidencia a importância e as vantagens dessa etapa de processamento de texto.

Agradecimentos

Primeiramente à minha família e parentes que sempre me incentivaram e foram os responsáveis pelo sucesso desta graduação. Aos meus orientadores, Flávia Barros e Ricardo Prudêncio pelas dicas e dúvidas respondidas e pelo acompanhamento. E a todos os colegas que fiz durante a graduação, em especial aos da minha turma.

Sumário

Lista de Figuras

Quadros

1. Introdução

O incrível aumento de documentos on-line, possibilitado pela expansão da Internet [Graham, 2006], renovou os interesses na classificação automática de documentos e na mineração de dados. A classificação de textos é a tarefa de associar textos em linguagem natural a rótulos pré-definidos. Esse problema vem sendo tratado desde 1960, porém só por volta de 1990, a classificação de textos começou a ser amplamente aplicada, graças ao desenvolvimento de máquinas mais potentes e da facilidade de publicação de textos em forma eletrônica [Sebastiani, 1999].

Os trabalhos iniciais na área de classificação eram principalmente baseados em métodos heurísticos, isto é, em aplicar um conjunto de regras baseadas em conhecimento especializado. Atualmente, contudo, o foco tem se voltado quase que completamente para aprendizagem automática e técnicas de clustering, passando a englobar muitos conceitos de extração de informação, e possuindo muitas características em comum com outras tarefas como extração de conhecimento e mineração de textos [Kushmerick & Thomas 2003].

O maior problema da classificação de texto nos dias de hoje tem sido a grande quantidade de termos nos documentos usados como características (atributos) que representam esses documentos. Isto acaba sendo proibitivo para a maioria dos algoritmos de classificação. Então, é imprescindível reduzir a quantidade de atributos, sem contudo sacrificar a precisão na classificação. Por isso, uma etapa de seleção de atributos é essencial para remover os termos não-informativos dos documentos.

Outro benefício que a remoção de termos irrelevantes traz é eliminar ou reduzir o *Overfitting* que muitas vezes ocorre quando o classificador se ajusta demais à base de treinamento, e não conseguindo assim classificar bem os documentos de teste.

Este trabalho tem como objetivo investigar técnicas para seleção de características aplicadas ao problema de Classificação de Texto baseada em Aprendizado de Máquina. para tanto, foram implementados algoritmos de classificação para estimar resultados.

Além deste capítulo, este documento conta com mais cinco capítulos, como a seguir:

Capítulo 2 – Apresenta conceitos sobre a classificação de texto utilizando aprendizagem de máquina.

Capítulo 3 – Discute sobre a importância de um termo em um documento, e apresenta algumas das técnicas de seleção de termos mais usadas na literatura, mostrando seus pontos fortes e fracos.

Capítulo 4 – Descreve o trabalho desenvolvido, mostrando como foi implementado o sistema de classificação, com foco nas técnicas de seleção de características.

Capítulo 5 – Apresenta a metodologia de testes e os resultados obtidos.

Capítulo 6 – Traz as conclusões deste Trabalho de Graduação.

2. Classificação de Texto

Considere C = {c1, c2, ..., cm} como um conjunto de categorias (classes) e D = {d1, d2, ..., dm} como um conjunto de documentos. A tarefa de classificação de texto consiste em atribuir para cada par (ci, dj) de C x D (com 1 <= i <= m e 1 <= j <= n) um valor de 0 ou 1, isto é, 0 se o documento dj não pertence a ci.

Na comunidade de pesquisa, a abordagem dominante para esse problema é baseado em técnicas de aprendizagem de máquina: um processo indutivo e geral que automaticamente constrói um classificador por aprendizagem, de um conjunto predefinido de categorias e suas características. A vantagem dessa abordagem é a precisão alcançada comparável à alcançada por humanos especialistas no domínio.

Existe uma série de problemas que são de difícil solução que podem ser bem resolvidos com as técnicas de aprendizado de máquina. Por exemplo: algumas tarefas só podem ser bem definidas através de exemplos, onde não é conhecida nenhuma relação à priori entre os dados de entrada e a saída desejada. Além disso, tarefas que são sujeitas a muitas mudanças podem ser aprendidas pela máquina, uma vez que a máquina se adapta a tais mudanças.

A capacidade de adaptação de sistemas de aprendizado de máquina também permite que tais sistemas sejam portados para domínios completamente diferentes sem a necessidade de recriar o sistema para suportar o novo domínio. Assim, se uma máquina é capaz de aprender a classificar notícias de jornais em uma hierarquia de categorias, a mesma máquina também é capaz de aprender a filtrar spams ou até mesmo resolver problemas de ambigüidade em palavras no processamento de linguagem natural.

Na seção 2.1 veremos uma breve abordagem sobre aprendizagem automática de classificadores e na seção 2.2 discutiremos dois bem conhecidos algoritmos de classificação.

2.1. Aprendizagem Automática de Classificadores

Nos sistemas classificadores observam-se duas fases bem distintas: a primeira é a fase de treinamento onde o sistema constrói uma base de conhecimento a partir de um conjunto de exemplos que lhe são fornecidos (o sistema “aprende”), para que na próxima

fase, a fase de uso, utilize esse conhecimento adquirido para classificar os documentos que têm categoria desconhecida. A seguir estas duas fases são melhor detalhadas.

2.1.1 Fase de Treinamento

Na abordagem de aprendizagem de máquina, os documentos pré-classificados são a chave da classificação. Os classificadores construídos por técnicas de aprendizagem de máquina hoje em dia alcançam expressivos níveis de efetividade fazendo a classificação automática qualitativamente viável sobre a classificação manual.

O funcionamento de um sistema de categorização automática na fase de treinamento funciona de acordo com a Figura 2.1 [Moulinier, 96]. Dada uma coleção de documentos ao sistema, ocorre inicialmente o pre-processamento do texto. Esta fase é semelhante à utilizada em sistemas de IR. Após isto as representações iniciais dos documentos são enviadas ao módulo de seleção de termos, que pode reduzir drasticamente o tamanho das representações criando o conjunto de treinamento. Este conjunto é então enviado ao algoritmo de aprendizagem, que ao fim inclue novas regras de categorização na base de conhecimento.

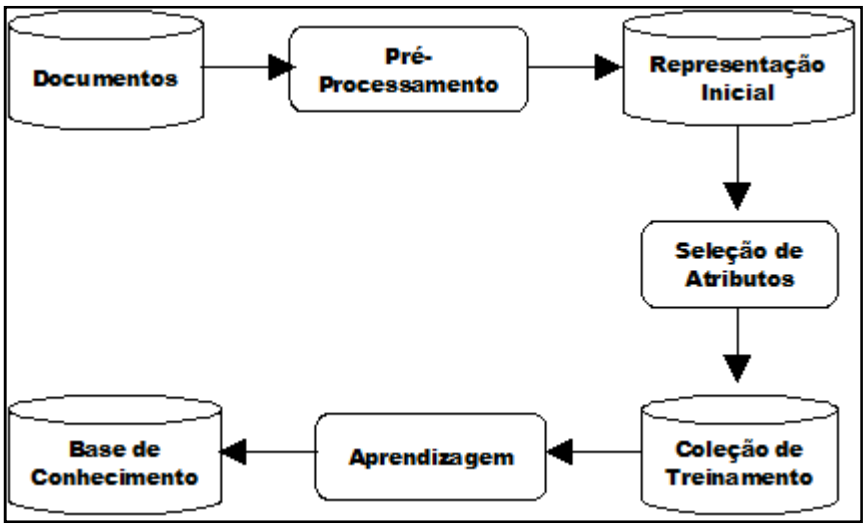

Figura 2.1 Fase de Treinamento

2.1.2 Fase de Uso

A fase de uso de um sistema de classificação baseada em aprendizagem de máquina se dá com os documentos de teste, primeiramente os documentos são pre-processados até que sejam transformados numa representação apropriada (estágio igual à fase de treinamento). Após isso, aplica-se a seleção de atributos de acordo como foi definido e feito no conjunto de treinamento, criando a coleção de teste. A partir daí, o classificador usa a base de conhecimento adquirido (produzido na fase de treinamento) para classificar cada item da coleção de teste.

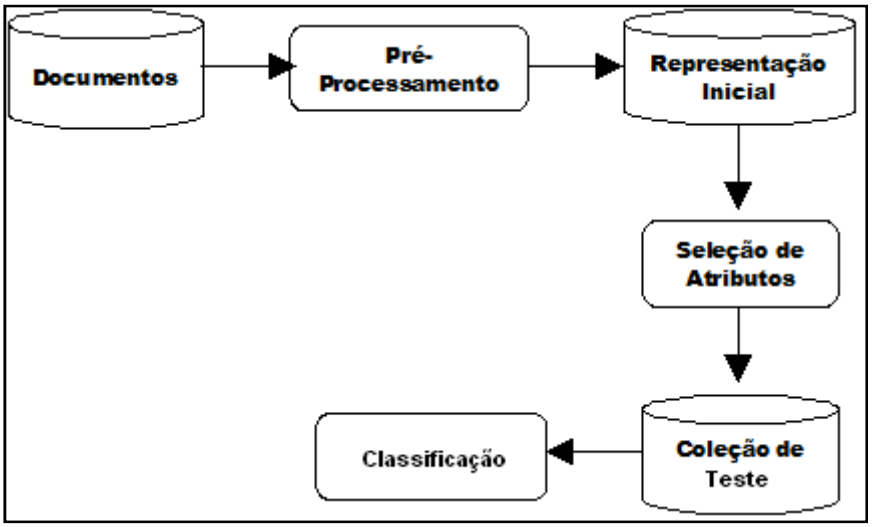

Figura 2.2 Fase de Uso

2.2. Algoritmos de Aprendizagem de Classificadores de Texto

Existem diversas técnicas de aprendizagem de máquina utilizadas para a categorização automática de textos, destacam-se: Redes Neurais (backpropagation, multi layers perceptron), Aprendizagem Baseada em Instâncias (KNN) [WANG, 2006], Algoritmos Genéticos (Genitor e CHC) [Chen94], Aprendizagem Simbólica (algoritmos ID3 e ID5R) [Moulinier, 96] e Aprendizagem Bayesiana (redes bayesianas, naive bayes) [Lewis94].

Dentre essa gama de possibilidades, serão apresentados dois importantes algoritmos de classificação bastante utilizados pela comunidade por serem bem simples e eficazes e que serão implementados neste trabalho.

2.2.1 K-Nearest Neighbors (KNN)

O KNN é um algoritmo de aprendizagem supervisionado, pertencente a um grupo de técnicas denominado de *Instance-based Learning* ou *Lazy Learning* que tem sido usado em muitas aplicações no campo da mineração de dados, reconhecimento de padrões estatísticos, processamento de imagens e entre outros. É considerado um dos melhores métodos para a classificação de texto, é simples e efetivo e escalonável para grandes aplicações. Algumas aplicações de sucesso incluem reconhecimento de escrita à mão e imagens de satélite.

O algoritmo K-Nearest Neighbors classifica um dado elemento desconhecido (de teste) baseado nas categorias dos K elementos vizinhos mais próximos, ou seja, os elementos do corpus de treinamento que obtiveram os graus de similaridade mais altos com o elemento de teste.

Calcula-se a similaridade de cada um dos elementos do corpus de treino com o elemento que se quer classificar e então ordena os componentes da base de treino do mais similar ao menos. Dos elementos ordenados, selecionam-se os K primeiros, que irão servir como parâmetro para a regra de classificação.

Dois pontos importantes do KNN são: a regra de classificação e a função que calcula a similaridade. A regra de classificação diz como o algoritmo vai tratar a importância de cada uns dos K elementos mais próximos. A função de similaridade vai mensurar quão dois elementos são semelhantes de forma a poder identificar quais são os K-NN.

Para K = 1 não existe regra de classificação, pois o elemento de classe desconhecida terá a mesma classificação do vizinho mais próximo. Para k > 1 é preciso uma regra para decidir a qual classe se atribuirá o elemento. Em [WANG, 2006] são revistas duas regras classificação bem comuns: maioria na votação (majority voting scheme) e a soma do peso da similaridade (weighted-sum voting scheme). Na primeira, cada elemento tem uma influência igual, a classe escolhida será aquela que tiver mais representantes entre os k elementos. Na segunda, entre os k elementos, são somadas as

similaridades dos elementos de mesma categoria, o elemento desconhecido será classificado na categoria que obtiver maior valor. As funções do cálculo de similaridade mais conhecidas na literatura são: a do cálculo do cosseno de dois vetores e a distância euclidiana. A seguir a fórmula do cosseno:

$$sim(X, D_j) = \frac{\sum_{t_i \in (X \cap D_j)} x_i \times d_{ij}}{\|X\|_2 \times \|D_j\|_2}$$

Figura 2.3 Cálculo da similaridade pelo cosseno

Onde X é documento de teste representado como vetor; D_j é um documento de treino também representado como vetor t_i é um termo que ocorre nos dois documentos; x_i é o peso do termo em X; d_{ij} é o peso do termo em D_j; $||X||_2$ e $||D_j||_2$ são as normas.

A performance do sistema é muito sensível à escolha do parâmetro k. A computação do algoritmo é linear (tempo e espaço) para o tamanho do conjunto de treino, o maior trabalho consiste em ordenar os elementos do conjunto de treinamento para se obter os K melhores.

As desvantagens desse algoritmo é sua eficiência, ele precisa comparar o elemento de teste com todos os elementos do conjunto de treino. Além disso, a performance dele depende fortemente de dois fatores que são a função de similaridade e o valor do parâmetro k [Wang, 2006]. A maior desvantagem do KNN é que ele usa todos os atributos quando computa a similaridade dos documentos de treino com o de teste. Isso pode levar a medidas pobres e erros de classificação, quando apenas um subconjunto de características é útil para a classificação. Outra é que categorias com bastante exemplos de treino tendem a dominar a classificação do elemento desconhecido, devido a isso eles tendem a saírem mais entre os K vizinhos.

A melhor escolha do parâmetro K depende da base de documentos (treinamento + teste), normalmente grandes valores de K reduzem o efeito do ruído na classificação, mas prejudicam categorias menos distintas. Um bom valor de K pode ser obtido por várias técnicas de heurísticas, por exemplo, validação cruzada (Cross Validation), a ser descrita posteriormente. A precisão do algoritmo pode ser severamente prejudicada pela presença de ruído ou características irrelevantes ou pela seleção de características que não são consistentes com sua importância.

2.2.2 K-Naive Bayes

Classificadores bayesianos tem ganhado popularidade ultimamente, mostrando terem ótima performance. Um dos modelos mais simples é classificador Naive Bayes, ele supõe que todos os atributos dos exemplos são independentes uns dos outros dado o contexto da categoria. É a chamada “suposição de Naive Bayes”. Mesmo sendo esta suposição claramente falsa no mundo real, este modelo executa a classificação muito bem, [Domingos & Pazzani, 1997] mostraram teoricamente que a suposição de independência de palavras na maioria dos casos não prejudica a eficiência do classificador. Este paradoxo é explicado pelo fato de que a avaliação da classificação é apenas uma função de atribuir de uma função de estimação; a função de aproximação pode ser baixa enquanto a precisão da classificação é alta. Por causa da suposição de independência, os parâmetros de cada atributo podem ser aprendido separadamente, isto simplifica bastante a aprendizagem, especialmente quando o número de atributos é alto, apesar de ser ruim para a classificação.

Basicamente, existem dois tipos de modelos estatísticos para os classificadores Naive Bayes. O modelo multinomial, que representa um documento como um vetor de freqüências das palavras no texto (ou uma outra forma de peso para o termo) e o modelo binário, que representa um documento como um vetor binário de palavras, considerando apenas a ocorrência das palavras no texto, onde um valor 0 na posição k significa que o documento não possui nenhuma ocorrência do termo t_k e um valor 1 significa que o documento possui pelo menos uma ocorrência do termo t_k. O modelo binário foi apresentado em [Robertson & Sparck Jones, 1976; Koller & Sahami, 1997]. Já o modelo multinomial foi apresentado em [Joachims, 1998; McCallum & Nigam, 1998]. [Lewis 1994] apresenta uma comparação teórica entre o modelo binário e o modelo multinomial. Já [McCallum & Nigam, 1998], através de experimentos, comparam o modelo multinomial com o modelo binário e concluem que o modelo multinomial apresenta melhores resultados.

A fórmula para calcular a probabilidade (modelo multinomial) de um dado documento pertencer a uma categoria é dado por:

$$p(C|D) = \frac{p(C)}{p(D)} p(D|C)$$

Figura 2.4 Probabilidade da categoria dado o documento

Onde:

p(C) = Probabilidade da categoria ocorrer. Equivale ao número de documentos nessa categoria dividido pelo número total de documentos na base de treino.

p(D) = Probabilidade do documento ocorrer. Esta probabilidade é resultado da multiplicação das probabilidades de cada termo (que compõe o documento) ocorrer na base de treino (suposição de Naive Bayes, os termos são considerados independentes uns dos outros).

p(D|C) = Probabilidade do documento ocorrer, dado que a categoria ocorreu. Esta probabilidade é resultado da multiplicação das probabilidades condicionais de cada termo (que compõe o documento) ocorrer dado que ocorreu a categoria.

p(C|D) = Dado que o documento ocorreu, qual a probabilidade de ele pertencer a categorias.

Este cálculo é feito para cada categoria existente, a categoria que obtiver o maior valor será a categoria do documento.

Em p(D|C), se uma dada classe e um termo não ocorrem juntos no conjunto de treinamento a probabilidade para esse caso resultará em zero. Isto é problemático pois zerará todas as outras probabilidades quando forem multiplicadas. Pode ser desejável incorporar pequenos ajustes nos cálculos das probabilidades para evitar a probabilidade nula. Um dos ajustes bem conhecidos chama-se: Regra da Sucessão de Laplace [Polya, 1968] onde no cálculo de cada uma das probabilidades soma-se mais um no numerador e mais dois no denominador, isso elimina a possibilidade de ocorrer zero no resultado.

2.3. Avaliando a Precisão da Classificação – Cross Validation

Cross Validation, às vezes chamado de rotação de estimativas, é uma prática estatística de particionamento de uma amostra dos dados em subconjuntos de tal modo que a análise é inicialmente executada num único subconjunto, enquanto os outros subconjuntos são mantidos para treino [Kohavi, 1995].

Esta técnica pode ser usada para estimar a generalização do erro de um dado modelo, ou pode ser usado para a seleção de um modelo pela escolha de um em muitos modelos, o qual terá a menor generalização do erro estimado. A seguir, três formas de Cross Validation serão apresentadas.

2.3.1 Holdout Validation

É a forma mais simples de validação. O corpus é dividido em dois subconjuntos, chamados de subconjunto de treino e subconjunto de teste (alguns não consideram como cross validation porque os dados não são cruzados de nenhuma forma). Normalmente menos de um terço do corpus é usado para testes. A vantagem dessa modalidade é que ela não é computacionalmente pesada. Entretanto o resultado da avaliação pode ter alta variação, pois o resultado será significativamente diferente a depender de como foi a divisão.

2.3.2 K-Fold Cross Validation

É uma maneira de otimizar a modalidade anterior. Nesse método de avaliação, os documentos são aleatoriamente divididos em *K* partições mutuamente exclusivas (“folds”) de tamanho aproximadamente igual a *n*/*K*, onde *n* é o tamanho do conjunto de documentos. Então, são realizados *K* experimentos, onde, em cada experimento, uma partição diferente é escolhida para o teste e as *K*–1 partições restantes são escolhidas

para o treinamento. A medida de eficiência é a média das medidas de eficiência calculadas para cada uma das partições.

A grande vantagem dessa técnica, é que todos os documentos são usados tanto para treinamento quanto para teste. Assim, a forma como a divisão do corpus foi feita influi menos no resultado final, qualquer documento será usado exatamente uma vez para teste e k-1 vezes para treinamento. A variação da estimativa é reduzida a medida que o parâmetro k aumenta. A desvantagem dessa modalidade é que o algoritmo de treino terá que ser re-executado k vezes, o que significa que levará k vezes mais computação para fazer a avaliação.

2.3.3 Leave-one-out Cross Validation

É o K-Fold Cross Validation levado ao extremo, ou seja, com K igual a N, onde N representa o número de documentos no corpus. Isto quer dizer que o algoritmo de treino será executado N vezes, com todos os documentos no conjunto de treino exceto um que será o de teste. O resultado dado por essa modalidade é bom, mas ela toma um custo computacional muito alto.

2.4. Considerações Finais

Como foi visto, categorização de textos é o problema de atribuir categorias predefinidas para documentos de texto livres. Enquanto mais e mais informações textuais ficam disponíveis de forma on-line, a recuperação efetiva fica difícil sem uma boa indexação e redução (problema da alta dimensionalidade) do conteúdo dos documentos. Isto é tão importante que muitos autores consideram a seleção de atributos como uma fase do processo de classificação [Moulinier, 1996].

No próximo capítulo, veremos com detalhes como se dá esta etapa e serão mostradas algumas técnicas bastante usadas.

3. Técnicas de Seleção de Atributos

O problema de classificar documentos de texto tem grande importância prática dado o massivo volume de textos on-line disponíveis através da Web, feeds de notícias, e-mails, banco de dados corporativos, registros médicos de pacientes e bibliotecas digitais. Procurar algo de valor nessa imensa coleção requer organização. Muito do trabalho de organização pode ser automatizado através de sistemas de classificação de texto. A precisão e o nosso conhecimento desses sistemas influencia fortemente sua utilidade.

Para possibilitar o trabalho destes sistemas os documentos necessitam ser transformados em um formato adequado, tais como uma tabela atributo-valor para serem submetidos a algoritmos de aprendizado. No entanto, a representação de documentos no formato atributo-valor é caracterizada pela alta dimensionalidade e por dados bastante esparsos, sendo fundamental que essa etapa seja realizada com devido cuidado para obter um bom desempenho do processo de classificação.

Diferente da recuperação de texto, na classificação de texto a alta dimensionalidade do espaço de termos (isto é, o número de termos na base, representado por |T|) pode ser problemático. De fato, enquanto algoritmos usados na recuperação de texto (como cálculo do cosseno) podem trabalhar com altos valores de |T|, o mesmo não ocorre com muitos dos sofisticados algoritmos usados na classificação. Por causa disso, antes de executar a classificação freqüentemente se aplica um passo de redução de dimensionalidade cujo efeito é reduzir o vetor-espaço de termos.

Este capítulo discute o problema da Redução da Dimensionalidade (RD) de representações de texto utilizando técnicas de seleção de atributos.

Inicialmente, abordaremos o grau de importância dos termos em um texto, algo bastante pertinente para a classificação. Em seguida, é discutido sobre a necessidade da preparação do texto para que ele possa ser submetido aos algoritmos de seleção e classificação. Por fim são apresentadas diversas técnicas de eliminação de características.

3.1. A relevância de um Termo no Texto

Esta seção está baseada em um estudo feito por Zipf [Zipf, 1949] e outros autores [Booth, 1967][Goffman,1966][Luhn, 1958][Rijsbergen, 1979].

Zipf identificou que, num texto suficientemente extenso, existia uma relação entre a freqüência que uma dada palavra ocorria, e sua posição na lista de palavras ordenadas segundo sua freqüência de ocorrência [Guedes, 1998]. Essa lista era produzida, levando-se em conta a freqüência decrescente de ocorrências. À posição nesta lista dá-se o nome de ordem de série (rank). Assim, a palavra de maior freqüência de ocorrência tem ordem de série 1, a de segunda maior freqüência de ocorrência, ordem de serie 2, e assim sucessivamente. Analisando a distribuição das palavras em várias listas, ele percebeu que existia um pequeno número de palavras que ocorrem muito freqüentemente, enquanto a maioria das palavras ocorre com pouca freqüência. Zipf também notou que se multiplicasse o rank (R) de um termo com sua frequência (F) de ocorrência resultaria em um valor aproximadamente constante (C). Então ele elaborou a primeira lei que afirma que:

R . F = C

Contudo, essa fórmula só é aplicável a palavras com alta frequência de ocorrência em um texto.

[Goffman, 1966] observou que a Primeira Lei de Zipf era válida apenas para um subconjunto de palavras muito frequentes. Na maioria dos casos, essas palavras têm a característica de ocupar ranking único na lista de distribuição de palavras, isto é, das palavras de alta freqüência de ocorrência, dificilmente, existem duas palavras com a mesma freqüência de ocorrência.

Para palavras de baixa freqüência de ocorrência, Zipf propôs uma segunda lei, revisada e modificada por [Booth, 1967]. A Segunda Lei de Zipf enuncia que, em um determinado texto, várias palavras de baixa freqüência de ocorrência (alto rank) têm a mesma freqüência [Guedes, 1998].

Esses dois comportamentos, inteiramente distintos, definem as duas extremidades da lista de distribuição de palavras de um dado texto. Assim, é razoável esperar uma região crítica, na qual ocorre a transição do comportamento das palavras de alta freqüência para as de baixa freqüência. [Goffman, 1966] admitiu como hipótese que

nessa região de transição estariam as palavras de maior conteúdo semântico de um dado texto (Figura 3.1)[Lunh, 1958].

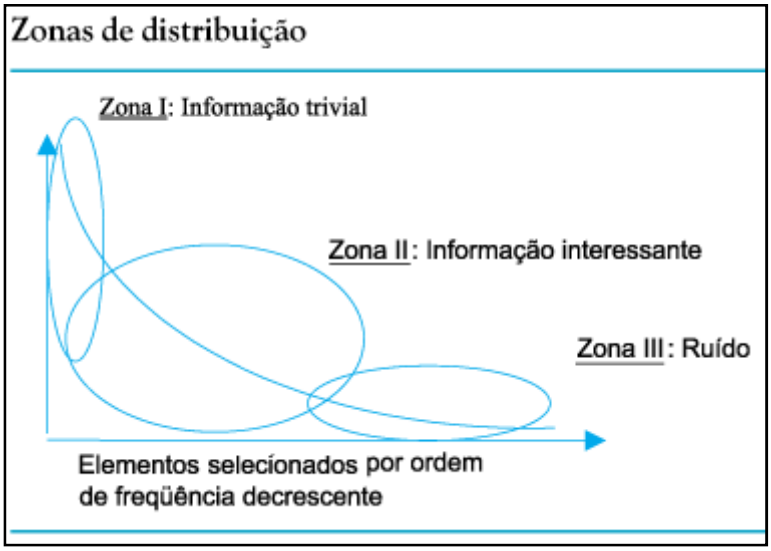

Figura 3.1 Distribuição de Zipf

[Luhn, 1958] usou a lei de Zipf como uma hipótese nula para especificar dois pontos de cortes: superior e inferior. Os termos que excedem o corte superior são os mais freqüentes e são considerados comuns por aparecerem em qualquer tipo de documento, como as preposições, conjunções e artigos. Já os termos abaixo do corte inferior são considerados raros e, portanto, não contribuem muito na discriminação dos documentos.

Assim, [Luhn, 1958] propôs uma técnica para encontrar termos relevantes, assumindo que os termos mais significativos para discriminar o conteúdo de um documento estão em um pico imaginário posicionado no meio dos dois pontos de corte. Porém, uma certa arbitrariedade está envolvida na determinação dos pontos de corte, bem como na curva imaginária, os quais são estabelecidos por tentativa e erro [Rijsbergen, 1979].

Essa discussão é central para a atividade de seleção de atributos, pois tenta dar fundamentação conceitual para essa tarefa.

A seguir veremos como os documentos devem ser preparados para que possam ser trabalhados por sistemas classificadores.

3.2. Pré-processamento dos Documentos

Textos não podem ser diretamente interpretados por um classificador ou por um algoritmo de classificação. Por causa disso, é necessário um processo de indexação que mapeia o texto de um documento em uma representação compacta de seus conteúdos. Assim eles podem ser uniformemente utilizados, seja um documento de treinamento ou de teste (etapa não trivial e bastante custosa, devido à forma não estruturada dos documentos).

O primeiro passo é fazer a análise léxica do texto para remover tudo aquilo que não vai ser considerado como termo. É preciso remover as pontuações (!,.,?,...,: e etc), os símbolos (@,#,$,%,&,* e etc), e os números (não têm valor semântico). Isto é o básico para que texto possa ser representado na forma de termos de maneira correta.

Em Classificação de Texto, normalmente o texto é representado como um vetor de termos (palavras que ocorrem no texto pelo menos uma vez) com seus pesos, seguindo a abordagem bag-of-words. Existem outras representações mais sofisticadas como frases ou sentenças, entretanto, resultados experimentais mostraram que representações mais sofisticadas perdem em desempenho com relação à abordagem bag-of-words [Apté et al., 1994, Lewis, 1994]. A abordagem bag-of-words é acessível à maioria dos algoritmos de aprendizado, os quais requerem que cada exemplo seja descrito como um vetor de dimensionalidade fixa.

Na abordagem bag-of-words, cada elemento do vetor representa o peso de um atributo (termo) do documento. O valor de um atributo, ou termo, tj, j = 1, 2, ..., m, para um documento di, i = 1, 2, . . . , n, pode ser um valor booleano, indicando se o termo aparece ou não no documento, o número de vezes que o termo aparece no documento, entre outros.

Muitas vezes os pesos são calculados com a função padrão TF-IDF (Term Frequency - Inverse Document Frequency) (Ver Figura 3.2). Essa função expressa a idéia de que: (1) quanto maior a freqüência de um termo num documento mais ele é representativo para o conteúdo, e (2) e quanto mais documentos contiverem um termo, menos discriminante ele é.

$$tf(t_k, d_j) = \#(t_k, d_j)$$

$$idf(t_k, d_j) = \log \frac{|Tr|}{\#Tr(t_k)}$$

$$tfidf(t_k, d_j) = tf(t_k, d_j) idf(t_k, d_j)$$

Figura 3.2 Cálculo comum do TF-IDF

Nota-se que essa fórmula computa a importância do termo para um documento considerando apenas as suas ocorrências. Em outras palavras, a semântica do documento é reduzida para a semântica léxica dos termos que nele ocorrem. Para deixar os pesos num intervalo [0,1], e para considerar os documentos como vetores de tamanho igual, os pesos resultantes do TF-IDF são freqüentemente normalizados pela norma dos pesos de todos os termos do corpus (Figura 3.3):

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|V|} (tfidf(t_s, d_j))^2}}$$

Figura 3.3 Cálculo normalizado do TF-IDF

Embora o TF-IDF seja o mais popular, outras funções de cálculo de peso costumam ser usadas, incluindo técnicas probabilísticas.

3.3. Redução da Dimensionalidade pela Seleção de Termos

As técnicas de seleção de características são usadas para identificar quais termos são mais discriminantes para os algoritmos de classificação. As duas maiores motivações para realizar essa etapa de redução de dimensionalidade são: a precisão dos algoritmos de classificação e a escalabilidade da técnica (aprendizagem).

A precisão de muitos algoritmos de aprendizagem pode ser otimizada pela seleção dos termos mais preditivos. Por exemplo, o Naive Bayes tende a ter um baixo desempenho sem seleção de características. O propósito da seleção, às vezes, é descrito como a necessidade de eliminar os termos irrelevantes e que geram ruído para a classificação, mas estudos mostram que mesmo palavras com baixas pontuações ainda possuem valor para a predição [Joachims, 1998]. Apenas um pequeno conjunto de palavras tem alta probabilidade de ocorrer para cada categoria. Objetivamente, a razão de selecionar qualquer subconjunto de características é produzir um classificador de maior precisão.

Como dito, outra motivação para a redução de dimensionalidade é a escalabilidade. Um grande corpus de textos pode facilmente ter dezenas para milhares de palavras distintas. Selecionando apenas uma fração do vocabulário como entrada, consegue-se diminuir a computação do processo de classificação. Isto também diminui a demanda por armazenamento (sistemas de arquivo ou Banco de Dados) e largura de banda (Internet).

Métodos de seleção de termos tentam selecionar, do conjunto original T, o subconjunto T' de termos que quando usados na indexação dos documentos possibilitam uma maior eficiência. [Yang, 1997] mostrou que a redução da dimensionalidade pela seleção de termos podem resultar num aumento moderado da eficiência, dependendo do classificador, do nível da redução e da técnica utilizada.

A redução de dimensionalidade é também pode reduzir o *Overfitting*, isto é, o fenômeno pelo qual um classificador é tão otimizado que características particulares dos dados de treino tornam-se relevantes para as categorias. Ou seja, o classificador é ajustado de forma muito específica para o conjunto de treinamento, implicando em um baixo desempenho na classificação de documentos não conhecidos pelo classificador.

Entretanto, na remoção de termos, há o risco de remover informações potencialmente úteis do significado dos documentos. Conseqüentemente, o processo de redução deve ser executado com cuidado para se obter o ponto ótimo de custo-efetividade.

A seguir veremos algumas das técnicas mais utilizadas na redução de dimensionalidade em textos.

3.3.1 StopWords

"Stopwords" são palavras consideradas não relevantes para a análise de textos ou para a busca. Na maioria das vezes, são palavras auxiliares ou conectivas, não fornecendo nenhuma informação discriminativa na expressão do conteúdo do texto. Palavras como pronomes, artigos, preposições e conjunções podem ser consideradas "stopwords". A lista de "stopwords" é conhecida como Stoplist.

A remoção das "stopwords" da representação dos documentos otimiza a performance e a precisão dos algoritmos de classificação. Esta operação também pode ser considerada como uma técnica de compressão de textos, pois a eliminação de "stopwords" reduz o número de palavras a serem analisadas no documento, e também o número de palavras a serem armazenadas em uma base de dados.

3.3.2 Thesaurus

Um thesaurus é uma compilação de palavras indexadas com significados similares (sinônimos), relacionados e opostos (antônimos). Um thesaurus não define os significados dos termos, isto é papel de um dicionário.

Um thesaurus pode ser útil para sistemas de classificação de documentos de textos. Ele deve ser aplicado na fase de redução dos termos. Palavras que possuem sinônimos podem ser expressas de uma só maneira. Por exemplo, as palavras "bonito", "belo", "lindo" são sinônimas. Com o objetivo de reduzir esses termos no corpus, podemos usar só a palavra "belo" onde ocorrerem as palavras "bonito" ou "lindo".

Em Recuperação da Informação, essa técnica também pode ser aplicada para realizar expansões em consultas levando em conta as palavras com mesmos significados.

3.3.3 Stemming

Em Processamento de Linguagem Natural (PLN), existe uma categoria de técnicas denominada “normalização de variações lingüísticas”, que permite transformar palavras em elementos mais simples. O Stemming é uma técnica de normalização que produz as flexões de uma palavra a uma representação comum, o “radical” (stem). Essa transformação consiste na remoção de sufixos das palavras, ou na transformação de verbos para sua forma no infinitivo (identificam-se as diferentes inflexões de uma mesma palavra).

O “stem” não é necessariamente igual à raiz lingüística (radical), mas permite identificar variações de uma palavra. Por exemplo, “machine” e “machines” são essencialmente iguais, mas sem sofrerem a redução por Stemming serão tratadas como palavras distintas.

Um dos algoritmos de Stemming mais conhecidos é o algoritmo de Porter [Porter, 1980]. O método de Stemming de Porter remove até 60 sufixos diferentes. Este algoritmo leva em conta as classes morfológicas das palavras, executando uma série de passos de remoção de sufixos conhecidos. Os passos são aplicados na seguinte seqüência:

1. Redução do plural;
2. Redução do feminino;
3. Redução do advérbio;
4. Redução do aumentativo e do diminutivo;
5. Redução das formas nominais;
6. Redução das terminações verbais;
7. Redução da vogal temática;
8. Remoção dos acentos.

Cada passo utiliza um conjunto de regras. As regras dentro de um passo são examinadas em seqüência, e somente uma das regras pode ser aplicada. O possível sufixo mais longo é sempre removido primeiro por causa da ordem das regras dentro de um passo. Por exemplo, o sufixo de plural - “es" deve ser testado antes do sufixo – “s”.

3.3.4 Redução para Algoritmos de Aprendizagem

Várias técnicas de redução de dimensionalidade (RD) têm sido propostas, umas vindas da Teoria da Informação outras vindas da literatura da Álgebra Linear, e suas contribuições têm sido testadas experimentalmente, avaliando a variação da efetividade que um dado classificador alcança.

Existem duas maneiras de efetuar RD [Sebastiani, 1999]: Redução Local (realizada no conjunto de termos que ocorrem em cada categoria), ou Redução Global (realizada na base inteira).

Veremos a seguir técnicas de ambas as abordagens de redução de termos: redução global (Document Frequency) e redução local (Information Gain, Mutual Information e X^2 Estatistic).

Document Frequency

Uma das mais simples e efetivas técnicas de redução de dimensionalidade na abordagem global que consiste em contabilizar a quantidade de documentos que um dado termo ocorre. Segundo essa técnica, os termos mais raros devem ser removidos da representação dos documentos. Experimentos de [Yang, 1997] mostraram que é possível fazer uma grande redução de termos sem perda de eficiência do classificador utilizado.

O método parece indicar que termos que ocorrem com maior freqüência na coleção de documentos são mais relevantes para a classificação de texto. A idéia dessa técnica é medir a freqüência de todos termos DF, e se a freqüência for menor que um determinado limite o termo deve ser removido do vocabulário do corpus. A suposição básica é a de que termos raros são pouco informativos para predizerem a categoria ou não influenciam no desempenho total.

Isto parece contradizer uma “lei” bem conhecida da Recuperação de Informação que afirma que termos de baixa à média freqüência são mais informativos [Salton & Buckley, 1988]. Mas uma afirmação não contradiz a outra, dado que a grande maioria das palavras que ocorrem no corpus têm baixíssima freqüência, então uma grande redução removeria muitos termos, mas somente aqueles de menor freqüência.

Além disso, essa técnica é facilmente escalável para conjuntos com muitos documentos, com uma complexidade computacional praticamente linear em relação à quantidade de documentos.

Entretanto, esta é considerada uma técnica “ad hoc” para melhorar a eficiência, não um critério baseado em princípios para a seleção preditiva de características.

Information Gain

Ganho de informação é amplamente empregado como um critério de avaliar a importância de termos na área de aprendizagem de máquina. A “quantidade de informação” relativa à predição da categoria pela presença ou ausência de um termo em um documento é medida. Dado um conjunto de categorias C = {c_1, c_2, c_3, ..., c_n}, o ganho de informação de um termo t é definido como (Figura 3.4):

$$
\begin{aligned}
G(t_k) = & -\sum_{i=1}^{|C|} P(c_i)\log P(c_i) \\
& + P(t_k)\sum_{i=1}^{|C|} P(c_i \mid t_k)\log P(c_i \mid t_k) \\
& + P(\overline{t_k})\sum_{i=1}^{|C|} P(c_i \mid \overline{t_k})\log P(c_i \mid \overline{t_k})
\end{aligned}
$$

Figura 3.4 Cálculo do ganho da informação

Esta definição é mais genérica do que a usada em modelos de classificação binária. Usa-se essa forma mais geral porque, nos problemas de classificação de texto, normalmente existem mais de duas categorias. Sendo necessário medir a importância do termo com respeito a todas as categorias.

Dado um conjunto de documentos de treinamento, o ganho de informação é calculado para cada termo, e são removidos do corpus os termos que possuírem um valor inferior a um limiar pré-determinado. O processamento da técnica inclui o cálculo das probabilidades condicionais de uma categoria, dado um termo, e a computação da entropia.

O ganho de informação é igual ao total da entropia para um atributo, se para cada valor do atributo, uma única classificação for obtida. Neste caso, a entropia relativa subtraída do total da entropia é 0.

Embora essa técnica seja uma boa medida para decidir a relevância de um termo, ela não é perfeita. Um problema observável ocorre quando o ganho da informação é aplicado para atributos que possam assumir um grande número de valores distintos. Por exemplo, supõe-se que se está construindo uma árvore de decisão para os dados que descrevem os negócios de um cliente. O ganho de informação é freqüentemente usado para decidir quais características são mais relevantes, então eles serão testados próximos à raiz da árvore. Uma das características pode ser o número do cartão de crédito do cliente. Este atributo tem alto valor porque ele identifica unicamente cada cliente, contudo, este não é um bom atributo para se colocar na árvore de decisão: decidir como tratar o cliente baseado em seu número do cartão é inviável para generalizar os clientes que não foram vistos.

Mutual Information

Dado um termo t_k e uma categoria ci, considere A o número de vezes que t_k e c_i co-ocorrem; B o número de vezes que t_k ocorre sem ci; C o número de vezes que ci ocorre sem t_k; e N o número total de documentos de treinamento. A medida de informação mútua é definida como (Figura 3.5):

$$I(t_k, c_i) = \frac{P(t_k \wedge c_i)}{P(t_k)P(c_i)}$$

Figura 3.5 Definição da informação mútua

Estima-se a medida da informação mútua usando-se a equação na Figura 3.6:

$$I(t_k, c_i) \approx \frac{\log A \times N}{(A+C)(A+B)}$$

Figura 3.6 Estimativa da informação mútua

A medida MI(t_k, c_i) tem o valor de zero caso t_k e c_i sejam independentes. Para medir a importância de um termo globalmente, combinam-se as pontuações de um termo específicas para cada categoria em duas formas alternativas (Figura 3.7 e 3.8):

$$I_{avg}(t_k) = \sum_{i=1}^{|C|} P(c_i) I(t_k, c_i)$$

Figura 3.7 MI global: média dos valores

$$I_{\max}(t_k) = \max_{i=1}^{|C|} \{I(t_k, c_i)\}$$

Figura 3.8 MI global: máximo dos valores

Desta forma, para cada termo, é calculada a informação mútua, e são removidos do corpus os termos que possuírem um valor inferior a um limiar predeterminado.

X² Estatistic

Esta técnica mede o grau de dependência entre um termo t e uma categoria c.

$$\chi^2(t_k, c_i) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)}$$

Figura 3.9 Cálculo da estatística X²

Segundo a fórmula acima: “A” representa o número de documentos em que t e c co-ocorrem; B o número de documentos em que t ocorre sem c; C o número de documentos em que c ocorre sem t; D o número de documentos em que nem t e nem c

ocorrem; e, por último N representa o número total de documentos que compõem o conjunto de treino.

Esta fórmula resulta em zero se t e c são independentes. Calcula-se, para cada categoria, o valor do X² entre cada termo único do corpus de treinamento e uma categoria. Para obter a importância global do termo, combinam-se os valores específicos de um termo nas diversas categorias em dois possíveis cálculos (Figura 3.10 e 3.11):

$$\chi^2_{\max}(t_k) = \max_{i=1}^{|C|} \left\{ \chi^2(t_k, c_i) \right\}$$

Figura 3.10 X² global: máximo dos valores

$$\chi^2(t_k) = \sum_{i=1}^{|C|} P(c_i)\chi^2(t_k, c_i)$$

Figura 3.11 X² global: média dos valores

A computação dessa técnica tem complexidade quadrática, similar à Mutual Information e ao Information Gain. Maior diferença entre MI e CHI é que a segunda tem o valor normalizado, então os valores são comparáveis entre os termos de uma mesma categoria. Entretanto, essa normalização é ruim para termos de baixa freqüência.

Quadro das principais técnicas de seleção de termos

A seguir, no Quadro 3.1 [Sebastiani, 1999], encontra-se variações das técnicas já discutidas e outras também empregadas para seleção de termos.

Function	Denoted by	Mathematical form
DIA association factor	$z(t_k, c_i)$	$P(c_i \mid t_k)$
Information gain	$IG(t_k, c_i)$	$\sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)}$
Mutual information	$MI(t_k, c_i)$	$\log \frac{P(t_k, c_i)}{P(t_k) \cdot P(c_i)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{\lvert Tr \rvert \cdot [P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)]^2}{P(t_k) \cdot P(\bar{t}_k) \cdot P(c_i) \cdot P(\bar{c}_i)}$
NGL coefficient	$NGL(t_k, c_i)$	$\frac{\sqrt{\lvert Tr \rvert} \cdot [P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)]}{\sqrt{P(t_k) \cdot P(\bar{t}_k) \cdot P(c_i) \cdot P(\bar{c}_i)}}$
Relevancy score	$RS(t_k, c_i)$	$\log \frac{P(t_k \mid c_i) + d}{P(\bar{t}_k \mid \bar{c}_i) + d}$
Odds ratio	$OR(t_k, c_i)$	$\frac{P(t_k \mid c_i) \cdot (1 - P(t_k \mid \bar{c}_i))}{(1 - P(t_k \mid c_i)) \cdot P(t_k \mid \bar{c}_i)}$
GSS coefficient	$GSS(t_k, c_i)$	$P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)$

Quadro 3.1Principais técnicas de seleção [Sebastiani, 1999]

3.4. Considerações Finais

Como visto, muitos autores [Zipf, 1949][Booth, 1967][Goffman,1966][Luhn, 1958][Rijsbergen, 1979] vêm trabalhando em como se caracteriza a importância de um termo em um texto, isso é vital para a tarefa de classificação de documentos. É por esse princípio que os algoritmos de seleção de atributos funcionam (tentam separar os termos bons dos termos irrelevantes).

Existem diversas técnicas para realizar a redução de atributos da representação dos documentos, provindas de diferentes áreas de estudo e fundamentadas em conceitos bem específicos. Mas um ponto é comum a todas técnicas, antes de processarem efetivamente os termos no texto, é preciso deixá-lo em uma representação uniforme (por exemplo, bag-of-words).

4. O Trabalho Desenvolvido

Considerando a importância e aplicabilidade da Classificação de Texto, assim como o problema da alta dimensionalidade de características, é proposta nesse trabalho uma aplicação que possibilite a análise comparativa das principais técnicas de seleção de termos baseadas em aprendizagem de máquina, segundo [Yang, 1997].

Em seu artigo, [Yang, 1997] levantou que os melhores métodos para seus experimentos foram: Document Frequency, Information Gain e Estatistic X^2. Os resultados obtidos com essa pesquisa levou a focar nesses métodos neste TG. A idéia é ir variando o nível da redução de termos com um dada técnica, e analisar a precisão obtida pelo classificador. Obviamente, para avaliar a otimização alcançada pelas técnicas, é necessário o desenvolvimento de classificadores. Os classificadores implementados aqui foram o KNN e o Naive Bayes, por serem de eficiência comprovada e largamente empregados em sistemas de classificação de texto.

A seguir é detalhado o sistema desenvolvido. Na seção 4.1 apresentamos as etapas do processo, mostrando o que foi definido a nível de projeto. Na seção 4.2 apresentamos todas as classes e seus métodos, especificando como o sistema foi feito a nível de implementação. Na seção 4.3 resumimos como se dá o fluxo de atividades do sistema.

4.1. O Sistema

A implementação foi inspirada nas etapas de cada uma das fases (treinamento e uso) de um sistema de classificação expostos nas Figuras 4.1 e 4.2:

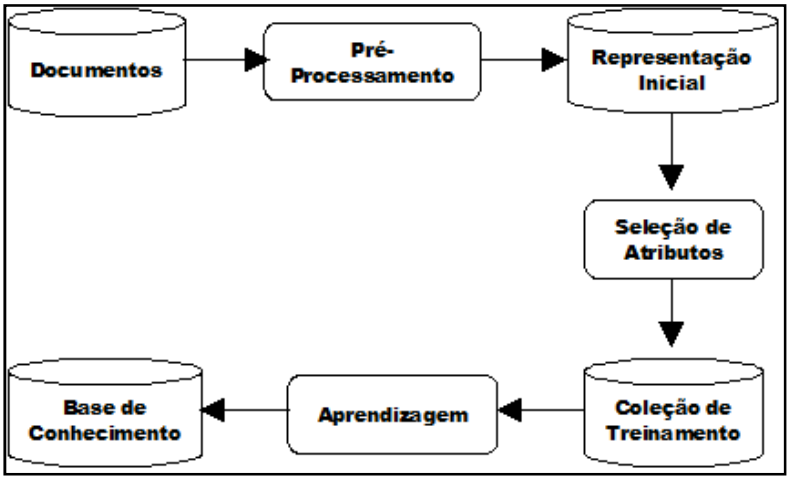

Figura 4.1 Fase de Treinamento

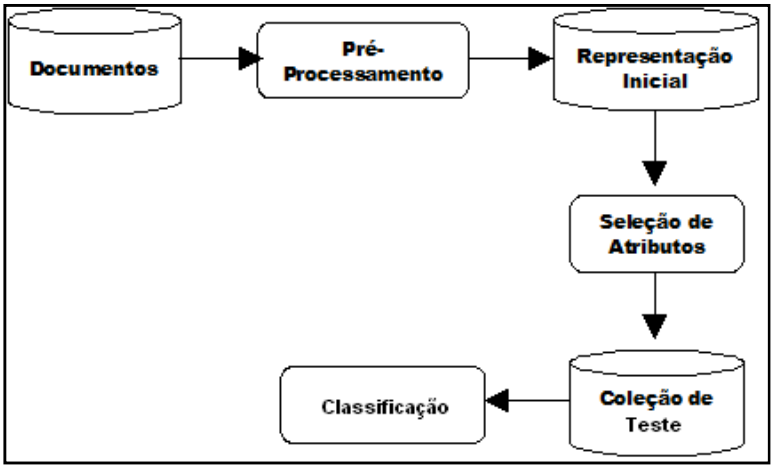

Figura 4.2 Fase de Uso

A seguir serão apresentados os pacotes definidos. Pela modelagem da aplicação, os pacotes representam as etapas (exceto o pacote “Base”) do processo de classificação. Os diagramas que serão apresentados seguem o padrão UML 2.0.

4.1.1 Base

Este pacote contém as classes principais da aplicação. Elas definem o “modelo de negócio” do sistema, ou seja, uma abstração das entidades básicas para a atividade de classificação. O foco destas classes não é realizar tarefas, mas sim guardar o estado atual dos objetos ao longo do fluxo de classificação. Todas as outras classes dos outros pacotes manipulam esses objetos.

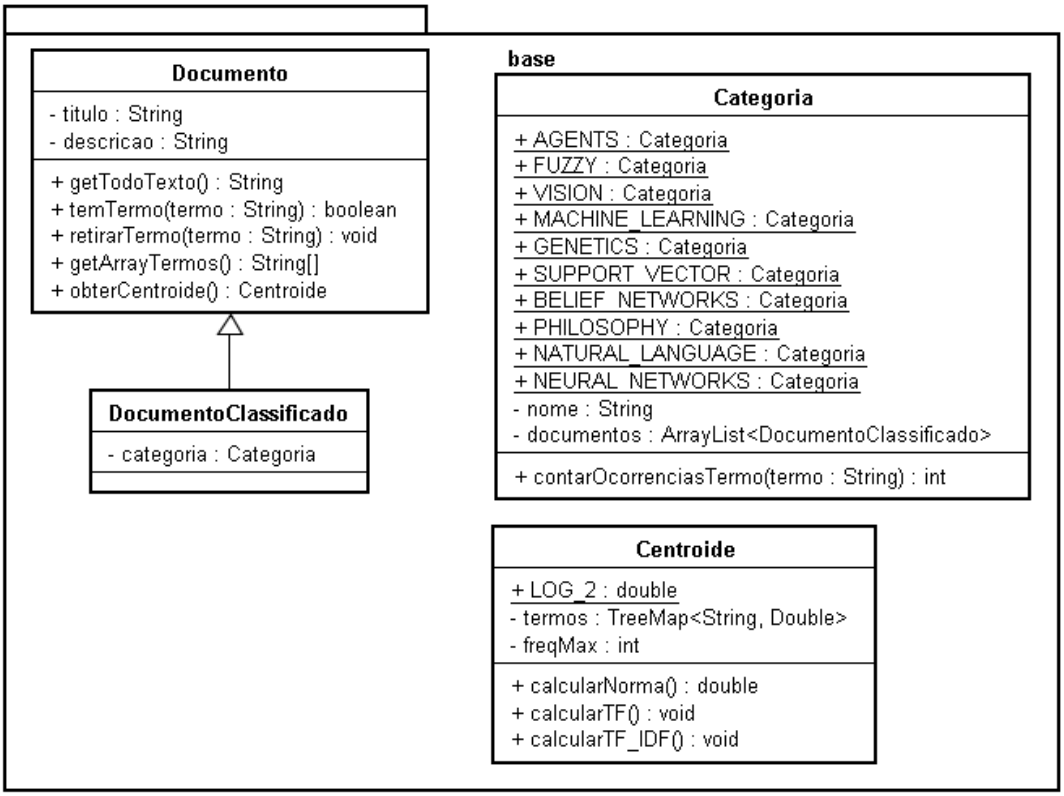

Figura 4.3 Pacote Base

4.1.2 Aquisição

Representa a primeira etapa do processo de classificação, que é responsável pela leitura dos documentos (as entradas do sistema) que estão armazenados em dez arquivos *.txt*. Nessa etapa os documentos são alocados ou para o grupo de teste ou para os grupos de treinamento.

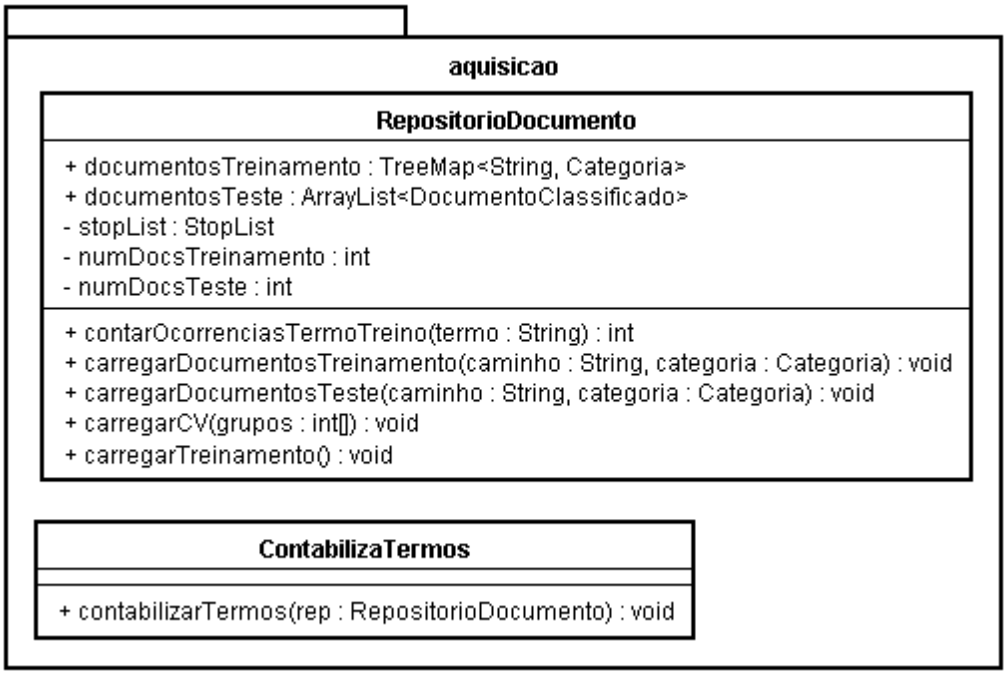

Figura 4.4 Pacote Aquisição

4.1.3 Preparação

É nessa etapa que é feita a análise léxica dos documentos (treinamento e teste) e a primeira redução de características (por StopList). Obviamente, o StopList poderia estar na fase de seleção, mas nesta aplicação, consideramos a análise léxica e o StopList como passos obrigatórios, diferente dos outros algoritmos de seleção que são de uso

opcional (para comparação de resultados entre eles). Isso explica a separação entre as etapas de Preparação e Seleção. Ver Figura 4.5 e Figura 4.6.

A análise léxica é um processo imprescindível, sem ela todo o fluxo restante do sistema seria comprometido. É ela que “limpa” o texto e permite que ele seja particionado em termos da forma correta.

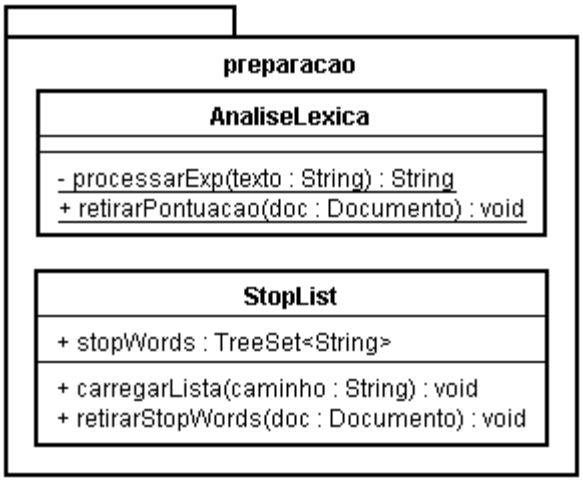

Figura 4.5 Pacote Preparação

4.1.4 Seleção

Representa a etapa de seleção dos termos por um dos algoritmos de aprendizagem de máquina escolhidos no momento da execução da aplicação. É aqui que os atributos dos documentos de treino são avaliados, separados em irrelevantes e relevantes, e por último, os termos irrelevantes são eliminados do conjunto de treinamento e teste. Essa etapa é o foco deste trabalho. Baseado na configuração do *Cross Validation*, na parametrização do algoritmo selecionador e do classificador, e da precisão do classificador, cada uma das técnicas pode ser analisada.

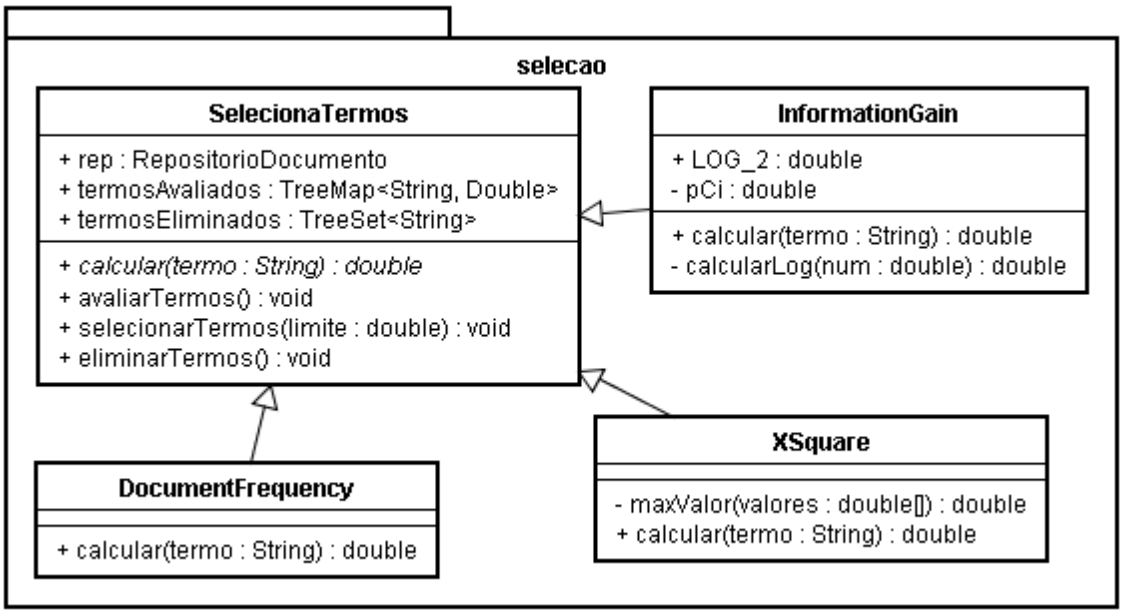

Figura 4.6 Pacote Seleção

4.1.5 Classificação

Depois das etapas de “Preparação” e “Seleção”, que reduzem a quantidade de termos irrelevantes e melhoram a qualidade dos documentos, pode-se finalmente prosseguir para a etapa de classificação, que terá sua precisão otimizada e será menos custosa computacionalmente. Ao fim do processo, é contabilizada a quantidade de documentos classificados corretamente.

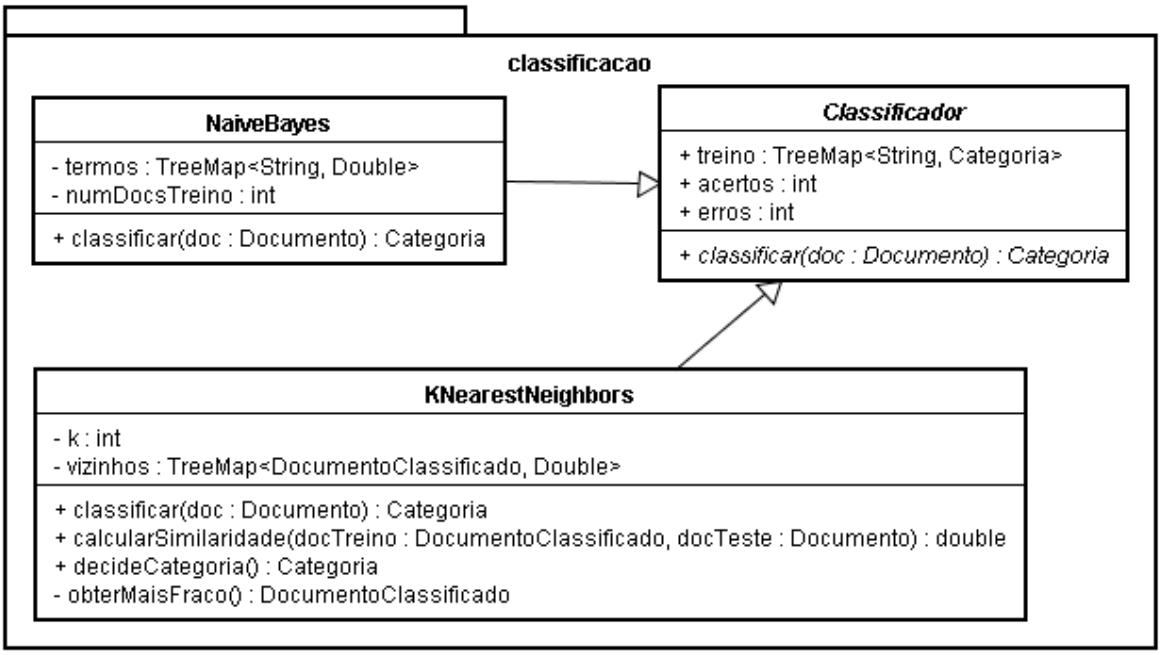

Figura 4.7 Pacote Classificação

4.1.6 Programa

Pacote que contém a classe que executa toda a aplicação. Ela executa todas as etapas do sistema de forma seqüencial, é aqui que todo o processo é parametrizado. Define-se:

- O número de divisões de grupos para o *K-Fold Cross Validation*.

- O grupo de documentos de teste.

- A técnica de seleção de termos por aprendizagem de máquina utilizada.

- A agressividade da redução das características.

- O classificador utilizado. Caso seja o KNN, configura-se o parâmetro k.

No fim da execução obtém-se:

- Número de documentos de teste.

- Número de documentos de treinamento.

- Número de termos distintos após a primeira redução (StopList).

- Número de termos distintos restantes após a segunda redução (seleção).

- Precisão do classificador.

- Tempo total de execução em segundos.

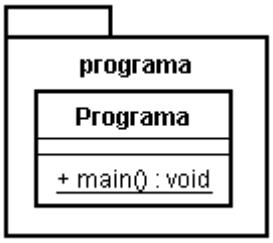

Figura 4.8 Pacote Programa

4.2. A Implementação

O programa foi desenvolvido usando a linguagem Java, versão 5.0. A seguir serão detalhadas as classes com suas variáveis membros e seus métodos implementados. As classes Java são componentes que possuem propriedades e desempenham tarefas no decorrer do processo. Os métodos “getters” e “setters” das classes não serão mostrados e nem descritos, por questões de conveniência e simplicidade.

Documento

Representa a principal classe do sistema. Ela define um Web Snippet não classificado, composto somente com título e descrição. O conjunto destes documentos formam o conjunto de teste e treinamento. Seus métodos são:

- getTodoTexto: Retorna na forma de uma só String a junção do título com a descrição (pela suposição que os dois elementos tem a mesma relevância).
- temTermo: Verifica se tem o termo passado como parâmetro em todo o texto.
- retirarTermo: Remove todas as ocorrências de um dado termo no texto caso ele esteja presente.
- getArrayTermos: Retorna na forma de uma seqüência de Strings todas as palavras que compõem o texto.

- obterCentroide: Converte a representação textual do documento para a forma de “bag-of-words” (centróide).

DocumentoClassificado

É uma extensão da classe Documento (tem as mesmas propriedades e funções da classe estendida). O detalhe é que esse objeto pertence a uma categoria. O conjunto destes documentos formam o conjunto de treinamento.

Centróide

Classe que define a representação atributo-valor de um documento. Simplesmente consiste de uma lista dos termos distintos com suas respectivas freqüências. Seus métodos são:

- calcularTF: Normaliza as freqüências de todos os termos pela maior freqüência obtida no centróide.
- calcularTF_IDF: calcula o TF-IDF normalizado, como descrito na figura tal, de cada termo do centróide.
- calcularNorma: A norma é calculada pela equação:

$$\|X\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \ldots}$$

Figura 4.9 Cálculo da norma

Onde “x_i” é a freqüência de cada termo no centróide. O valor da norma é influenciado pela escolha de se usar TF ou TF-IDF como peso dos termos (depende de qual dos dois métodos acima será executado antes). O valor do cálculo é usado como parte da função de similaridade do KNN.

Categoria

Classe que representa uma categoria para o sistema de classificação. Ela possui um nome e uma lista de todos os documentos que pertencem a essa categoria

(documentos de treinamento). Já que na aplicação todas as categorias são previamente conhecidas e fixas, 10 instâncias de Categoria, uma para cada um dos tópicos relacionados à Inteligência Artificial, são criadas e colocadas como constantes. A classe possui um único método:

- contarOcorrenciasTermo: Contabiliza o número de vezes que uma dada palavra aparece naquela categoria, ou seja, contabiliza entre todos os documentos pertencentes a categoria. É usado pelas técnicas de seleção para estimar seus cálculos.

RepositorioDocumento

Classe que efetivamente faz a leitura dos documentos do corpus. Ela funciona como um repositório tanto para os documentos que serão usados para treino quanto para teste, sendo o componente que alimenta a aplicação. Além disso, ela conta a quantidade de documentos usados para testes e treino. Seus métodos:

- contarOcorrenciasTermoTreino: Contabiliza o número de vezes que uma dada palavra aparece considerando o conjunto de treino, não importando a que categoria pertença. É usado pelas técnicas de seleção para estimar seus cálculos.
- carregarDocumentosTreinamento: Lê todos os documentos de um dado arquivo .txt (caminho) os colocando na categoria configurada e no conjunto de treinamento.
- carregarDocumentosTeste: Lê todos os documentos de um dado arquivo .txt (caminho) os colocando no conjunto de teste. Observa-se que uma categoria é repassada na função, isso é necessário para que no fim da classificação do conjunto de teste seja possível contabilizar o número de acertos e erros do classificador. Em outras palavras, é sabido previamente a categoria correta de cada documento de teste, mas só no fim do processo é feita a verificação da resposta do classificador.
- carregarCV: Invoca os dois métodos acima para enfim dar início a classificação. A divisão dos documentos no momento da leitura é feita na forma de *K-Fold Cross Validation*. Ou seja, fica um grupo de teste e o restante como grupos de treinamento. O parâmetro da função define a quantidade de grupos e qual será o de teste.
- carregarTreinamento: Lê todos os documentos dos arquivos como treinamento e só um como documento de teste. É um tipo de *Holdout Validation*, esta abordagem foi

somente usada para testar o correto funcionamento das etapas seguintes visto que essa forma é de processamento mais simples.

ContabilizaTermos

Classe auxiliar que simplesmente produz uma lista de todos os termos distintos com sua respectiva quantidade de ocorrências, levando em consideração todo o conjunto de treinamento, não importando a que categoria pertença. Seu método:

- contabilizaTermos: Método que produz a lista acima citada com o conjunto de treino do repositório de documentos. Essa lista é utilizada pelo Naive Bayes.

AnaliseLexica

Classe que faz o trabalho de remover caracteres inapropriados e irrelevantes do texto e organiza os termos para serem divididos devidamente. Seus métodos:

- processarExp: Função que faz qualquer String ser filtrada pelas seguintes expressões regulares:

  "[!'*=+#%:&;™.,\\/()?\"]" – Remove símbolos e pontuações.

  "[0-9]" – Remove números.

  "[0-9]th" – Remove números com terminações “th”.

  "[Z-^]" – Remove mais caracteres entre “Z” e “^” segundo a tabela ASCII.

  "--" – Remove as ocorrências de hífens duplos.

  " - | -|- " – Remove hífens quando não unem duas palavras.

  " {2,}" – Separa os termos da String por um só espaço.

- retirarPontuacao: Aplica a função descrita acima sobre o texto de um documento.

StopList

Classe que contém uma lista de termos (constando de 359) considerados certamente irrelevantes para a classificação. É responsável pela primeira redução de termos. Seus métodos:

- carregarLista: Lê o arquivo *.txt* que contém as *stopwords*.
- retirarStopWords: Método que verifica se cada termo do documento está presente na StopList, se estiver, todas as suas ocorrências no texto são removidas.

SelecionaTermos

Classe genérica que define um algoritmo de seleção de atributos por aprendizagem de máquina. Ela possui variáveis membros e métodos que serão herdados pelos algoritmos de seleção propriamente ditos. Essa estratégia possibilidade o reuso de software, que reduz o trabalho de implementação e o trabalhado de manutenção quando necessário. Seus métodos são:

- calcular: Método que calcula a importância do termo segundo o conjunto de treinamento. Na verdade esse método não possui implementação para esta classe (método abstrato), seu propósito é que cada uma das suas subclasses (DF, IG e CHI) defina como calcular a importância do termo especificamente.
- avaliarTermos: Método que itera sobre todos os termos distintos da base de treino invocando o método “calcular” (definido pela subclasse) para um. A relação termo e valor obtido é armazenada na lista de termos avaliados.
- selecionarTermos: Dado a lista de termos avaliados preenchida, o método funciona como um filtro, apenas os termos que tiveram seus valores inferiores ou igual ao dado limite serão removidos dos termos avaliados e colocados na lista de termos removidos.
- eliminarTermos: Método responsável pela segunda redução de características, executado após o método anterior. Ele itera sobre todos os documentos de treino e teste removendo todas as ocorrências dos termos contidos na lista de termos eliminados.

DocumentFrequency

Classe que é subclasse de “SelecionaTermos”. Responsável pela implementação da técnica Document Frequency, já descrita no trabalho. Seu método:

- calcular: Implementação do “calcular” de “SelecionaTermos”. Este método simplesmente contabiliza todas as ocorrências de um dado termo considerando o conjunto de treino. O DF como já comentado, não faz a distinção entre categorias, diferente das duas outras técnicas desenvolvidas.

XSquare

Classe que é subclasse de “SelecionaTermos”. Responsável pela implementação da técnica X^2 Statistic já descrita no trabalho. A forma de calcular a importância global do termo utiliza a variação que corresponde ao maior valor dos valores obtidos em cada categoria. Segundo [Yang, 1997], essa foi a melhor estimativa para seus experimentos. Seus métodos:

- maxValor: Método auxiliar usado para retornar o maior valor entre um conjunto de valores de tamanho qualquer.
- calcular: Implementação do “calcular” de “SelecionaTermos”. Primeiramente calcula o grau de dependência do termo para cada uma das categorias e usa a função acima para retornar o valor final.

Information Gain

Classe que é subclasse de “SelecionaTermos”. Responsável pela implementação da técnica Information Gain já descrita no trabalho. A fórmula da técnica possui uma parte de valor fixo (primeiro somatório - somatório das probabilidades vezes os logs das probabilidades da ocorrência de cada categoria), ou seja, não depende do termo a ser calculado, baseado nisso isolei esse valor na variável membro “pCi” assim posso reduzir a computação da expressão e deixá-la mais simples. Seus métodos:

- calcularLog: Método auxiliar que retorna o log de qualquer número na base 2.
- calcular: Implementação do “calcular” de “SelecionaTermos”. Calcula o valor da expressão de Information Gain para um dado termo.

Classificador

Classe genérica que representa um algoritmo classificador por aprendizagem de máquina. Ele recebe do repositório de documentos o conjunto de treinamento otimizado. Assim como a classe “SelecionaTermos”, suas propriedades e funções serão herdadas por suas subclasses. Seu método:

- classificar: Método que categoriza um dado documento de teste em uma das dez categorias já citadas baseado em suas características. Esse método não possui implementação para esta classe (método abstrato), seu propósito é que cada uma das suas subclasses (o KNN e o Naive Bayes) defina como classificar.

KNearestNeighbors

Classe que é subclasse de “Classificador”, responsável pela implementação do classificador KNN já descrito no trabalho. O peso do termo é dado por TF-IDF (Figura 3.2), o cálculo da similaridade entre dois documentos é dado pelo cálculo do cosseno (Figura 2.3) e a política de classificação é a decisão pela soma das similaridades. As variáveis membros da classe são o parâmetro “k” e “vizinhos”, que são os melhores vizinhos do documento de teste.

- calcularSimilaridade: Retorna a similaridade de um documento de teste com algum documento de treino.
- decideCategoria: Dado que os k melhores vizinhos foram obtidos, esse método aplica a política de classificação e retorna a melhor categoria.
- obterMaisFraco: A medida que os documentos de treino são verificados com o de teste os k melhores vizinhos são armazenadas, esse método auxiliar obtém o pior dos vizinhos e compara com o documento de treino atual, se o vizinho tiver um valor menor que o documento atual então ele é trocado.
- classificar: Implementação do “classificar” de “Classificador”. Executa os métodos acima descritos, primeiramente itera sobre todos os documentos de treino calculando a similaridade com o documento de teste sempre armazenando os k melhores vizinhos e por fim decide a categoria resultante.

NaiveBayes

Classe que é subclasse de “Classificador”, responsável pela implementação do classificador Naives Bayes multinomial, já descrito no trabalho, além disso utiliza a regra de Laplace para evitar a probabilidade nula. Tem como variáveis membros da classe uma lista de termos distintos com sua respectiva freqüência considerando a base de treinamento, assim como o número total de documentos. Esses valores serão usados para o cálculo das probabilidades necessárias.

- classificar: Implementação do “classificar” de “Classificador”. A fórmula está descrita na figura x. Calcula-se a probabilidade do documento de teste pertencer a cada uma das categorias e em seguida retorna como resultado a categoria com maior probabilidade.

4.3. Processo de Classificação da Aplicação

Na figura a seguir, é mostrado um resumo do fluxo de tarefas desempenhadas pelo sistema. As tarefas de carregar documentos, retirar pontuações e símbolos e retirar as stopwords são feitas de maneira igual para os documentos de treinamento e teste.

A partir daí, os termos dos documentos de treinamento são avaliados, selecionados e eliminados pela sua relevância de acordo com o algoritmo de seleção utilizado (seleção da fase de treinamento). O conjunto de termos eliminados produzido na fase de treinamento é utilizado para fazer a eliminação dos termos do conjunto de teste (seleção na fase de uso). Agora que todos os documentos foram aprimorados, o classificador escolhido poderá executar sua tarefa de classificação sobre os documentos de teste.

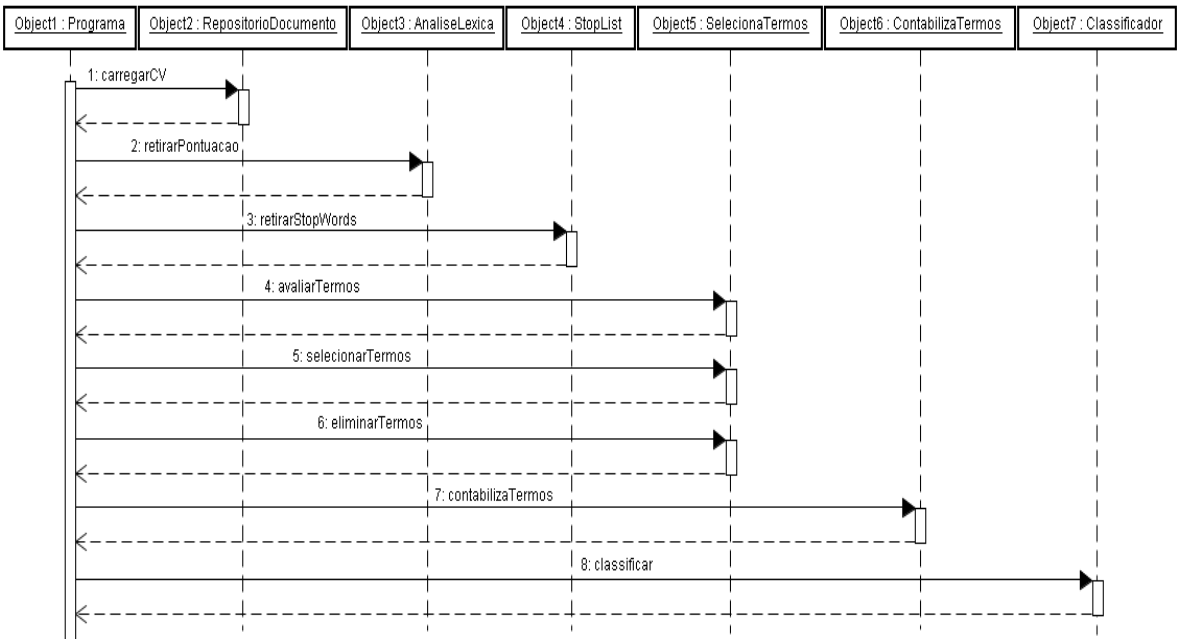

Figura 4.10 Fluxo da classificação

4.4. Considerações Finais

Depois de exposto como foi elaborado e implementado o projeto do sistema, na seção seguinte veremos como foram realizados os teste, e os resultados obtidos.

5. Testes e Resultados

Nesse capítulo é descrito o que é o corpus em questão para a aplicação, suas características gerais (quantidade de documentos, as categorias, e entre outros) e algumas amostras de documentos. Depois, é descrita a metodologia de teste, e são mostrados os gráficos gerados a partir dos resultados.

5.1. Corpus

Os documentos utilizados são os chamados Web Snippets: pequenos textos (geralmente não passam de 20 palavras, na língua inglesa) retornados por engenhos de busca como o Google, por exemplo, em resposta a pesquisas feitas pelos usuários. Normalmente o texto é formados por um título (título de uma página ou tópicos ou subtítulos destacados no texto), uma descrição (trecho do conteúdo de uma página) e um link (endereço para a página que contém o texto completo). Para este trabalho, o link não será considerado, por suas propriedades próprias e tratamento diferenciado exigido do restante do documento. Aqui, o título e a descrição terão a mesma relevância É comum considerar que o título possua termos mais discriminantes, mas devido à complexidade e ao custo para o sistema tratar esses dois elementos de maneira distinta, essa suposição foi deixada de lado.

Todas as categorias utilizadas dizem respeito à área de Inteligência Artificial, são elas: Agents; Belief Networks; Fuzzy; Genetic; Machine Learning; Natural Language; Neural Networks; Philosophy; Support Vector; Vision.

O corpus consta num total de 1009 documentos divididos da seguinte maneira nas 10 categorias:

- Agents – 77 documentos (7,63%)
- Belief Networks – 30 documentos (2,97%)
- Fuzzy – 47 documentos (4,66%)
- Genetic – 48 documentos (4,75%)
- Machine Learning – 247 documentos (24,48%)

- Natural Language – 125 documentos (12,38%)
- Neural Networks – 294 documentos (29,13%)
- Philosophy - 43 documentos (4,26%)
- Support Vector – 28 documentos (2,77%)
- Vision – 70 documentos (6,93%)

Como características gerais pode-se dizer que o corpus apesar de não ser extenso, foi o suficiente para demonstrar o propósito deste TG. O texto é resumido (poucas palavras), genérico e pouco informativo, como manchetes de jornais. Existe a predominância de documentos relativos a redes neurais e aprendizagem de máquina (ultrapassam a metade do corpus). As categorias possuem várias características em comum, visto que todas pertencem a mesma área. Contabilizamos um total de 4033 termos distintos sem nenhuma redução (somente análise léxica). A seguir, exemplos de documentos são apresentados:

Machine Learning
Título: Pattern Recognition on The Web
Descrição: Links to various pattern recognition and machine learning resources

Título: Machine Learning
Descrição: A list of links to papers and other resources on machine learning.

Título: Bibliography of ALT
Descrição: Annual Algorithmic Learning Theory conference. Since 1990. Maintained by DBLP at University of Trier.

Agents
Título: DALT 2004
Descrição: Declarative Agent Languages and Technologies workshop at AAMAS'04. New York, NY, USA; 19/20 July 2004.

Título: Software agents for business automation
Descrição: International workshop series on agent technology for business applications.

Belief Networks
Título:Cause, chance and Bayesian statistics
Descrição: Briefing document with a short survey of Bayesian statistics

Vision
Título: Edouard Duchesnay Home Pages
Descrição: Distributed Artificial Intelligence Computer Vision Image segmentation Papers, articles, thesis

Título: Rachid Deriche
Descrição: Research Director at INRIA (French National Institute for Research in Computer Science and Control).

Fuzzy
Título: FAQs
Descrição: Frequently Asked (or Answered) Questions

Título: Personal Home Pages
Descrição: Maintained by Robert Fullér.

Philosophy
Título: A Definition of AI
Descrição: A proposition for a formal definition of AI.

Natural Language
Título: Eliza
Descrição: Web-based version of this person-centered therapist emulator.

Genetic
Título: JavaEvA
Descrição: A Java based optimization framework for evolutionary algorithms.

5.2. Metodologia de Teste

Inicialmente, para validar a correta implementação dos algoritmos de seleção e classificação, usou-se a modalidade de validação Holdout Validation [Kohavi, 1995], onde só um documento era usado como teste e todos os outros faziam parte do grupo de treinamento. Dessa forma, o processo é mais simples e dá para acompanhar as modificações no documento em cada etapa detalhadamente, além, é claro, de ser computacionalmente mais rápido.

Para testar a precisão da classificação, usou-se K-Fold Cross Validation [Kohavi, 1995]. Primeiramente, decide-se em quantos grupos o corpus será dividido, depois qual o algoritmo de seleção (entre CHI, DF e IG) com o seu limite de redução, e o classificador utilizado (entre KNN e Naive Bayes). É feito o rodízio entre os grupos. A aplicação executará cada um como um grupo de teste. Ao final de cada uma dessas execuções, obtém-se o percentual de precisão. É computada a média aritmética com todas as precisões após todos os grupos serem testados. O resultado final dará a precisão média da classificação para o corpus.

Como é possível perceber pode-se fazer estas seis combinações:

1) Information Gain com KNN
2) X^2 Estatistic com KNN
3) Document Frequency com KNN
4) Information Gain com Naives Bayes
5) X^2 Estatistic com Naives Bayes
6) Document Frequency com Naives Bayes

5.3. Resultados Obtidos

A seguir, serão apresentados 4 gráficos (dois com KNN e dois com Naive Bayes) todos variando o K do Cross Validation em 6 e 8. Para o KNN, o número de melhores vizinhos foi considerado igual a 12. Os gráficos mostram a relação da média de precisão do classificador com o número de termos distintos restantes no corpus, após a segunda etapa de eliminação de termos, com cada uma das técnicas implementadas.

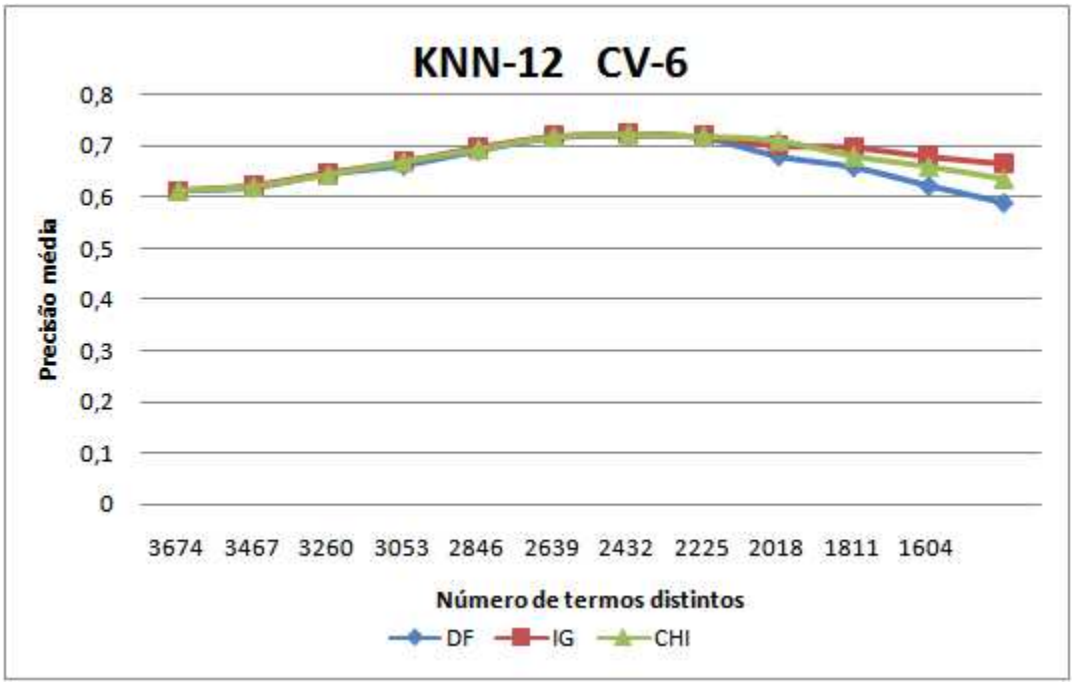

Figura 5.1 KNN com k=12 e Cross Validation com 6 grupos

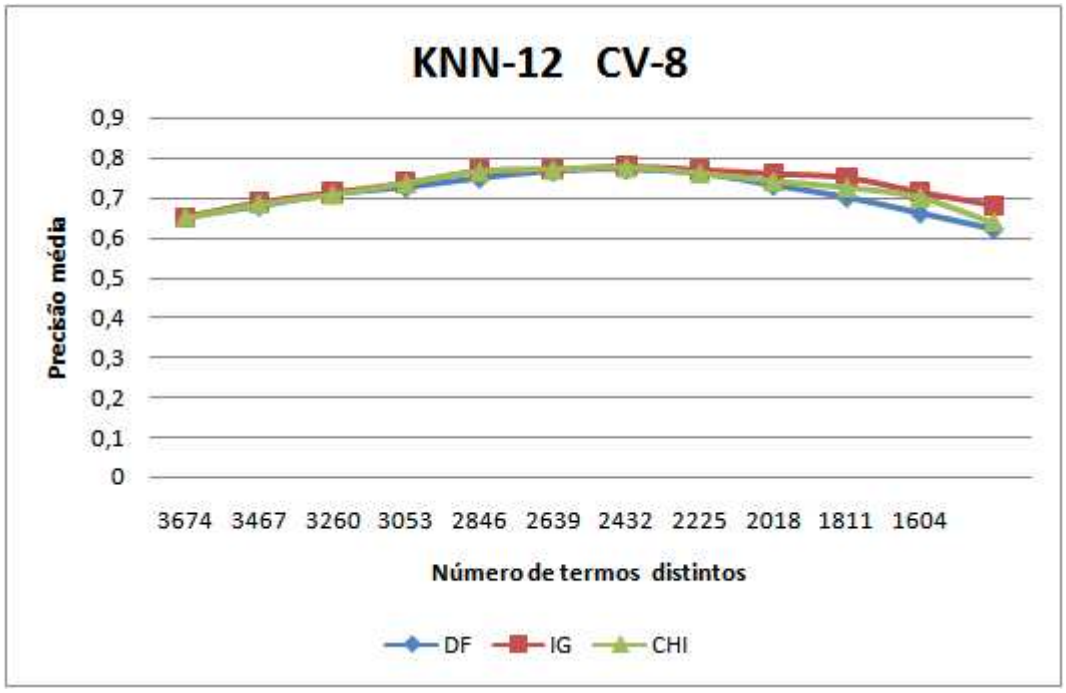

Figura 5.2 KNN com k=12 e Cross Validation com 8 grupos

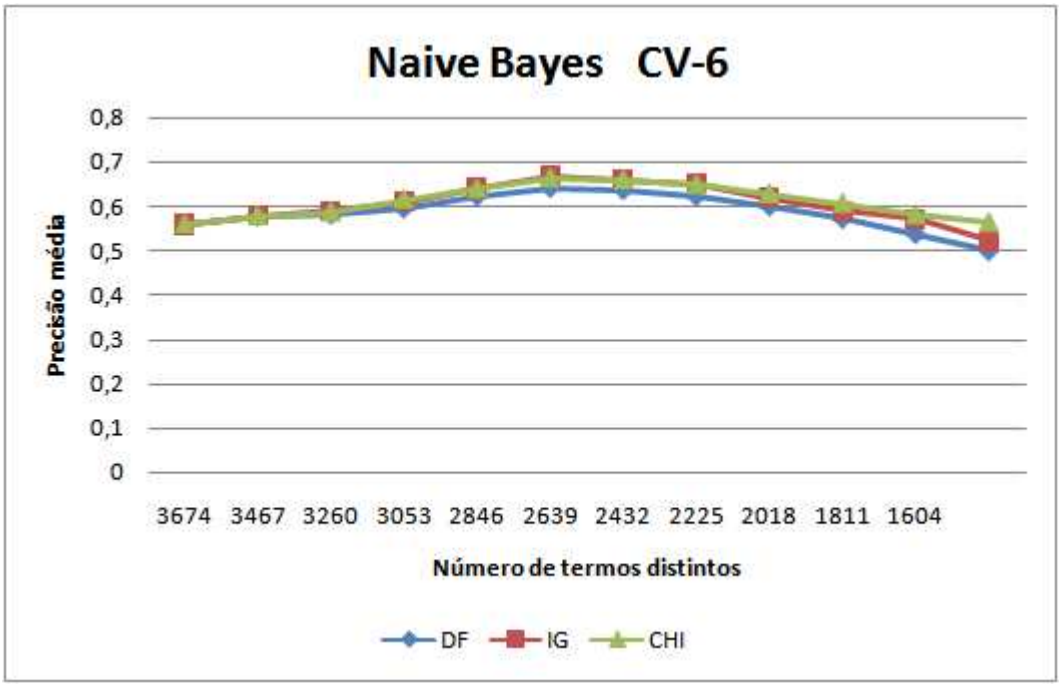

Figura 5.3 Naive Bayes e Cross Validation com 6 grupos

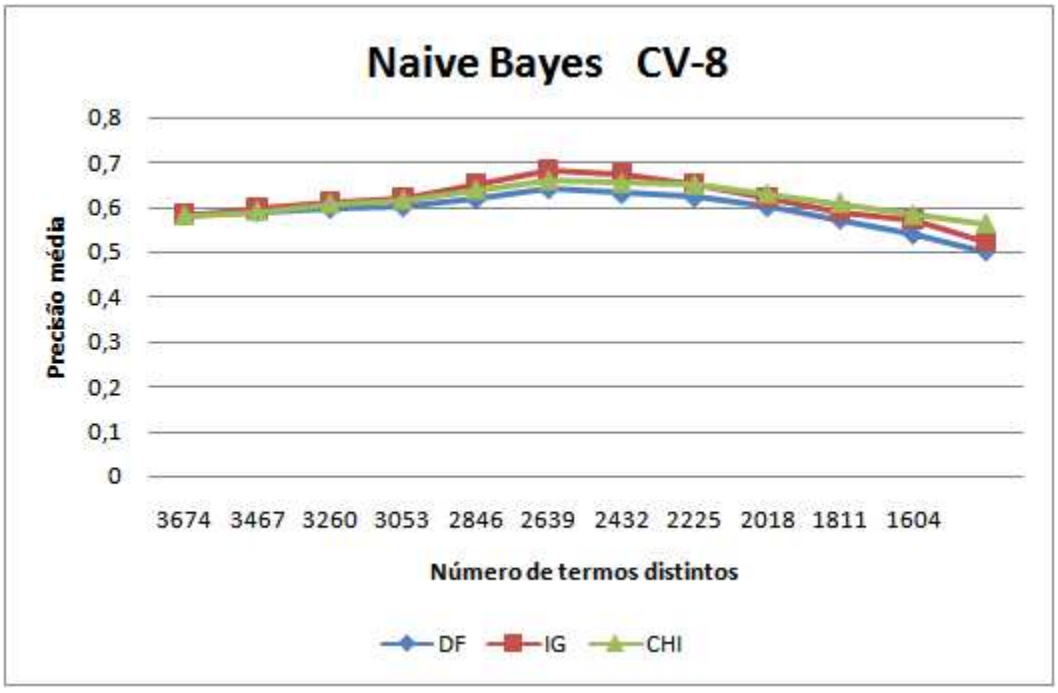

Figura 5.4 Naive Bayes e Cross Validation com 8 grupos

5.3.1 Conclusões

Como pode-se perceber, o classificador KNN obteve melhores resultados (máximo de 74% para CV-6 e 77% para CV-8) que o Naive Bayes (máximo de 67% para CV-6 e 68% para CV-8). Para o KNN, o ponto ótimo ocorreu em média com 2432 termos (maior redução e precisão) enquanto para o Naive Bayes, ocorreu com 2639 termos. Notadamente, o aumento do número de “Folds” (grupos) otimizou os classificadores (o número de documentos de treino aumentou e os de teste diminuíram) sem que ocorresse o problema de overfitting.

No geral, as técnicas de redução testadas se diferenciaram pouco entre si (evoluíram e involuíram de forma semelhante), a não ser por DF, que apresenta uma queda mais brusca depois que passa do ponto ótimo. IG se mostrou como a melhor técnica de redução, seguido por CHI e depois pelo DF. Essa baixa variação pode ser explicada pelo pequeno número de documentos e de atributos distintos. Como todos os gráficos apresentam curvas que se elevam desde do momento que o corpus não apresenta redução de características por algoritmos de aprendizagem de máquina (3674 termos restantes, depois do Stoplist), pode-se concluir que a redução de termos por algoritmos de aprendizagem de máquina aprimorou a precisão da classifição e reduziu o custo computacional.

6. Conclusão

Como foi visto neste TG, a Classificação de Texto é uma área de pesquisa que tem muita aplicabilidade nos dias de hoje, devido à explosão da Internet, que possibilitou um aumento significativo no número de documentos disponíveis no formato digital. Atualmente, os classificadores já rendem bons resultados (equiparando-se à classificação manual), mas sentem de forma significativa o efeito negativo de uma coleção de documentos com um grande número de termos (sobretudo quando muitos dos termos não são significativos).

Para assegurar e/ou melhorar a precisão e a escalabilidade de sistemas classifidores, é imprescindível que exista uma fase redução de atributos. Muitos estudos hoje recaem sobre esse problema, e muitas técnicas já foram definidas (algumas até empregam aprendizagem de máquina). Movido por esse problema da alta dimensionalidade, este TG teve por objetivo, aprofundar os estudos sobre a fase de seleção de termos, além de analisar os diversos métodos já criados.

Com o intuito de compreender detalhadamente como essa fase ocorre num sistema de classificação, e avaliar e mensurar os benefícios que ela possibilita, foi desenvolvida uma pequena aplicação de classificação de texto que implementa 3 técnicas bem conhecidas de redução (Document Frequency, Information Gain e X^2 Estatistic) baseadas em aprendizagem de máquina. Os resultados obtidos comprovam que se deve alcançar um ponto ótimo na redução (caracterizado pelas curvas no gráfico) que permita o ganho máximo de precisão para o classificador com a adoção das técnicas, e que a "natureza" do corpus e o classificador utilizado influênciam bastante "como" e "que técnicas" retornam um melhor resultado.

7. Referências

APTÉ, C., DAMERAU, F. J., AND WEISS, S. M. 1994. Automated learning of decision rules for text categorization. ACM Trans. on Inform. Syst. 12, 3, 233–251.

BOOTH, Andrew D. A. 1967. A “law” of occurrences for words of low frequency. Information and Control, [s.l.], v. 10, n.4, p.386-393.

Chen, H. 1994. Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms. University of Arizona.

DOMINGOS, P. AND PAZZANI, M. J. 1997. On the the optimality of the simple Bayesian classifier under zero-one loss. Mach. Learn. 29, 2–3, 103–130.

GILES, C. L.; HAN, H.; MANAVOGLU, E.; ZHA, H.; ZHANG, Z.; FOX, E. A. 2003. Automatic document metadata extraction using support vector machines. In JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries, pages 37–48, Washington, DC, USA, IEEE Computer Society.

GOFFMAN, W. 1966. Mathematical approach to the spread of scientifics ideas: the history of mast cell research. Nature, [s.1], v. 212, p.449-452.

GRAHAM, P. A plan for Spam. Disponível em <http://www.paulgraham.com/spam.html> Acesso em 27 dez. 2007.

GUEDES, V. L. S. 1998. Bibliometria: Uma ferramenta estatística para a gestão da informação e do conhecimento, em sistemas da informação, de comunicação e de avaliação científica e tecnológica. Posto de Serviço de Informação da Escola de Química da Universidade Federal do Rio de Janeiro. vania.doc

JOACHIMS, T. 1998. Text categorization with support vector machines: learning with many relevant features. In Machine Learning: ECML98, Tenth European Conference on Machine Learning, pages 137–142.

KOHAVI, R. 1995. "A study of cross-validation and bootstrap for accuracy estimation and model selection".
Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 1137–1143.

KOLLER, D. AND SAHAMI, M. 1997. Hierarchically classifying documents using very few words. In Proceedings of ICML-97, 14th International Conference on Machine Learning (Nashville, TN, 1997), 170–178.

KUSHMERICK, N. & THOMAS, B. 2003. Adaptive information extraction: Core technologies for information agents, Lecture Notes in Computer Science, Vol. 2586, Springer, pp. 79-103.

LEWIS, D. D. AND CATLETT, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In Proceedings of ICML-94, 11th International Conference on Machine Learning (New Brunswick, NJ, 1994), 148–156.

LEWIS, D. D. AND RINGUETTE, M. 1994. A comparison of two learning algorithms for text categorization. In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval (Las Vegas, NV, 1994), 81–93.

LUHN, H. P. 1958. The automatic creation of literature abstracts. IBM Journal of Research and Development, 2(2):159–165.

MARON, M. 1961. Automatic indexing: an experimental inquiry. J. Assoc. Comput. Mach. 8, 3, 404–417.

MCCALLUM, A. K.; NIGAM, K. 1998. A comparison of event models for naive Bayes text classification. In Proceedings of the 1st AAAI Workshopon Learning for Text Categorization, pages 41–48, Madison, US.

MCCALLUM, A. K.; ROSENFELD, R.; MITCHELL, T., NG, A. 1998. Improving text classification by shrinkage in a hierarchy of classes. Machine Learning: Proceedings of the Fifteenth International Conference, pp. 359–367.

MOSTELLER, F.; WALLACE, D. 1964. Inference and Disputed Authorship: The Federalist. Addison-Wesley, Reading, Massachusetts.

MOULINIER, I. Gailius R. Ganascia J. 1996. Text Categorization: a Symbolic Approach. Proceeding of the Fifth Annual Symposium on Document Analysis and Information Retrieval.

POLYA, G. 1968. Mathematics and Plausible Reasoning:Volume II, Princeton University Press, Princeton, New Jersey.

PORTER, M. 1980. An algorithm for suffixing stripping. Program 14(3), 130–137.

RIJSBERGEN, V. C. J. 1979. Information Retrieval, 2nd edition. Dept. of Computer Science, University of Glasgow.

ROBERTSON, S. E. AND SPARCK JONES, K. 1976. Relevance weighting of search terms. J. Amer. Soc. Inform. Sci. 27, 3, 129–146. Also reprinted in Willett [1988], pp. 143–160.

SALTON, G. AND BUCKLEY, C. 1988. Term-weighting approaches in automatic text retrieval. Inform. Process. Man. 24, 5, 513–523. Also reprinted in Sparck Jones and Willett [1997], pp. 323–328.

SEBASTIANI, F. 1999. Machine learning in automated text categorization, Tech. Rep. IEI-B4-31-1999, Consiglio Nazionale delle Ricerche, Pisa, Italy.

WANG, H. 2006. “Nearest Neighbor by Neighborhood Counting”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.28, no.6, pp. 942-953.

YANG, Y. AND PEDERSEN, J. O. 1997. A comparative study on feature selection in text categorization. In Proceedings of ICML-97, 14th International Conference on Machine Learning (Nashville, TN, 1997), 412–420.

ZIPF, G. K. 1949. Human behavior and the principle of least effort. Cambridge, Ma: Addison Wesley.