Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Uma Experiência em Mineração de Textos Utilizando *Clustering* Probabilístico *Clustering* Hierárquico

Claudia Aparecida Martins
Daniela Godoy
Maria Carolina Monard
Edson Takashi Matsubara
Analia Amandi

Nº 205

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Julho/2003

Uma Experiência em Mineração de Textos Utilizando *Clustering* Probabilístico e *Clustering* Hierárquico

Claudia Aparecida Martins[1,2]
Daniela Godoy[3]
Maria Carolina Monard[1]
Edson Takashi Matsubara[1]
Analia Amandi[3]

[1]Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
C.P. 668, 13560-970 - São Carlos, SP - Brasil
e-mail: {cam, mcmonard, edsontm}@icmc.usp.br

[2]Universidade Federal de Mato Grosso
Instituto de Ciências Exatas e da Terra
78060-900, Cuiabá, MT - Brasil

[3]Universidad Nacional de La Província de Buenos Aires
Instituto de Sistemas de Tandil
7000, Tandil, BA - Argentina
e-mail: {dgodoy, amandi}@exa.unicen.edu.ar

Resumo: Mineração de Textos é um processo de descoberta de conhecimento em dados textuais. Devido sua natureza não estruturada, o processo de Mineração de Textos é uma tarefa que exige um tratamento diferenciado do processo de Mineração de Dados em geral.

Uma abordagem simples, porém não trivial, para resolver o problema de descoberta de conhecimento em textos, consiste em analisar a freqüência das palavras em cada documento e construir tabelas no formato atributo-valor possibilitando, assim, o uso de algoritmos de Aprendizado de Máquina supervisionado e não supervisionado que trabalham com esse tipo de informação. O interessante dessa abordagem é que ela permite realizar o processo de Mineração de Textos sem a necessidade de compreender o conteúdo do texto.

O objetivo deste trabalho é descrever experimentos realizados em Mineração de Textos envolvendo duas abordagens diferentes de algoritmos de Aprendizado de Máquina não supervisionado: *clustering* probabilístico e *clustering* hierárquico. O pré-processamento dos textos utilizados nos experimentos, escritos em inglês, é realizado utilizando uma ferramenta computacional baseada na técnica de *stemming*.

Palavras Chaves: Mineração de Textos, *Clustering*, Pré-processamento.

Julho 2003

Projeto de Cooperação Brasil-Argentina, CAPES-SCyT No. 170/00.

Sumário

Resumo 2
Lista de Tabelas i

1 Introdução **1**

2 O processo de Mineração de Textos **2**
2.1 Representação de documentos 3
2.2 Redução de atributos 4

3 Pré-Processamento do Conjunto de Dados Textuais **5**

4 Experimentos Realizados **7**
4.1 Descrição do Experimento 1: *Clustering* Probabilístico 7
4.2 Resultados do Experimento 1: *Clustering* Probabilístico 8
4.3 Descrição do Experimento 2: *Clustering* Hierárquico 14
4.4 Resultados do Experimento 2: *Clustering* Hierárquico 17

5 Conclusão **18**

Referências **18**

Lista de Tabelas

1 Representação de documentos 3
2 Número de exemplos em cada cluster 7
3 Experimento 1 com valores *default* 9
4 Experimento 1 com 11 clusters 10
5 Experimento 2 com valores $\tau = 0.7$, $\varphi = 0.1$, $\sigma = 0.3$ and $\lambda = 0.9$ . . . 17

1 Introdução

Considerando a natureza dos dados e as aplicações a que está freqüentemente relacionado, o processo de descoberta de conhecimento em dados textuais, denominado Mineração de Textos (MT), muitas vezes combina técnicas de Recuperação de Informação (RI), Aprendizado de Máquina (AM) e Processamento de Língua Natural (PLN), ao longo de suas etapas. As técnicas da área de RI são empregadas na maioria das etapas do processo. Os métodos de PLN podem ser usados para preparar os dados de forma a permitir que se obtenha uma representação mais estruturada relacionada ao conteúdo dos textos, enquanto os métodos de AM são geralmente empregados na descoberta de padrões presentes nos textos.

O processo de Mineração de Textos é semelhante ao processo de Mineração de Dados (MD). Porém, enquanto MD trabalha com dados estruturados, o processo de MT trabalha com dados não estruturados, geralmente na forma de textos ou documentos, havendo, portanto, um tratamento diferenciado em algumas etapas do processo.

Uma abordagem simples, porém bastante custosa, para transformar textos em dados estruturados, consiste na construção de uma tabela no formato atributo-valor, na qual cada texto é considerado um exemplo nessa tabela e as palavras nos textos determinam os atributos. Essa forma de representação possibilita o uso de algoritmos de AM supervisionado e não supervisionado que trabalham com esse tipo de informação. O interessante dessa abordagem é que ela permite realizar o processo de MT sem a necessidade de compreender o texto. Essa transformação dos textos em tabelas atributo-valor é apenas uma das primeiras etapas de um processo de MT e é denominada de pré-processamento.

Após o pré-processamento, a tabela atributo-valor pode ser submetida a algoritmos de AM para descobrir padrões presentes nos textos ou agrupar os mesmos em clusters, entre outros. A escolha de qual algoritmo de AM usar depende de alguns fatores, tais como se a categoria dos textos são conhecidas ou não. Caso as categorias dos textos sejam conhecidas, é possível utilizar algoritmos de AM supervisionado, caso contrário, algoritmos de AM não supervisionado, mais especificamente algoritmos de *clustering*.

Neste trabalho foram realizados alguns experimentos com um conjunto de dados textuais utilizando dois algoritmos distintos de *clustering*, um probabilístico e outro hierárquico, bem como uma ferramenta computacional que realiza o pré-processamento de textos utilizando a técnica de *stemming*.

O trabalho está organizado da seguinte forma: na Seção 2 é descrito brevemente o processo de Mineração de Textos. Na Seção 3 é mostrado como foi realizado o pré-processamento dos textos, bem como algumas características da ferramenta PreTex utilizada para realizar esse pré-processamento. Na Seção 4 são apresentados os experimentos realizados e os resultados obtidos e, por fim, na Seção 5 são apresentadas algumas conclusões.

2 O processo de Mineração de Textos

Dado um conjunto de documentos[1] $D = \{d_1, d_2, ..., d_n\}$ e um conjunto de categorias $C = \{c_1, c_2, ..., c_z\}$, a tarefa de categorização de textos consiste em determinar se o documento d_i pertence (ou não) a categoria c_j para $i = 1, 2, ..., n$ e $j = 1, 2, ..., z$. As categorias, no entanto, são apenas rótulos simbólicos e nenhum conhecimento adicional de seu significado está disponível ao processo de construção do classificador induzido por um algoritmo de AM supervisionado. A atribuição de documentos às categorias deveria ser realizada baseada na compreensão do conteúdo dos documentos e não em "metadados", como por exemplo data de publicação ou tipo do documento.

No entanto, tarefas como sumarização de textos e *clustering* (agrupamento) de documentos, também podem estar relacionados ao processo de MT dependendo do tipo da aplicação. Independente da tarefa ser de categorização, sumarização ou agrupamento de documentos, algumas etapas são essenciais no processo de MT as quais, basicamente, são quatro:

1. coleta de documentos,;
2. pré-processamento;
3. extração de conhecimento;
4. avaliação e interpretação dos resultados.

Na primeira etapa do processo, coleta de documentos, o objetivo é recuperar documentos relevantes ao domínio de aplicação do conhecimento a ser extraído. Existem várias fontes, tais como livros e documentos, que podem ser obtidos na internet. Para facilitar o acesso a esses documentos, várias ferramentas de apoio têm sido desenvolvidas utilizando técnicas de RI, AM e PLN.

Após a coleta de documentos é necessário um pré-processamento dos documentos preparando-os para serem representados em um formato adequado para serem submetidos aos algoritmos de extração automática de conhecimento. Essa etapa, denominada de pré-processamento, é responsável por obter uma estrutura, geralmente, no formato de uma tabela atributo-valor, que represente o conjunto de documentos.

Com os documentos representados em uma forma adequada é possível aplicar técnicas de extração de conhecimento, utilizando sistemas de aprendizado, com a finalidade de descobrir padrões úteis e desconhecidos presentes nos documentos. Finalmente, a etapa de avaliação é necessária para verificar se o objetivo foi alcançado ou se necessitam ser refeitas todas ou algumas das etapas do processo de MT. A etapa final geralmente é realizada com o auxílio do usuário e/ou especialista do domínio.

[1] Neste trabalho, as palavras "textos" e "documentos" serão usadas indistintamente.

2.1 Representação de documentos

Considerando que a primeira etapa do processo tenha sido cumprida, ou seja, os documentos estejam disponíveis, é necessário realizar o pré-processamento dos documentos. O procedimento geralmente adotado é a representação de cada documento d_i como um vetor de m termos que ocorrem no documento. A identificação dos termos em um documento pode se referir às palavras presentes no texto (*bag of words*), ou podem ser usadas representações mais sofisticadas como frases ou sentenças.

Resultados experimentais mostraram que representações mais sofisticadas perdem em desempenho com relação a representações usando palavras simples (Apté, Damerau, & Weiss 1994; Dumais, Platt, Heckerman, & Sahami 1998; Lewis 1992). De acordo com Lewis (1992), a razão mais provável para explicar esses resultados é que, embora os termos mais sofisticados tenham qualidade semântica superior, a qualidade estatística é inferior em relação a termos baseados em palavras simples. Porém, muitas pesquisas estão sendo realizadas nas duas áreas.

Nesta implementação, um termo pode ser representado por palavras simples (1-*gram*) ou compostas (*n-grams*) que ocorrem no documento. Cada termo, portanto, será um elemento do conjunto de atributos da tabela atributo-valor. Dessa forma, a coleção de documentos, após o pré-processamento, pode ser representada como ilustrada na Tabela 1.

Tabela 1: Representação de documentos

	t_1	t_2	...	t_m
d_1	a_{11}	a_{12}	...	a_{1m}
d_2	a_{21}	a_{22}	...	a_{2m}
...	...	...	...	...
d_n	a_{n1}	a_{n2}	...	a_{nm}

Mais especificamente, a Tabela 1 representa n documentos (exemplos) e m termos (atributos[2]), e cada documento d_i é uma tupla $d_i = (a_{i1}, a_{i2}, ...a_{im})$. O valor a_{ij} refere-se ao valor associado ao j-ésimo termo do documento i, ou seja, a_{ij} representa o valor do termo t_j no documento d_i e pode ser calculado de diversas formas. Alguns autores utilizam valores binários, neste caso, o valor 1 para a_{ij} significa presença do termo j no documento i e o valor 0 ausência do termo. Mas, geralmente, medidas estatísticas são empregadas levando em consideração a quantidade de vezes que um termo aparece no documento e a quantidade de vezes que esse termo é encontrado em todos os documentos da coleção de documentos que está sendo processada.

Por exemplo, a freqüência de um termo (tf[3]) é uma medida utilizada que considera o número de ocorrências do termo em um documento d_i. O valor dos termos pode ser calculado também levando em consideração, além da freqüência de um termo, o fator relacionado a freqüência inversa do documento (idf[4]) favorecendo termos que aparecem

[2]Neste trabalho, as palavras "termos", "atributos" e "palavras" são usadas indistintamente.
[3]Do inglês *term frequency.*
[4]Do inglês *inverse document frequency.*

em poucos documentos de uma coleção, denominada de medida $tfidf$. Ainda, a medida $tfidf$ pode ser normalizada ($tfidfn$) para igualar o tamanho dos vetores tratando todos os documentos com a mesma importância independentes do seu tamanho.

2.2 Redução de atributos

Geralmente, MT trabalha com um grande número de atributos devido a sua natureza textual. É necessário, portanto, levar em consideração alguns aspectos que são essenciais na especificação dos atributos e seus valores. Tais aspectos estão relacionados com a recuperação de atributos relevantes que possam discriminar os documentos e, ao mesmo tempo, a rejeição de atributos irrelevantes que não discriminem bem os diversos documentos.

Várias ações podem ser utilizadas a fim de reduzir a quantidade de atributos visando uma melhor representatividade e melhor desempenho do sistema. Entre outras, a transformação de palavras ao radical que a originou é um método amplamente difundido e utilizado na redução da quantidade de atributos. Essa transformação é, geralmente, realizada por meio de algoritmos de *stemming*.

Algoritmos de *stemming* são bastante utilizados no pré-processamento de um conjunto de dados textuais. Basicamente, um algoritmo de *stemming* consiste em uma normalização lingüística, na qual as formas variantes de uma palavra são reduzidas a uma forma comum — *stem*. A conseqüência da aplicação de algoritmos de *stemming* consiste na diminuição do número de atributos, visto que o algoritmo remove prefixos ou sufixos de uma palavra ou transforma um verbo para sua forma no infinitivo, reduzindo a somente uma representação as variações de palavras com o mesmo radical. Percebe-se, portanto, que um algoritmo de *stemming* é fortemente dependente da linguagem na qual os textos estão escritos.

Um dos algoritmos de *stemming* mais conhecidos é o algoritmo do Porter (1980), que remove sufixos de palavras em inglês. O algoritmo tem sido amplamente usado, referenciado e adaptado nos últimos 20 anos. Diversas implementações do algoritmo estão disponibilizadas na *Web*, entre elas a página oficial escrita e mantida pelo autor, Martin Porter, para distribuição do seu algoritmo de *stemming* (`http://www.tartarus.org/~martin/PorterStemmer`).

Também, uma outra forma de reduzir a dimensionalidade dos atributos é a aplicação da Lei de Zipf. Essa lei descreve uma maneira de descobrir palavras consideradas pouco representativas a um determinado conjunto de documentos. A Lei de Zipf, formulada por George Kingsley Zipf professor de lingüística de Harvard (1902-1950), declara que a freqüência de ocorrência de algum evento está relacionada a uma função de ordenação. Zipf mostrou que uma das características das linguagens humanas, populações das cidades e muitos outros fenômenos humanos e naturais, seguem uma distribuição similar, a qual denominou de "*Principle of Least Effort*" (Zipf 1949).

Existem diversas maneiras de enunciar a Lei de Zipf, a mais simples é procedimental: pegar todas as palavras de um documento e contar o número de vezes que cada uma

aparece. Se o histograma resultante for ordenado de forma decrescente, ou seja, a palavra que ocorre mais freqüentemente aparece primeiro, então a forma da curva é "curva de Zipf", para aquele documento. Se a curva de Zipf for plotada em uma escala logarítmica, ela aparece como uma reta com inclinação -1. A Lei de Zipf em documentos de linguagem natural pode ser aplicada não apenas a palavras mas, também, a frases e sentenças da linguagem.

Ainda, de acordo com Luhn (1958), a freqüência de ocorrência de uma palavra em um documento fornece uma medida útil de quão significativa é essa palavra. Ou seja, as freqüências das palavras podem ser usadas para extrair palavras e sentenças mais relevantes na representação dos documentos. Luhn utiliza a própria Lei de Zipf a qual afirma que o produto da freqüência do uso de palavras e a posição de ordenação é, aproximadamente, um valor constante. Entretanto, enquanto Zipf verificou sua lei utilizando jornais escritos em inglês, Luhn usou a lei como uma hipótese nula para especificar dois pontos de corte, os quais denominou de superior e inferior, para excluir palavras não relevantes.

As palavras que excedem o corte superior são as mais freqüentes e são consideradas comuns por aparecer em qualquer tipo de documento, como as preposições, conjunções e artigos. Já as palavras abaixo do corte inferior são consideradas raras e, portanto, não contribuem significativamente para discriminar o conteúdo do documento. Assim, Luhn propôs uma técnica para encontrar palavras relevantes, assumindo que as palavras mais significativas para discriminar o conteúdo do 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 (Van Rijsbergen 1979). Como a Lei de Zipf, a técnica não é restrita apenas a palavras, mas também, pode ser aplicada a *stem* ou sentenças do documento.

3 Pré-Processamento do Conjunto de Dados Textuais

Neste trabalho, o pré-processamento dos textos foi realizado utilizando a ferramenta computacional PreTex (Matsubara, Martins, & Monard 2003) desenvolvida no Labic[5]. O PreTex, implementado na linguagem Perl, é uma ferramenta que contém a implementação do algoritmo de *stemming* do Porter para a língua inglesa. Além disso, contém também a implementação do algoritmo de *stemming* para textos escritos em português e espanhol, ambos adaptados do algoritmo do Porter. O objetivo do algoritmo é, para cada palavra presente em um texto, eliminar o sufixo da palavra de acordo com algumas regras pré-definidas e um comprimento mínimo estabelecido, retornando o *stem* dessa palavra. Dentre as características da ferramenta PreTex, podem-se destacar algumas, tais como:

- extrair *stems* de palavras em português, espanhol e inglês;

[5]Laboratório de Inteligência Computacional - http://labic.icmc.usp.br

- ignorar palavras que não são significativas para os textos, como por exemplo as conjunções e os artigos;

- criar arquivos com as freqüências dos *stems* de um ou de todos os textos, separados ou juntos;

- calcular quatro diferentes tipos de medidas de atribuição de valores ao atributos: booleano, tf, $tfidf$, $tfidfn$;

- aplicar a Lei de Zipf e cortes de Luhn;

- trabalhar com palavras simples ou compostas — 1, 2 e 3-*grams*;

- gerar gráficos;

- criar a tabela atributo-valor.

Considerando que o pré-processamento de textos é uma etapa crucial e bastante custosa no processo de Mineração de Textos, a ferramenta PreTex foi criada com o objetivo de ser um recurso poderoso nesta fase de pré-processamento. Portanto, a ferramenta foi desenvolvida para que o usuário tenha liberdade em definir diversos parâmetros e, assim, analisar posteriormente as diversas formas de representação obtidas do conjunto de textos.

Nos experimentos realizados neste trabalho, os parâmetros definidos para o pré-processamento dos textos foram:

1. inglês, como linguagem na qual os textos estão escritos;

2. $tfidfn$, como medida para atribuição de valor aos atributos;

3. freqüência < 15, como limite, tal que palavras com freqüência menor do que 15 sejam eliminadas — cortes de Luhn;

4. 1-*gram*, como a quantidade de *grams*, ou seja, foram consideradas apenas palavras simples para representar os atributos.

Após o pré-processamento, realizado segundo os parâmetros especificados, percebeu-se que a quantidade de *stems* que representariam os atributos finais ainda era muito elevada. Foi feito, então, uma análise dos resultados obtidos e verificou-se que algumas palavras poderiam ser inseridas na lista de *stopwords* por serem pouco representativas para o domínio em questão. Como resultado final, cada documento ficou representado como um vetor de 432 atributos, incluindo o atributo 'classe' o qual classifica o exemplo ou documento, e o atributo 'reg' o qual identifica o exemplo.

Tabela 2: Número de exemplos em cada cluster

Music	Goat	Sheep	Biomedical	Total
61 (18,38%)	70 (21,08%)	65 (19,58%)	136 (40,96%)	332 (100%)

4 Experimentos Realizados

O conjunto de dados textuais contém 332 documentos escritos em inglês e classificados em quatro diferentes temas — music, goat, sheep e biomedical — Tabela 2.

Dessa forma, a tabela atributo-valor gerada por PreTex consiste dos 332 exemplos e 432 atributos selecionados pelos parâmetros definidos. Como o objetivo deste trabalho é a análise dos resultados obtidos por algoritmos de *clustering*, aplicados a documentos textuais, a classificação dos documentos foi omitida na execução dos algoritmos nos experimentos realizados.

Os documentos, após serem transformados em uma tabela atributo-valor pela ferramenta PreTex, foram submetidos a dois diferentes algoritmos de Aprendizado de Máquina não supervisionado, um probabilístico e outro hierárquico, para realizar o agrupamento dos documentos em clusters. A descrição dos experimentos e os resultados obtidos são apresentados a seguir.

4.1 Descrição do Experimento 1: *Clustering* Probabilístico

No Experimento 1, foi utilizada a metodologia proposta em Martins, Monard, & Halembeck (2002). A metodologia consiste, basicamente, em submeter os exemplos a um algoritmo de *clustering* para agrupar os exemplos em clusters. Com a descoberta dos clusters, cada exemplo é rotulado com o cluster ao qual pertence, aumentando o conjunto de exemplos originais com um novo atributo "classe". Assim, os novos exemplos podem ser submetidos a algoritmos de AM supervisionado para indução de regras que descrevem esses clusters.

Como uma instanciação dessa metodologia, neste trabalho foi utilizado o software AutoClass como algoritmo de *clustering* probabilístico, a ferramenta computacional InClass para instanciar o novo atributo classe em cada exemplo, e o software $\mathcal{S}$ee5 como algoritmo de AM supervisionado que induz regras, descritos brevemente a seguir.

- AutoClass (Cheeseman & Stutz 1990) é uma ferramenta de *clustering* baseada na técnica probabilística que usa o modelo de Bayes. É um software robusto, de domínio público, bastante utilizado e citado na literatura. AutoClass tem várias características, tais como: determinar o número de clusters automaticamente; usar exemplos com valores discretos e reais em um mesmo conjunto de exemplos; manipular valores ausentes/desconhecidos; utilizar o tempo de processamento linear à quantidade de exemplos, aproximadamente, como critério de parada; entre outros. Basicamente, AutoClass descreve clusters a partir de uma distribuição probabilística sobre os atributos dos exemplos, considerando que existe

independência condicional nos dados. A teoria probabilística que suporta o AutoClass encontra-se bem documentada em diversas publicações (Hanson, Stutz, & Cheeseman 1991; Cheeseman, Stutz, Self, Kelly, Taylor, & Freeman 1988).

- InClass (Martins, Monard, & Halembeck 2002) é uma ferramenta computacional, desenvolvida no Labic e implementada na linguagem Perl (Wall, Christiansen, & Schwartz 1996). A ferramenta auxilia no processo de rotulamento automático dos exemplos. Mais especificamente, o InClass rotula cada exemplo do conjunto de exemplos com o cluster ao qual pertence, encontrado pelo AutoClass.

- $\mathcal{S}$ee5 (Quinlan 1993) é um software comercial de AM supervisionado para plataforma Windows™. O $\mathcal{S}$ee5 foi projetado para trabalhar com conjuntos de exemplos relativamente grandes e, como seus precursores, manipula atributos com valores discretos ou contínuos, induzindo conceitos expressos como árvores de decisão ou conjuntos de regras não ordenadas **if-then**. Seu desempenho tem se mostrado muito bom na maioria dos casos.

Em resumo, a idéia geral da metodologia consiste em combinar técnicas de *clustering* tradicionais e aprendizado indutivo, neste caso utilizando o AutoClass e o $\mathcal{S}$ee5 respectivamente, para descobrir e interpretar conceitualmente os clusters encontrados em um conjunto de exemplos.

4.2 Resultados do Experimento 1: *Clustering* Probabilístico

Num primeiro momento foi executado o AutoClass com valores *default* para seus parâmetros, isto é, sem especificar o número de clusters a serem retornados. Nesse caso, o AutoClass busca probabilisticamente o melhor agrupamento do conjunto de exemplos, determinando automaticamente qual o número de clusters, bem como os exemplos pertencentes a cada cluster. Na Tabela 3 é mostrada a quantidade de exemplos pertencentes a cada um dos seis clusters encontrados por AutoClass, bem como a distribuição dos exemplos em cada uma das quatro classes distintas[6]: music, goat, sheep e biomedical, na qual:

- Cl representa o número do cluster;

- # ExCl representa o número de exemplos e a percentagem de exemplos em cada cluster.

Do total de 332 exemplos, apenas um dos exemplos — exemplo 117 —, pertence ao cluster com probabilidade 0.995, ou seja, menor que 1. Porém, ainda assim, é uma alta probabilidade. Em todos os outros exemplos, a probabilidade de um exemplo pertencer a um dos seis clusters é igual a 1. Pode ser observado que o cluster 0 contém 48.2% dos exemplos. No entanto, os exemplos que pertencem ao cluster 0 estão distribuídos

[6]Como mencionado, essas classes não foram submetidas a AutoClass.

Tabela 3: Experimento 1 com valores *default*

Cl	# ExCl		music		goat		sheep		biomedical	
0	160	(48.2%)	10	(16.4%)	44	(62.9%)	42	(64.6%)	64	(47.1%)
1	67	(20.2%)	0		1	(1.4%)	3	(4.7%)	63	(46.3%)
2	56	(16.9%)	51	(83.6%)	2	(2.9%)	1	(1.5%)	2	(1.5%)
3	36	(10.8%)	0		18	(25.7%)	18	(27.7%)	0	
4	10	(3.0%)	0		5	(7.1%)	1	(1.5%)	4	(2.9%)
5	3	(0.9%)	0		0		0		3	(2.2%)
	332	(100.0%)	61	(100.0%)	70	(100.0%)	65	(100.0%)	136	(100.0%)

nas quatro classes distintas. Assim, é possível concluir que os exemplos no cluster 0 não têm relação com as classes a eles atribuídas pelo especialista.

As regras obtidas pelo $\mathcal{S}$ee5, bem como a matriz de confusão utilizando todos os exemplos, são mostrados a seguir:

```
Extracted rules:

Rule 1: (cover 271)
        min <= 0
    ->  class classe_0  [0.553]

Rule 2: (cover 16)
        artist > 0
    ->  class classe_2  [0.778]

Rule 3: (cover 7)
        thread > 0
    ->  class classe_3  [0.889]

Default class: classe_2


Evaluation on training data (332 cases):

        Decision Tree           Rules
      ----------------    ----------------
      Size      Errors      No      Errors

        35   19( 5.7%)       3  123(37.0%)   <<


       (a)   (b)   (c)   (d)   (e)   (f)    <-classified as
      ----  ----  ----  ----  ----  ----
       149          11                      (a): class classe_0
        66           1                      (b): class classe_1
         3          53                      (c): class classe_2
        29                 7                (d): class classe_3
         9           1                      (e): class classe_4
         3                                  (f): class classe_5
```

Como pode ser observado, o algoritmo induziu apenas 3 regras que "explicam" os exemplos nos clusters 0, 2 e 3 encontradas por AutoClass. Porém, o erro associado ao classificador é alto (37.0%). Pode-se verificar também que a maioria dos exemplos foram classificados na classe 0 (271 exemplos). Isso pode ser justificado porque enquanto o AutoClass conseguiu agrupar bem os exemplos considerando a probabilidade dos exemplos pertencerem a um mesmo cluster, o $\mathcal{S}$ee5, que divide o espaço utilizando hiperplanos, não conseguiu um bom desempenho na separação desses clusters por meio de hiperplanos.

Após a análise dos resultados, decidiu-se realizar um novo experimento com AutoClass porém fixando o número de clusters em 11. De fato, quando foi fixado em 11 clusters, o AutoClass conseguiu agrupar os exemplos em 9 clusters. Como no experimento anterior, a probabilidade de um exemplo pertencer a um dos 9 clusters é 1, exceto o exemplo 272 que pertence ao cluster com probabilidade 0.996. O resultado da distribuição dos exemplos nos clusters é mostrado na Tabela 4.

Tabela 4: Experimento 1 com 11 clusters

Cl	# ExCl		music		goat		sheep		biomedical	
0	170	(51.2%)	10	(16.4%)	52	(74.3%)	41	(63.1%)	67	(49.2%)
1	56	(16.6%)	0		0		2	(3.1%)	54	(39.7%)
2	51	(15.42%)	47	(77.0%)	1	(1.4%)	1	(1.5%)	2	(1.5%)
3	25	(7.5%)	0		4	(5.7%)	16	(24.6%)	5	(3.7%)
4	14	(4.2%)	0		9	(12.9%)	3	(4.6%)	2	(1.5%)
5	5	(1.5%)	0		0		0		5	(3.7%)
6	4	(1.2%)	0		1	(1.4%)	2	(3.1%)	1	(0.7%)
7	4	(1.2%)	4	(6.6%)	0		0		0	
8	3	(0.9%)	0		3	(4.3%)	0		0	
	332	(100.0%)	61	(100.0%)	70	(100.0%)	65	(100.0%)	136	(100.0%)

Nesse experimento, novamente se verifica a predominância do cluster 0 contendo 51% dos exemplos, porém distribuídos nas quatro classes distintas.

As regras obtidas pelo $\mathcal{S}$ee5, bem como a matriz de confusão utilizando todos os exemplos, para o experimento com 11 clusters, são mostrados a seguir:

```
Extracted rules:

Rule 1: (cover 162)
        environ <= 0
        print <= 0
        east <= 0
        shepherd <= 0
        thread <= 0
        set <= 0
        lactat <= 0
        vallei <= 0
        tutori <= 0
        newsgroup <= 0
        stori <= 0
        blood <= 0
        person <= 0

Rule 2: (cover 159)
        environ <= 0
        director <= 0
        injuri <= 0.33
        east <= 0
        shepherd <= 0
        thread <= 0
        set <= 0
        lactat <= 0
        vallei <= 0
        tutori <= 0
        newsgroup <= 0
        stori <= 0
        blood <= 0
```

```
        problem <= 0
        profession <= 0
        centr <= 0.24
        increas <= 0
        electron <= 0.17
        american <= 0.15
        kid <= 0.29
        min <= 0
    ->  class classe_0  [0.933]

Rule 3: (cover 156)
        environ <= 0
        print <= 0
            east <= 0
        shepherd <= 0
        thread <= 0
        set <= 0
        vallei <= 0
        tutori <= 0
        newsgroup <= 0
        stori <= 0
        blood <= 0
        person <= 0
        centr <= 0.24
        increas <= 0
        electron <= 0.17
        american <= 0.15
        kid <= 0.29
            min <= 0
        comput <= 0
    ->  class classe_0  [0.918]

Rule 7: (cover 2)
         stori > 0
         min <= 0
         plai > 0
    ->  class classe_0  [0.750]

Rule 9: (cover 12)
        multimedia <= 0
        electron > 0.17
        kid <= 0.29
        min <= 0
    ->  class classe_1  [0.929]

Rule 11: (cover 7)
        print <= 0
        injuri <= 0.33
        east <= 0
        shepherd <= 0
        stori <= 0
        electron <= 0.17
        american > 0.15
        min <= 0
    ->  class classe_1  [0.889]
```

```
        person <= 0
        problem <= 0
        centr <= 0.24
        increas <= 0
        electron <= 0.17
        american <= 0.15
        kid <= 0.29
        min <= 0
    ->  class classe_0  [0.925]

Rule 4: (cover 10)
        special <= 0
        engin <= 0
        min > 0
        sound <= 0
        guitar <= 0
        produc <= 0
    ->  class classe_0  [0.917]

Rule 5: (cover 12)
        engin <= 0
        min > 0
        sound <= 0
        guitar <= 0
        produc <= 0
    ->  class classe_0  [0.786]

Rule 6: (cover 2)
        multimedia > 0
        person <= 0
        electron > 0.17
    ->  class classe_0  [0.750]

Rule 8: (cover 2)
        kid > 0.29
        aid > 0
    ->  class classe_0  [0.500]

Rule 10: (cover 8)
        newsgroup > 0
    ->  class classe_1  [0.900]

Rule 12: (cover 7)
        environ > 0
        photograph <= 0
        min <= 0
    ->  class classe_1  [0.889]

Rule 13: (cover 7)
        east <= 0
        set > 0
        vallei <= 0
        min <= 0
    ->  class classe_1  [0.889]
```

```
Rule 14: (cover 6)
        east <= 0
        centr > 0.24
    ->  class classe_1  [0.875]

Rule 15: (cover 6)
        injuri <= 0.33
        lactat <= 0
        blood > 0
        kid <= 0.29
        min <= 0
     -> class classe_1  [0.875]

Rule 16: (cover 5)
        shepherd <= 0
        problem > 0
        comput > 0
    ->  class classe_1  [0.857]

Rule 17: (cover 4)
        injuri <= 0.33
        tutori > 0
    ->  class classe_1  [0.833]

Rule 18: (cover 4)
        print > 0
        injuri <= 0.33
        individu <= 0
        min <= 0
    ->  class classe_1  [0.833]

Rule 19: (cover 4)
        appear <= 0
        person > 0
        servic > 0
    ->  class classe_1  [0.833]

Rule 20: (cover 3)
        director > 0
        profession > 0
    ->  class classe_1  [0.800]

Rule 21: (cover 9)
        photograph <= 0
        shepherd <= 0
        lactat <= 0
        problem > 0
        increas <= 0
        min <= 0
    ->  class classe_1  [0.727]

Rule 22: (cover 8)
        injuri <= 0.33
        stori <= 0
        person <= 0
        problem <= 0
        profession > 0
        min <= 0
    ->  class classe_1  [0.700]

Rule 23: (cover 6)
        print > 0
        injuri <= 0.33
        lactat <= 0
        min <= 0
    ->  class classe_1  [0.625]

Rule 24: (cover 8)
        appear <= 0
        thread <= 0
        person > 0
        kid <= 0.29
        min <= 0
    ->  class classe_1  [0.500]

Rule 25: (cover 27)
        engin <= 0
        min > 0
        guitar > 0
    ->  class classe_2  [0.966]

Rule 26: (cover 26)
        engin <= 0
        min > 0
        sound > 0
    ->  class classe_2  [0.964]

Rule 27: (cover 18)
        engin <= 0
        min > 0
        produc > 0
    ->  class classe_2  [0.950]

Rule 28: (cover 6)
        special > 0
        min > 0
        sound <= 0
    ->  class classe_2  [0.875]

Rule 29: (cover 49)
        recent <= 0
        min > 0
    ->  class classe_2  [0.765]

Rule 30: (cover 2)
        print > 0

Rule 31: (cover 8)
        injuri <= 0.33
        east > 0
```

```
        injuri > 0.33
    ->  class classe_2  [0.750]

Rule 32: (cover 7)
        thread > 0
    ->  class classe_3  [0.889]

Rule 34: (cover 3)
        appear > 0
        injuri <= 0.33
        east <= 0
        photograph <= 0
        lactat <= 0
        min <= 0
    ->  class classe_3  [0.800]

Rule 36: (cover 2)
        print > 0
        individu > 0
        lactat <= 0
    ->  class classe_3  [0.750]

Rule 38: (cover 6)
        lactat <= 0
        person > 0
        kid <= 0.29
        min <= 0
        servic <= 0
    ->  class classe_3  [0.625]

Rule 40: (cover 7)
        photograph <= 0
        stori > 0
        person <= 0
        min <= 0
    ->  class classe_3  [0.556]

Rule 42: (cover 4)
        thread <= 0
        vallei > 0
        min <= 0
    ->  class classe_4  [0.833]

Rule 44: (cover 3)
        photograph > 0
    ->  class classe_4  [0.600]

Rule 46: (cover 3)
        engin > 0
        recent > 0
        min > 0
    ->  class classe_7  [0.800]


Default class: classe_0
```

```
        photograph <= 0
        vallei <= 0
    ->  class classe_2  [0.600]

Rule 33: (cover 5)
        east <= 0
        shepherd > 0
    ->  class classe_3  [0.857]

Rule 35: (cover 2)
        lactat <= 0
        learn > 0
        person > 0
        kid <= 0.29
        servic <= 0
    ->  class classe_3  [0.750]

Rule 37: (cover 5)
        photograph <= 0
        stori > 0
        person <= 0
        min <= 0
        plai <= 0
   ->  class classe_3  [0.714]

Rule 39: (cover 3)
        photograph <= 0
        lactat <= 0
        increas > 0
        min <= 0
    ->  class classe_3  [0.600]

Rule 41: (cover 4)
        aid <= 0
        kid > 0.29
    ->  class classe_4  [0.833]

Rule 43: (cover 2)
        thread <= 0
        learn <= 0
        person > 0
        kid <= 0.29
        min <= 0
        servic <= 0
    ->  class classe_4  [0.750]

Rule 45: (cover 2)
        print <= 0
        injuri > 0.33
    ->  class classe_5  [0.500]

Rule 47: (cover 3)
        lactat > 0
    ->  class classe_8  [0.600]
```

```
Evaluation on training data (332 cases):

        Decision Tree           Rules
      ----------------    ----------------
      Size      Errors      No      Errors

        40   20( 6.0%)      47   19( 5.7%)   <<


       (a)   (b)   (c)   (d)   (e)   (f)   (g)   (h)   (i)  <-classified as
      ----  ----  ----  ----  ----  ----  ----  ----  ----
       170                                                   (a): class classe_0
         3    53                                             (b): class classe_1
         1          50                                       (c): class classe_2
         3                22                                 (d): class classe_3
                           1    12                       1   (e): class classe_4
         3           1                 1                    (f): class classe_5
         2           1                 1                    (g): class classe_6
                     1                              3        (h): class classe_7
                                  1                       2   (i): class classe_8
```

Percebe-se que neste experimento, o $\mathcal{S}$ee5 induziu um número alto de regras. Porém, o erro geral foi reduzido de forma bastante significativa (de 37.0% para 5.7%), melhorando também o fator de confiança das regras.

Neste novo experimento, percebe-se que vários exemplos são cobertos por mais de uma regra, isto porque o $\mathcal{S}$ee5 é um algoritmo que induz regras não ordenadas (Baranauskas & Monard 2000). Ou seja, as regras podem se sobrepor e um exemplo pode ser coberto por mais de uma regra.

Também, é possível observar que não considerando os clusters 5, 6, 7 e 8, os quais contém um total de 16 exemplos, os 316 exemplos restantes nos outros clusters conseguem ser bem "explicados" pelas regras induzidas por $\mathcal{S}$ee5. Entretanto, esses clusters contém exemplos das quatro classes music, goat, sheep e biomedical. Assim, pode-se concluir que AutoClass agrupou os exemplos considerando outras características diferentes que aquelas que permitam reconhecer essas quatro classes, determinadas pelo especialista.

4.3 Descrição do Experimento 2: *Clustering* Hierárquico

O Experimento 2 utilizou um algoritmo de *clustering* hierárquico denominado WebIHCA (Web Document Incremental Hierarchical Clustering Algorithm), desenvolvido por Daniela Godoy. Nesse algoritmo, um número arbitrário de categorias, $C = \{c_1, c_2, \ldots, c_c\}$, é incrementalmente descoberto pelo algoritmo a cada novo exemplo disponível. Para cada uma dessas categorias existe um conjunto descritivo de palavras chaves, $K = \{k_1, k_2, \ldots, k_k\}$, com um peso associado de acordo com sua importância

para a descrição da categoria. Essas categorias são organizadas em uma hierarquia de especificidade crescente que poderia ser vista como uma árvore. Cada nó interno na árvore corresponde a uma categoria, c_i, a qual sumariza palavras chaves que são compartilhadas pelas categorias filhas, clusters e exemplos pertencentes ao nó. Assim, a raiz corresponde a categoria mais geral, a qual sumariza todos os exemplos apresentados ao algoritmo e as categorias mais internas se tornam mais específicas à medida que se desce na hierarquia. Finalmente, nos nós terminais ou folhas, clusters de exemplos são encontrados, os quais herdam todas as características das categorias superiores.

O algoritmo básico WebIHCA pode ser descrito como mostrado no Algorithm 1.

Algorithm 1 WebIHCA Algorithm

1: **WebIHCA Algorithm**($Instance$, $Tree$)
{where $Instance$ is a new instance to add to the current hierarchy $Tree$}
2: **if** $Tree$ is empty **then**
3: Initialize $Tree$ with a single root category c_{root}
4: **end if**
5: $c_{parent} \leftarrow c_{root}$
6: **while** $(\exists c : F_c(Instance) \geq \tau)$ **do**
7: c_{parent} =**BestCategory**(c_{parent})
8: **end while**
9: $s_{i,parent}$ =**BestCluster**(c_{parent})
10: Add $Instance$ to $s_{i,parent}$ =
11: **if Evaluation**($s_{i,parent}$)$\geq \varphi$ **then**
12: K =**FeatureSelection**($s_{i,parent}$)
13: Create a new category c_{new} with the set of features K
14: Create a subtree $Tree_{new}$ with c_{new} as root
15: Add $Tree_{new}$ to $Tree$ as child of c_{parent}
16: **for all** i such that c_{parent} is the parent of i **do**
17: WebIHCA(i,$Tree_{new}$)
18: **end for**
19: **end if**

Portanto, o objetivo do algoritmo WebIHCA é agrupar hierarquicamente e incrementalmente exemplos não rotulados representados como vetores de atributos. A cada novo exemplo, o algoritmo busca recursivamente a melhor categoria em cada nível hierárquico da árvore. O processo termina quando encontra uma categoria terminal (com apenas clusters como filhos) ou quando o exemplo não é suficientemente similar a qualquer nó filho.

Observando os atributos que são comuns aos exemplos em um cluster, novas categorias são extraídas para facilitar uma identificação mais rápida dos novos membros dos clusters. Assim, categorias são basicamente classificadores gerados automaticamente por um processo indutivo a partir de cada cluster de exemplos e, de acordo com o conjunto de atributos, um novo documento pode ser classificado.

Mais formalmente, um classificador para uma categoria é uma função $F_i : i_j \rightarrow [0, 1]$ tal que, dado um exemplo i_j, retorna um número entre 0 e 1 que representa a evidência

para o fato que i_j deveria ser classificado com c_i. Esta função também tem um *threshold* τ tal que $F_i(i_j) \geq \tau$ é interpretado como uma decisão em classificar i_j com c_i, enquanto $F_i(i_j) < \tau$ é interpretado como uma decisão de não classificar i_j com c_i. Inicialmente, são considerados classificadores lineares que representam uma categoria por um vetor $c_i = \langle (k_1, w_1), ..., (k_r, w_r) \rangle$ no qual w_j é o peso associado com o termo k_j na categoria c_i. Assim, essa abordagem de *clustering* consiste na construção de um conjunto hierárquico de classificadores, cada um baseado em seu próprio conjunto de atributos relevantes.

Uma vez que o exemplo foi classificado em uma dada categoria na hierarquia (nó terminal ou não), o exemplo é colocado no cluster mais similar de exemplos abaixo dessa categoria de acordo com a abordagem *k-nearest neighbor* (k-NN). Entretanto, se o exemplo não é suficientemente similar a qualquer um dos clusters existentes na categoria, um novo cluster é criado com apenas um exemplo.

A medida de distância usada pelo algoritmo é a similaridade do coseno que avalia o coseno do ângulo formado por dois vetores no espaço (Salton & McGill 1983). Essa medida é calculada utilizando a Equação 1.

$$sim_{coseno}(i_i, i_j) = \frac{i_i \bullet i_j}{\|i_i\| \, \|i_j\|} = \frac{\sum_{k=1}^{r} w_{ik} * w_{jk}}{\sqrt{\sum_{k=1}^{r} w_{ik}^2} * \sqrt{\sum_{k=1}^{r} w_{jk}^2}} \tag{1}$$

Nessa equação i_i e i_j são os exemplos considerados, w_{ik} e w_{jk} os pesos da palavra k em cada exemplo e r o número de palavras diferentes em ambos exemplos. Um *threshold* de similaridade σ é usado para decidir quando agrupar documentos em um mesmo cluster.

A inserção de um novo exemplo em um dado cluster causa a revisão da hierarquia e, possivelmente, um refinamento ou reestruturação da mesma. Nesse processo, uma nova categoria é criada iniciando de um cluster caso seja obtido um certo valor depois da aplicação de uma função de avaliação. Um classificador é criado para essa categoria baseado em um conjunto das palavras mais representativas dos documentos no clusters. Uma função de seleção de atributos é aplicada para determinar o melhor conjunto de palavras possíveis. Essa nova categoria será uma categoria filha abaixo da categoria ao qual o cluster pertence.

É considerado a *cohesiveness* do cluster como um bom indicador de quando é possível extrair atributos comuns de documentos dentro de um cluster para adequar uma categoria. O método usado para calcular a *cohesiveness* de um cluster S_r é utilizar similaridade ponderada da similaridade interna do cluster — Equação 2.

$$I = \frac{1}{|S_r|^2} \sum_{d_i, d_j \in S_r} cos(d_i, d_j) \tag{2}$$

Caso essa similaridade ultrapasse um dado *threshold* φ, considera-se que um novo cluster pode ser criado para esse exemplo.

Uma vez que o classificador é construído representando a nova categoria, exemplos pertencentes ao cluster que originaram a categoria são colocados na categoria e são

novamente organizados de acordo com a abordagem k-NN. Entretanto, nesse caso, termos extraídos para representar a categoria não são levados em consideração na comparação de exemplos, os quais devem resultar em uma partição diferente de clusters. E, desses clusters, novos classificadores serão obtidos e adicionados como nós filhos ao primeiro classificador, conseqüentemente, definindo uma hierarquia de classificadores.

4.4 Resultados do Experimento 2: *Clustering* Hierárquico

O algoritmo WebIHCA foi executado com os seguintes valores para *threshold* $\tau = 0.7$, $\varphi = 0.1$, $\sigma = 0.3$ e $\lambda = 0.9$. Nesse caso o algoritmo gerou a seguinte árvore:

```
|-- category 0 [excerpt 1, min 1, mono 1, mpeg 1,music 1, review 1, song 1, stereo 1]
    -- 36 clusters
|-- category 1 [sheep,1] -- 32 clusters
|-- category 2 [medic,1] -- 24 clusters
|-- 23 clusters
```

na qual são apresentadas três categorias, representadas por 36, 32 e 24 clusters respectivamente e, também, 23 clusters ligados diretamente à raiz da árvore. Cada uma dessas categorias é definida por um conjunto de palavras com sua correspondente importância para a categoria. Por exemplo, a categoria 1 é identificada com a palavra "sheep" que aparece em todos os documentos (probabilidade igual a 1) dentro da categoria.

Nos 198 clusters descobertos existe uma entropia total de 0.012 e pureza de 0.885. Considerando os 23 clusters que permanecem na raiz da árvore como um cluster, a distribuição de exemplos em cada uma das classes, music, goat, sheep e biomedical, é apresentada na Tabela 5.

Tabela 5: Experimento 2 com valores $\tau = 0.7$, $\varphi = 0.1$, $\sigma = 0.3$ and $\lambda = 0.9$

Cl	# ExCl		Entropia	Pureza	music		goat		sheep		biomedical	
0	37	(11.2%)	0.00	1.00	37	(60.6%)	0		0		0	
1	44	(13.2%)	0.39	0.86	0		6	(8.5%)	38	(58.5%)	0	
2	40	(12.0%)	0.23	0.95	0		1	(1.5%)	0		38	(28.0%)
3	211	(63.6%)			24	(39.4%)	63	(90.0%)	27	(41.5%)	98	(72.0%)
	332	(100.0%)			61	(100.0%)	70	(100.0%)	65	(100.0%)	136	(100.0%)

Como pode ser observado, o algoritmo encontrou quatro clusters, três desses clusters contém exemplos de três das quatro categorias. O cluster 0 contém somente exemplos da categoria music (60% deles). O cluster 1 contém 58% de exemplos da categoria sheep e 8.5% da categoria goat. O cluster 2 contém somente um exemplo da categoria goat e os restantes (28%) são da categoria biomedical. Entretanto, pode ser observado que no cluster 3 ficam exemplos de todas as categorias, sendo a majoritária, da categoria goat.

5 Conclusão

Neste trabalho foram realizados alguns experimentos com *clustering* probabilístico e hierárquico utilizando um conjunto de dados textuais. Para o pré-processamento do conjunto de dados textuais foi utilizada uma ferramenta computacional denominada PreTex, implementada no Labic. O pré-processamento de documentos é uma tarefa bastante custosa, porém, essencial a todo o processo de Mineração de Textos. Na realidade, um dos fatores que influencia a qualidade do processo de aprendizado está relacionado a quão refinada e representativa a tabela atributo-valor descreve os textos/exemplos.

É possível desenvolver ferramentas que trabalham com modelos complexos para a descoberta de conhecimento em textos. Porém, resultados experimentais mostram que ferramentas que utilizam modelos mais complexos perdem em desempenho quando comparadas com ferramentas que utilizam modelos mais simples.

Deve ser observado que algoritmos de aprendizado são fortemente dependentes do *bias* por eles utilizados. Esse aspecto é mostrado nos resultados obtidos neste trabalho utilizando dois algoritmos de aprendizado não supervisionado, um deles, AutoClass, baseado na técnica probabilística, o outro, WebIHCA, baseado na técnica hierárquica.

No caso de AutoClass, os clusters encontrados não apresentam relação direta com as categorias atribuídas pelo especialista aos documentos utilizados na experiência, enquanto que alguns dos clusters encontrados por WebIHCA estão diretamente relacionados com essas categorias. Entretanto, não é possível concluir que um método é melhor que outro. Na realidade, dado um conjunto de documentos, diferentes especialistas freqüentemente atribuem categorias distintas a esses documentos devido ao foco de atenção por eles utilizado.

Assim, se esse foco de atenção estiver relacionado com uma medida de probabilidade, espera-se que AutoClass apresente um melhor desempenho. Por outro lado, se o foco de atenção do especialista estiver relacionado com a idéia de hierarquias, e essa hierarquia não estiver relacionada com uma medida de probabilidade, espera-se que WebIHCA apresente um melhor desempenho.

Referências

Apté, C., F. Damerau, & S. M. Weiss (1994). Automated learning of decision rules for text categorization. *Information Systems 12*(3), 233–251. http://citeseer.nj.nec.com/apte94automated.html.

Baranauskas, J. A. & M. C. Monard (2000). Reviewing some machine learning concepts and methodos. Relatório Técnico 102, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_102.ps.zip.

Cheeseman, P., J. Stutz, M. Self, J. Kelly, W. Taylor, & D. Freeman (1988). Bayesian classification. In *Proceedings of the Seventh National Conference of Artificial*

Intelligence (AAAI-88). St. Paul, MN.

Cheeseman, P. & J. Stutz (1990). Bayesian classification (Autoclass): Theory and results advances in knowledge discovery and data mining. http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass-c-program.html.

Dumais, S., J. Platt, D. Heckerman, & M. Sahami (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th International Conference on Information and Knowledge Management (ACM CIKM 98)*, pp. 148–155. http://research.microsoft.com/~sdumais/cikm98.pdf.

Hanson, R., J. Stutz, & P. Cheeseman (1991, May). Bayesian classification theory. Technical report, NASA Ames Research Center. http://ic.arc.nasa.gov/ic/projects/bayes-group/images/tr-fia-90-12-7-01.ps.

Lewis, D. D. (1992, June). An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 37–50.

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

Martins, C. A., M. C. Monard, & G. C. Halembeck (2002). A computational framework for interpreting clusters through inductive learning. Technical Report 173, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_173.zip.

Matsubara, E. T., C. A. Martins, & M. C. Monard (2003). Descrição da ferramenta PreTex – uma abordagem ao pré-processamento de dados textuais. Relatório Técnico, ICMC-USP (in press).

Porter, M. (1980). An algorithm for suffix stripping. *Program 14*(3), 130–137.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Los Altos, California, USA: Morgan Kaufmann.

Salton, G. & M. McGill (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Van Rijsbergen, C. J. (1979). *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow. http://citeseer.nj.nec.com/vanrijsbergen79information.html.

Wall, L., T. Christiansen, & R. L. Schwartz (1996). *Programming in PERL*. O'Reilly,Inc.

Zipf, G. (1949). *Human Behaviour and the Principle of Least Effort*. Addison-Wesley.