Uma abordagem para pré-processamento de
dados textuais em algoritmos de aprendizado

Claudia Aparecida Martins

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 29/08/2003

Assinatura:_____

Uma abordagem para pré-processamento de dados textuais em algoritmos de aprendizado

Claudia Aparecida Martins

Orientadora: *Prof^a Dr^a Maria Carolina Monard*

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências de Computação e Matemática Computacional.

USP - São Carlos
agosto/2003

Resumo

A representação atributo-valor de documentos usada no processo de mineração de textos é uma estrutura adequada à maioria das tarefas de classificação e agrupamento de documentos. No contexto de algoritmos de aprendizado de máquina, a representação atributo-valor de documentos freqüentemente utiliza a abordagem *bag-of-words*. Essa abordagem é caracterizada pela alta dimensionalidade na representação dos dados, pois toda palavra presente no documento pode ser um possível atributo. Deve ser considerado, portanto, que uma boa representação de documentos tem uma influência fundamental no desempenho dos algoritmos de aprendizado (supervisionado ou não supervisionado).

Como uma das principais contribuições deste trabalho, é apresentada uma ferramenta para pré-processamento que eficientemente decompõe textos em palavras usando a abordagem *bag-of-words*, bem como o uso de métodos para reduzir a dimensionalidade da representação gerada. Essa ferramenta transforma os documentos em um formato acessível à maioria dos algoritmos de aprendizado, nos quais os dados são descritos como um vetor de dimensão fixa.

A ferramenta computacional implementada, entre as diversas funcionalidades, reduz a dimensionalidade da representação de documentos com o objetivo de obter um melhor desempenho dos algoritmos de aprendizado de máquina utilizados. A escolha do algoritmo de aprendizado a ser utilizado, supervisionado e não supervisionado, é dependente do problema em questão. Algoritmos de aprendizado supervisionado podem ser aplicados a documentos rotulados, enquanto algoritmos de aprendizado não supervisionado são freqüentemente aplicados a dados não rotulados.

No caso do aprendizado não supervisionado, para avaliar se um dado cluster corresponde a um certo conceito, neste trabalho é utilizada uma abordagem usando algoritmos de aprendizado indutivo para auxiliar na interpretação dos clusters. Nesta abordagem o interesse consiste em compreender como o sistema representa e raciocina sobre o conhecimento adquirido. Essa compreensão é necessária tanto para o usuário aceitar a solução gerada pelo sistema quanto para analisar o raciocínio utilizado.

Abstract

The attribute-value representation of documents used in text mining provides a natural framework for classifying or clustering documents based on their contents. In the context of machine learning algorithms, the attribute-value representation of documents is often accomplished by using the bag-of-words approach. Since every word in the document could be treated as an attribute, such an approach is characterized by yielding very high dimensional data. In fact, the representation of documents has a crucial influence on how well some learning algorithms (either supervised or unsupervised) can perform.

As one of the main contributions of this work, a framework to efficiently decomposing text into words by using the bag-of-words approach, as well as reducing the dimensionality of the representation generated, is presented. This framework makes text accessible to most machine learning algorithms for it only requires data to be described as a vector of fixed dimensionality.

In order to illustrate the framework proposed, a computational tool was implemented. This tool by, among other things, reducing the dimensionality of the text representation can lead to an improvement in the performance of the machine learning techniques being applied. The types of machine learning algorithms employed were either supervised or unsupervised, depending on the problem in hand. For instance, supervised learning algorithms can be applied whenever the documents have labels preassigned, whereas unsupervised learning algorithms are often applied to unlabelled data.

For the case of unsupervised learning, in order to evaluate if a given cluster corresponds to a certain concept, this work takes another approach by using inductive learning techniques. For example, with this approach one could understand how the system represents and reasons about the acquired knowledge. Explanation facilities are required both for user acceptance of the solution generated by the system, and for the purpose of understanding whether the reasoning is sound.

Dedicatória

Dedico esse trabalho em memória do meu pai José Pacífico Sobrinho, que sempre foi um exemplo para mim de garra, dedicação e amor. A você, por tudo que me ensinou na vida. Saudades.

Agradecimentos

A professora Carolina que é muito mais do que uma orientadora, pois orienta com sabedoria, questionamentos, ensinamentos e, principalmente, amizade. Reconheço e admiro sua atitude em ajudar seus “desorientandos”, mesmo que isso muitas vezes significa abrir mão de alguma coisa. Obrigada pela sua amizade, por aceitar ser minha orientadora, por me orientar, por me incentivar, enfim, por me ensinar. Aprendi muito com você! E a cada dia, percebo o quanto ainda tenho a aprender. A você Carolina, o meu muito obrigada por tudo!

Agradeço a todas as pessoas que me ajudaram na realização deste trabalho. Um agradecimento especial ao Edson (Takashi) por seu excelente trabalho, pela disposição em me ajudar várias vezes e, acima de tudo, por saber que sempre poderia contar com você. Ao Ronaldo que é uma pessoa bastante acessível e, por saber disso, não hesitava em importuná-lo diante de qualquer dúvida. Ao Gustavo que, entre outros, disponibilizou as suas implementações e fez os ajustes necessários para que eu pudesse utilizá-los. Ao Augusto que foi o primeiro a me dar dicas de Perl, ao Walter pelas configurações de máquinas que precisei, ao Marcelo pela ajuda com a formatação deste trabalho. Agradeço a todos vocês não só pelo trabalho, pois certamente sem a ajuda de vocês tudo seria muito mais difícil, mas também pelos bons momentos.

A convivência no Labic e do ICMC durante esses quatro anos será sem dúvida uma grande e boa recordação. Algumas pessoas foram muito mais do que apenas colegas de trabalho. Dentre essas pessoas, algumas convivi mais de perto, Patrícia, Jaque, Claudia, Katti, Huei, Edson (Melanda) e Fernanda, Lorena, Cristiane, Adriano, Gedson, Daniel, Valmir, Marquinho, Marcos, Chandler, Humberto, Sadao. Citar nomes é complicado e com certeza posso ter esquecido alguém, que me desculpe. Também, vou sempre guardar as boas recordações dos momentos divertidos e de amizade com Marcílio e Thaís, principalmente, nos dias de maior *stress* na finalização deste trabalho. Ao Marcílio agradeço também pela ajuda final com o inglês. Com muito carinho, obrigada a todos vocês!

Ao pessoal do Nilc por disponibilizar o conjunto de dados textuais e, principalmente, a Juliana Greggi que esclareceu diversas dúvidas e se mostrou disposta a me ajudar.

Aos professores Solange, André e Alneu que tornam esse ambiente de trabalho mais agradável, pela simpatia, dinamismo e disponibilização de material. Principalmente, Solange que sempre foi mais do que uma simples professora e coordenadora de pós.

Ao pessoal da pós-graduação do ICMC, Beth, Laura, Ana Paula, Marília pela eficiência, simpatia e por todos os esclarecimentos. E também ao pessoal da biblioteca.

Durante esse trabalho pude contar com a amizade e o carinho de Cleide, Rosana, Marcelinho, Alexandre, Regina, Edna, Cidinho, que mesmo longe, sempre tão perto.

Agradeço a minha família que é o meu porto seguro. A minha mãe Nilta que é força, paciência e amor. Aos meus irmãos José Antônio, Eleida e ‘Catita’, que sempre me apoiaram em qualquer coisa que tenha decidido fazer. Ao amor, algumas vezes inexplicável, que sempre recebi de meus sobrinhos: Filipe, Fabrício e Maria Beatriz. Aos meus cunhados Vanilda e Carlinhos. Todos vocês são especiais e representam muito para mim.

Agradeço o apoio financeiro da CAPES e da Universidade Federal de Mato de Grosso (UFMT). A todo o pessoal do Departamento de Ciência da Computação da UFMT.

Agradeço a Deus por estar sempre presente.

Sumário

Sumário	xv
Lista de Figuras	xix
Lista de Tabelas	xxiv
1 Introdução	1
1.1 Descoberta de Conhecimento	1
1.2 O Projeto DISCOVER	4
1.3 Motivação	6
1.4 Objetivos	8
1.5 Organização da Tese	10
2 Aprendizado de Máquina	13
2.1 Aprendizado de Máquina Indutivo	13
2.2 Aprendizado de Máquina Supervisionado	16
2.2.1 <i>Overfitting</i> e <i>Underfitting</i>	17
2.2.2 Erro e Precisão	17
2.2.3 Completude e Consistência	19
2.2.4 Algoritmos de Classificação	19
2.3 Aprendizado de Máquina não Supervisionado	21
2.3.1 Técnicas de Agrupamento	22
2.3.1.1 Técnicas de Otimização	22
2.3.1.2 Técnicas Probabilísticas	23

2.3.1.3	Técnicas Hierárquicas	24
2.3.1.4	Técnicas de Grupo (<i>clumping</i>)	25
2.3.2	Medidas de Similaridade	25
2.3.3	Softwares de Agrupamento	28
2.4	Considerações Finais	28
3	Mineração de Dados Textuais	31
3.1	Etapas do Processo de Mineração de Textos	31
3.2	O Pré-processamento de Textos	35
3.2.1	Representação de Documentos	36
3.2.2	Redução da Dimensionalidade dos Atributos	39
3.2.2.1	Eliminação de <i>Stopwords</i> e de Termos Pouco Frequentes	40
3.2.2.2	Algoritmos de <i>Stemming</i>	40
3.2.2.3	A Lei de Zipf e os Cortes de Luhn	42
3.2.2.4	Outras Técnicas de Seleção de Atributos	43
3.3	Medidas de Avaliação	44
3.4	Softwares de Mineração de Textos	49
3.5	Considerações Finais	50
4	Interpretação de Clusters Usando Aprendizado Indutivo	53
4.1	Metodologia de Interpretação de Clusters	53
4.1.1	A Metodologia Proposta	54
4.1.2	Implementação da Metodologia Proposta	56
4.1.2.1	AutoClass	58
4.1.2.2	See5	60
4.2	Ferramentas de Apoio Desenvolvidas	61
4.2.1	INCLASS	61
4.2.2	DISCOVER × AutoClass × DISCOVER — DAD	62
4.2.3	AutoClass × Prolog	63

4.2.4	Cluster $\times$ Classe	65
4.2.5	Influência dos Atributos — AutoClass $\times$ See5	66
4.3	Integração ao Projeto DISCOVER	67
4.4	Considerações Finais	69
5	PRETEXT: uma Ferramenta para Pré-processamento de Dados Textuais	71
5.1	A Ferramenta PRETEXT	71
5.2	As Medidas Implementadas para Atribuição de Valores aos Atributos . . .	74
5.3	A Implementação de PRETEXT	79
5.3.1	O Arquivo de Parâmetros	80
5.3.2	O Módulo stem.pl	83
5.3.2.1	Arquivos de Entrada	84
5.3.2.2	Arquivos de Saída	85
5.3.2.3	Execução do Módulo stem.pl	86
5.3.3	O Módulo report.pl	89
5.3.3.1	Arquivos de Entrada	89
5.3.3.2	Arquivos de Saída	90
5.3.3.3	Execução do Módulo report.pl	91
5.3.4	Scripts Auxiliares	94
5.4	Considerações Finais	96
6	Experimentos Realizados Usando a Metodologia para Interpretação de Clusters	97
6.1	O Conjunto de Dados Breast-Cancer	97
6.1.1	Configuração de Execução dos Experimentos	98
6.1.2	Resultados Obtidos sem o Atributo Classe Original	99
6.1.2.1	Fixando o Número de clusters em 2	100
6.1.2.2	Fixando o Número de Clusters em 3	101
6.1.2.3	Fixando o Número de Clusters em 5	102

6.1.2.4	Fixando o Número de Clusters em 7	103
6.1.2.5	Sem Fixar o Número de Clusters	105
6.1.3	Resultados Obtidos Usando o Atributo Classe Original	107
6.2	O Conjunto de Dados EPH	112
6.2.1	Configuração de Execução dos Experimentos	113
6.2.2	Resultados Obtidos	114
6.2.2.1	Fixando o Número de Clusters em 2	114
6.2.2.2	Fixando o Número de Clusters em 3	115
6.2.2.3	Fixando o Número de Clusters em 5	117
6.2.2.4	Fixando o Número de Clusters em 10	118
6.3	Considerações Finais	123
7	Experimentos em Mineração de Textos Usando PRETEXT	125
7.1	O Conjunto de Textos NILC	125
7.1.1	O Pré-processamento	126
7.1.2	Resultados Obtidos	127
7.2	O Conjunto de Textos 7SECTORS	131
7.2.1	O Pré-processamento	132
7.2.2	Resultados Obtidos	133
7.3	O Conjunto de Textos BGMS	135
7.3.1	O Pré-processamento	136
7.3.2	Resultados Obtidos para Categorização de Textos	137
7.3.3	Resultados Obtidos para Agrupamento de Textos	139
7.4	O Conjunto de Textos LNAI	142
7.4.1	O Pré-processamento	143
7.4.2	Resultados Obtidos	143
7.5	Considerações Finais	147
8	Conclusão	149

8.1	Principais Contribuições	149
8.2	Algumas Limitações	151
8.3	Trabalhos Futuros	152
A	Algoritmos de <i>Stemming</i>	155
A.1	Regras do Algoritmo de <i>Stemming</i> para o Inglês	155
A.2	Regras do Algoritmo de <i>Stemming</i> para o Português	158
A.3	Regras do Algoritmo de <i>Stemming</i> para o Espanhol	162
B	Sintaxe de Dados do Discover	165
	Apêndice	167
	Lista de Abreviaturas	170
	Referências	184

Lista de Figuras

1.1	O processo de KDD	2
2.1	Hierarquia do aprendizado	14
2.2	Classificação da hipótese quanto à sua completude e consistência	20
2.3	Técnicas de agrupamento: (a) Otimização (b) <i>Clumping</i> (c) Probabilística (d) Hierárquica	22
3.1	Etapas do processo de mineração de textos	33
3.2	A curva de Zipf e os cortes de Luhn	43
3.3	A curva de Zipf obtida com os <i>stems</i> desta tese	51
4.1	Metodologia proposta	54
4.2	Explicação do processo de clusters	56
4.3	Processo proposto para explicação de clusters	57
4.4	Esquema para busca search e relatório report	59
4.5	Esquema de predição predict	59
4.6	Estrutura de execução interna do DAD	63
4.7	Arquivo de saída do <i>script</i> Class2Prolog	64
4.8	Arquivo de saída do <i>script</i> cluster×classe	65
4.9	Arquivo de saída do <i>script</i> AutoClass × See5 : atributos presentes nas re- gras geradas por See5 e a influência desses atributos para AutoClass na descoberta dos clusters	67
4.10	Biblioteca de conversão para algoritmos de agrupamento	68

5.1	A ferramenta PRETEXT	72
5.2	Fator de ponderação $idf \times linear$	76
5.3	Documento <code>texto5.txt</code> representado por diferentes medidas	78
5.4	Documento <code>texto6.txt</code> representado por diferentes medidas	78
5.5	Documento <code>texto7.txt</code> representado por diferentes medidas	79
5.6	Exemplo de arquivo de parâmetros	81
5.7	<i>Rank</i> de <i>stems</i> construído a partir da frequência	83
5.8	Exemplo de arquivo <code>oneGram.all</code>	86
5.9	Exemplo de arquivo <code>oneGram.txt</code>	87
5.10	Exemplo de arquivo <code>stemWdTF.all</code>	87
5.11	Exemplo de execução do módulo <code>stem.pl</code>	88
5.12	Exemplo dos arquivos <code>.names</code> (a) e <code>.data</code> (b) na sintaxe padrão do DISCOVER	91
5.13	Exemplo de gráfico gerado utilizando o <i>rank</i> de <i>stems</i>	91
5.14	Exemplo de execução do módulo <code>report.pl</code>	93
5.15	Exemplo de execução do <i>script</i> <code>checkWord.pl</code>	94
5.16	Exemplo de execução do <i>script</i> <code>predict.pl</code>	95
6.1	Hierarquia de clusters	104
6.2	Breast-cancer: clusters descobertos por AutoClass - 10 clusters	106
6.3	Breast-cancer: clusters sobrepostos usando o atributo classe - 10 clusters	110
7.1	NILC: frequência dos <i>stems</i>	127
7.2	Erro dos classificadores	130
7.3	7SECTORS: frequência dos <i>stems</i>	132
7.4	7SECTORS: erros obtidos em relação ao número de <i>stems</i> e a % dados esparsos	134
7.5	BGMS: frequência dos <i>stems</i>	136
7.6	BGMS: erro dos classificadores	138
7.7	LNAI: frequência dos <i>stems</i>	143

A.1	Medidas de comprimento mínimo da palavra	159
B.1	Gramática do arquivo de nomes	166
B.2	O arquivo <code>voyage.names</code>	167
B.3	O arquivo <code>voyage.data</code>	168

Lista de Tabelas

2.1	Conjunto de dados no formato atributo-valor	15
2.2	Medidas de distância	27
2.3	Alguns algoritmos de agrupamento	28
3.1	Representação de documentos	37
3.2	Matriz de confusão de um classificador binário	44
3.3	Matriz de confusão para as categorias $C_{\clubsuit}, C_{\heartsuit}, C_{\spadesuit}, C_{\diamondsuit}$	46
3.4	Matriz de confusão para a categoria $C_{\clubsuit}$	46
3.5	Matriz de confusão para a categoria $C_{\heartsuit}$	46
3.6	Matriz de confusão para a categoria $C_{\spadesuit}$	47
3.7	Matriz de confusão para a categoria $C_{\diamondsuit}$	47
3.8	Medidas de desempenho para cada categoria: $C_{\clubsuit}, C_{\heartsuit}, C_{\spadesuit}, C_{\diamondsuit}$	47
3.9	Medidas de desempenho média-macro do classificador	47
3.10	Matriz de confusão geral	48
3.11	Medidas de desempenho média-micro do classificador	48
3.12	Alguns sistemas de mineração de textos	49
4.1	Relatório <code>.case-text-n</code> gerado por AutoClass	61
4.2	Arquivo final gerado pelo INCLASS	62
4.3	Relatório <code>.class-text-n</code> gerado por AutoClass	64
5.1	Representação de documentos utilizando <i>bag-of-words</i>	77
5.2	Exemplo de tabela gerada com o uso de taxonomias	90

6.1	Breast-cancer: descrição do conjunto de dados	98
6.2	Breast-cancer: descrição dos atributos	98
6.3	Breast-cancer: resultados - 2 clusters	100
6.4	Breast-cancer: regras induzidas - 2 clusters	101
6.5	Breast-cancer: resultados - 3 clusters	101
6.6	Breast-cancer: regras induzidas - 3 clusters	102
6.7	Breast-cancer: resultados - 5 clusters	102
6.8	Breast-cancer: regras induzidas - 5 clusters	103
6.9	Breast-cancer: resultados - 7 clusters	103
6.10	Breast-cancer: regras induzidas - 7 clusters	104
6.11	Breast-cancer: resultados - 10 clusters	105
6.12	Breast-cancer: clusters sobrepostos - 10 clusters	106
6.13	Breast-cancer: atributos no conjunto de regras - 10 clusters	107
6.14	Breast-cancer: regras induzidas ($P = 0$) - 10 clusters	108
6.15	Breast-cancer: regras induzidas ($P = 1$) - 10 clusters	109
6.16	Breast-cancer: resultados usando o atributo classe - 10 clusters	109
6.17	Breast-cancer: clusters sobrepostos usando o atributo classe - 10 clusters .	110
6.18	Breast-cancer: atributos no conjunto de regras usando o atributo classe - 10 clusters	110
6.19	Breast-cancer: regras induzidas usando o atributo classe ($P = 0$) - 10 clusters	111
6.20	Breast-cancer: regras induzidas usando o atributo classe ($P = 1$) - 10 clusters	112
6.21	EPH: descrição do conjunto de dados	113
6.22	EPH: resultados - 2 clusters	114
6.23	EPH: regras induzidas usando o atributo classe ($P = 0$) - 2 clusters	115
6.24	EPH: regras induzidas usando o atributo classe ($P = 1$) - 2 clusters	115
6.25	EPH: resultados - 3 clusters	116
6.26	EPH: clusters sobrepostos - 3 clusters	116
6.27	EPH: atributos no conjunto de regras - 3 clusters	116

6.28	EPH: resultados - 5 clusters	117
6.29	EPH: clusters sobrepostos - 5 clusters	117
6.30	EPH: atributos no conjunto de regras - 5 clusters	118
6.31	EPH: resultados para $C_5(2-3)$ - 5 clusters	118
6.32	EPH: atributos no conjunto de regras para $C_5(2)$ e $C_5(3)$ agrupados - 5 clusters	119
6.33	EPH: resultados - 10 clusters	119
6.34	EPH: Clusters sobrepostos - 10 clusters	119
6.35	EPH: atributos no conjunto de regras - 10 Clusters	120
6.36	EPH: resultado do agrupando em 4 conjuntos de clusters - 10 clusters . . .	121
6.37	EPH: atributos com $C_{10}(0)$ e $C_{10}(2)$, $C_{10}(3)$ e $C_{10}(4)$, $C_{10}(5)$ e $C_{10}(7)$ agrupados - 10 clusters	121
7.1	NILC: distribuição de documentos por classe	126
7.2	NILC: resultados com <i>See5</i> e CN2	129
7.3	NILC: resultados com SVMTorch II	131
7.4	7SECTORS: distribuição de documentos por classe	132
7.5	7SECTORS: resultados com C4.5rules	133
7.6	7SECTORS: resultados com Rainbow	135
7.7	BGMS: distribuição de documentos por classe	136
7.8	BGMS: resultados com C4.5rules e CN2	137
7.9	BGMS: resultados com SVMTorch II	138
7.10	BGMS: resultados com AutoClass	139
7.11	BGMS: resultados com 11 clusters	141
7.12	LNAI: resultados para 1-gram	144
7.13	LNAI: resultados para 1-gram com SVMTorch II	145
7.14	LNAI: resultados para 2-gram sem conhecimento do especialista	146
7.15	Resumo dos conjuntos de dados textuais	147
A.1	Regras de eliminação de sufixos para o inglês	158

A.2	Regras de eliminação de sufixos para o português	160
A.3	Terminações verbais do português	161
A.4	Regras de eliminação de sufixos para o espanhol	163
A.5	Terminações verbais do espanhol	164

Introdução

Uma das metas de Inteligência Artificial está relacionada com a descoberta de conhecimento, a qual constitui um dos principais desafios da área de Aprendizado de Máquina e a motivação para o desenvolvimento deste trabalho. A descoberta de conhecimento em base de dados, conhecida também como processo de *Knowledge Discovery in Databases* (KDD), é uma área interdisciplinar a qual envolve áreas como Inteligência Artificial, Estatística, Banco de Dados, entre outras. Neste capítulo é apresentada uma descrição geral do processo de descoberta de conhecimento em bases de dados. É apresentado também o projeto DISCOVER, um ambiente para descoberta de conhecimento que está em desenvolvimento no Laboratório de Inteligência Computacional (LABIC¹), e no qual serão incorporadas as implementações realizadas neste trabalho, bem como os aspectos que motivaram o desenvolvimento desta tese, os objetivos e a organização do trabalho.

1.1 Descoberta de Conhecimento

O grande volume de dados e informações disponível em diversos meios e para diferentes domínios, tem desafiado a habilidade dos seres humanos em interpretar e compreender esses dados e informações. Assim, uma nova geração de técnicas e ferramentas computacionais que, automaticamente e inteligentemente, processam e analisam os dados estão sendo desenvolvidas para atender esse desafio. Essas técnicas, derivadas de diversas áreas, são partes de um processo denominado de *Knowledge Discovery in Databases*

¹<http://labic.icmc.usp.br>.

(KDD) definido por [Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy \(1996\)](#) como:

O processo não trivial de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis presentes nos dados.

De acordo com essa definição, validade significa que os padrões descobertos devem ser válidos com algum grau de certeza; os padrões devem conter alguma novidade ou alguma informação não esperada; os padrões devem principalmente conduzir a alguma ação útil medida por uma função de utilidade; e, por fim, a meta de um processo *Knowledge Discovery in Databases* (KDD) é tornar os padrões compreensíveis a humanos para facilitar um melhor entendimento dos dados. KDD é um processo iterativo e interativo, envolvendo algumas etapas as quais são ilustradas na Figura 1.1 ([Rezende, Pugliesi, Melanda, & Paula, 2003](#)) e descritas, resumidamente, a seguir:

Figura 1.1: O processo de KDD

1. Conhecimento do domínio / identificação do problema: essa etapa é responsável pela compreensão do domínio de aplicação, bem como conhecimento relevante *a priori* e o objetivo do usuário final;
2. Pré-processamento: essa etapa é responsável pela seleção e limpeza dos dados. Essa etapa envolve operações relacionadas a tarefas específicas de mineração de dados, tais como tratamento de valores ausentes e detecção de ruídos. É nessa etapa também

que os dados são transformados em um formato adequado aos algoritmos de extração de padrões. Essa transformação também está relacionada tanto com a redução da dimensionalidade (atributos) dos dados quanto com a adição de novos atributos nos dados;

3. Extração de padrões: nessa etapa, central do processo [KDD](#), os dados são submetidos a um algoritmo para descobrir estruturas de interesse — padrões — nos dados. Os padrões descobertos são dependentes das tarefas de mineração como, classificação, regressão, agrupamento², entre outros.
4. Pós-processamento: essa etapa consiste na interpretação e avaliação dos padrões extraídos. Isso inclui a identificação de padrões interessantes e métodos de visualização dos padrões para auxiliar o usuário final na compreensão dos padrões. Critérios de avaliação incluem a validade, utilidade, novidade, compreensibilidade e interessabilidade desses padrões;
5. Utilização do conhecimento: após a obtenção, validação e compreensão dos padrões extraídos, o conhecimento pode ser utilizado pelo usuário ou integrado ao processo que o gerou.

O desempenho de todo o processo de [KDD](#) depende de quão bem cada etapa foi realizada. No entanto, a ênfase dos diversos trabalhos em [KDD](#) desenvolvidos, muitas vezes, está na etapa de extração de padrões. Essa etapa, mais conhecida como Mineração de Dados ([MD](#)) ou *Data Mining* (DM), muitas vezes é utilizada como sinônimo do próprio processo de [KDD](#). [Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy \(1996\)](#) referem-se ao processo [KDD](#) como todo o processo de descoberta útil em dados enquanto referem-se a [MD](#) como a aplicação de algoritmos de extração de padrões nos dados sem os passos adicionais do processo [KDD](#), tais como a incorporação de conhecimento *a priori* e interpretação apropriada dos resultados. Neste trabalho, os termos [KDD](#) e [MD](#) são utilizados indistintamente como sinônimos.

O processo de mineração de dados, portanto, tem como objetivo extrair e interpretar padrões a partir dos dados observados. Os padrões extraídos devem refletir algum conhecimento útil e interessante, sendo o julgamento subjetivo humano usualmente necessário. A maioria das tarefas de [MD](#) utilizam conceitos e técnicas desenvolvidas nas áreas de Inteligência Artificial, Estatística, Banco de Dados, entre outras. Apesar da interdisciplinaridade em [MD](#), a maior dificuldade encontrada na obtenção de conhecimento de alta

²*Clustering*.

qualidade está relacionada às limitações nos próprios dados. Os dados raramente são coletados com objetivo de mineração do conhecimento, mas usualmente como subproduto de outras tarefas (Fayyad, Grinstein, & Wierse, 2002). Como consequência, um conjunto de dados não será “minerado” se não existe algo a ser descoberto. Nesses dados, a análise cuidadosa entre expectativas realísticas e investigação criteriosa para exploração é ainda uma necessidade. Como citado por Fayyad (1998), ferramentas de mineração de dados não descobrem ouro quando este não existe.

Apesar desse problema estar diretamente relacionado à coleta dos dados, a etapa de pré-processamento tem como finalidade melhorar a qualidade dos dados já disponíveis e organizados na forma de um conjunto de dados (base de dados). As ações realizadas na etapa de pré-processamento de dados visam prepará-los para serem submetidos a algum algoritmo de extração de padrões. Essas ações consistem, basicamente, em identificar e tratar dados corrompidos, atributos irrelevantes e valores desconhecidos (Weiss & Indurkha, 1998; Batista, 2003).

No entanto, é comum a existência de uma próspera fonte de informações, com dados disponíveis em formato não estruturado, como textos ou documentos. Quando os dados disponíveis ao processo de mineração são textos, o processo é denominado de Mineração de Textos (MT), *Text Mining* (TM) ou *Knowledge Discovery in Texts* (KDT) (Dörre, Gerstl, & Seiffert, 1999). A etapa de pré-processamento, nesse caso, vai além das ações citadas anteriormente, pois é necessário transformar os textos em uma representação estruturada adequada para que, a partir disso, os dados possam ser submetidos ao processo como um todo. No entanto, durante a transformação dos textos em formato estruturado existe a possibilidade de que informação intrínseca ao conteúdo dos textos seja perdida. Um desafio, nesse caso, é obter uma boa representação minimizando a perda de informação. A etapa de pré-processamento em um processo de MT é, portanto, fundamental para o desempenho de todo o processo.

1.2 O Projeto Discover

A existência de ferramentas que realmente suportam todo o processo de KDD é, ainda, bastante restrita. Ferramentas comerciais, tais como Mineset™ (Silicon Graphics, 2000) e IntelligentMiner™ (IBM, 2000), geralmente tem um caráter mais exploratório e fazem uso de algoritmos e ferramentas proprietárias, o que dificulta o seu uso por pesquisadores. O uso de ferramentas de domínio público, tais como $\mathcal{MLC}++$ ³ — Machine

³<http://www.sgi.com/tech/mlc>.

Learning in C++ — (Kohavi, Sommerfield, & Dougherty, 1994, 1996), WEKA⁴ — Waikato Environment for Knowledge Analysis — (Witten & Frank, 2000) e YALE⁵ — Yet Another Learning Environment — (Fischer, Klinkenberg, Mierswa, & Ritthoff, 2002), também, não apresentam todas as características desejáveis para o uso por pesquisadores.

Apesar da biblioteca *MCC++* ter muitas facilidades, tais como interface para os principais algoritmos de aprendizado, formato padrão para os dados de entrada, obtenção de estatísticas de desempenho e visualização gráfica das estruturas simbólicas (classificadores) obtidas por alguns algoritmos, os classificadores são tratados como uma “caixa preta” não fornecendo, por exemplo, uma visão padronizada das regras induzidas por classificadores simbólicos. A incorporação de novos aplicativos à *MCC++* também não é uma tarefa trivial, já que é necessário recompilar a biblioteca a cada novo aplicativo adicionado, além da dificuldade da adaptação do novo aplicativo aos padrões da *MCC++*. Um outro problema é que, a partir de 1995, a *MCC++* passou a ser de responsabilidade da Silicon Graphics (SGI⁶) e a última versão disponibilizada data de 1997.

WEKA, ao contrário da *MCC++*, reimplementa os algoritmos de aprendizado na linguagem Java⁷. Essa abordagem padroniza as interfaces e produz código uniforme, facilitando a inclusão de novos aplicativos, mas as novas versões dos algoritmos originais podem não ser disponibilizadas no WEKA, pois exigem a sua conversão em código Java. Além disso, a recodificação de algoritmos sempre está sujeita a falhas, as quais podem causar um comportamento anômalo do novo algoritmo que não ocorria no código fonte original. O ambiente YALE também está sendo implementado em Java. Esse ambiente utiliza alguns dos algoritmos reimplementados no ambiente WEKA, mas também é capaz de executar alguns algoritmos em suas implementações originais.

Considerando essas características e aliado ao fato de que muitas pesquisas em Aprendizado de Máquina (AM), de diferentes pesquisadores, realizadas no laboratório de pesquisa LABIC têm em geral alguns componentes em comum, existindo em algumas situações duplicação de esforços no desenvolvimento de softwares, surgiu a idéia inicial do projeto DISCOVER, proposto por (Baranauskas & Batista, 2000). O projeto DISCOVER tem como objetivo fornecer um ambiente integrado para apoiar todas as etapas do processo de descoberta de conhecimento, oferecendo funcionalidades voltadas para aprendizado de máquina, mineração de dados e mineração de textos.

De uma forma geral, o projeto DISCOVER pode ser entendido como um conjunto de métodos que são aplicados sobre os dados ou sobre o conhecimento extraído a par-

⁴<http://www.cs.waikato.ac.nz/ml/weka>.

⁵<http://yale.cs.uni-dortmund.de>.

⁶<http://www.sgi.com>.

⁷<http://java.sun.com>.

tir dos dados. Dessa forma, é muito importante que o projeto DISCOVER ofereça uma base sólida para a manipulação de dados e conhecimento. Essa base é composta por sintaxes padrão para a representação de dados e conhecimento, e por bibliotecas que oferecem um conjunto de funcionalidades básicas de manipulação de arquivos nessas sintaxes. Atualmente, existem definidas sintaxes padrão para a representação de dados e para a representação de conhecimento extraído de diversos indutores simbólicos, bem como bibliotecas que oferecem funcionalidades sobre essas sintaxes padrão (Batista & Monard, 2003, Batista & Monard; Prati, Baranauskas, & Monard, 2001a,b). Novas funcionalidades estão sendo especificadas, principalmente para representação de regras de regressão (Dosualdo, 2003) e regras de associação (Melanda, 2002), bem como as funcionalidades apresentadas neste trabalho relacionadas a agrupamentos de dados. Ainda, outras implementações contemplam a combinação de múltiplos classificadores (Baranauskas & Monard, 2003; Baranauskas, 2001; Bernardini, 2002).

A vantagem do ambiente DISCOVER como ferramenta de apoio à pesquisa em KDD, em relação a outros sistemas, é a visão unificada que os formatos padrão oferecem para quem está desenvolvendo novos componentes, além de um conjunto de ferramentas de manipulação dos mesmos. Diferentemente de outros projetos, o DISCOVER tem como finalidade seu uso e extensão por pesquisadores de AM, MD e MT. O ponto crucial é que o projeto seja flexível o suficiente para permitir que novas pesquisas sejam englobadas e, simultaneamente, imponha determinados padrões de projeto e implementação que permitam a integração de todos os seus componentes, além de um conjunto de processos que suportam e criam um ambiente de execução apropriado para o sistema (Batista (2003), Capítulo 2).

Para atingir esses objetivos foi feito um estudo para fazer do processo de implementação do DISCOVER um processo de Engenharia de Software (Rozante, 2003). Em (Prati, 2003) encontra-se a proposta de um *framework* para a integração dos componentes do ambiente DISCOVER baseado em *software patterns*, no qual a composição desses componentes será realizada por meio de uma interface gráfica proposta em (Geromini, 2002). O trabalho desenvolvido nesta tese será integrado ao ambiente DISCOVER e, portanto, obedece os padrões de desenvolvimento estabelecidos nesses trabalhos.

1.3 Motivação

A enorme disponibilidade de informações existente faz com que a cada dia novas técnicas e ferramentas computacionais sejam desenvolvidas para auxiliar a extração automática de conhecimento nos dados. Se for considerado que a grande maioria dessas

informações está armazenada em formato textual não estruturado, as pesquisas em mineração de textos podem ser consideradas prósperas, úteis e aplicáveis a uma ampla variedade de tarefas.

Apesar da riqueza de conhecimento que, possivelmente, possa ser encontrada em dados textuais⁸, a maioria dos algoritmos de AM utilizados no processo de mineração, foi desenvolvido para trabalhar com dados estruturados. O desenvolvimento de algoritmos de aprendizado para manipulação direta de textos ainda é pouco desenvolvido. Visto que os algoritmos de AM para dados estruturados estão bem consolidados, uma possível solução em mineração de textos é a transformação dos textos em um formato adequado à maioria dos sistemas de AM.

Uma abordagem simples, mas que tem obtido um bom desempenho para representar textos, é a abordagem *bag-of-words*. Nessa abordagem, os textos são transformados em um formato estruturado, representados como um vetor das palavras que ocorrem no texto. Dessa forma, após serem transformados nesse formato estruturado, os textos podem ser submetidos a maioria dos algoritmos de AM existentes. Diversos estudos foram realizados com diferentes abordagens para representação de textos. Porém, os resultados mostram que a abordagem *bag-of-words* é uma boa escolha para a representação de textos. No entanto, o vetor de palavras que representam os textos na abordagem *bag-of-words* é caracterizado pela alta dimensionalidade, sendo que muitas dessas palavras não são representativas para discriminar o conteúdo dos textos. Assim, é importante o desenvolvimento e uso de técnicas para reduzir a dimensão do espaço de representação dos textos.

Um outro aspecto, relacionado a dados/textos no processo de descoberta de conhecimento, é se os dados estão associados a um conceito, em geral denominado de *classe*. Esse aspecto faz diferença quanto ao provável tipo de tarefa a ser realizada. Para dados rotulados, ou seja, associados a um conceito, é possível utilizar algoritmos de aprendizado na tentativa de generalização do conceito a partir dos dados. No caso dos dados não estarem associados a um conceito, é possível tentar agrupar dados similares de acordo com algum critério (medida) de similaridade. Nesse último caso, os algoritmos de aprendizado não conseguem explicar, necessariamente, o conceito embutido nesses dados agrupados.

Considerando essas particularidades relacionadas ao tratamento de dados em um processo de descoberta de conhecimento, a principal motivação deste trabalho está relacionada com o pré-processamento de dados não estruturados (textos) transformando-os em dados estruturados e na redução da dimensionalidade desses dados para serem submetidos a algoritmos de aprendizado. Uma outra motivação está relacionada com a explicação conceitual dos agrupamentos encontrados por algoritmos de aprendizado, caso não existir

⁸As palavras ‘textos’, ‘documentos’ e ‘dados textuais’ são usadas indistintamente nesse trabalho.

um conceito associado aos dados.

1.4 Objetivos

A pesquisa realizada neste trabalho tem dois grandes propósitos, os quais estão relacionados com a descoberta de conhecimento em dados não rotulados e/ou dados não estruturados. O primeiro consiste na investigação de uma metodologia para ser aplicada a dados não rotulados, com o objetivo de encontrar padrões que possam representar um conceito associado aos dados agrupados. O segundo consiste na investigação de métodos de pré-processamento de dados textuais não estruturados no processo de MT. Nesse caso, o objetivo consiste em obter uma boa representação dos dados textuais em uma forma estruturada para serem submetidos a algoritmos de aprendizado.

Extraír padrões de dados não rotulados consiste, basicamente, em construir um modelo descritivo dos dados⁹. Nesse caso, um modelo descritivo pode ser o agrupamento dos dados de acordo com alguma similaridade. Uma das principais contribuições deste trabalho, relacionado a dados não rotulados, consiste não somente em descobrir agrupamentos (clusters) de dados mas, também, auxiliar a interpretação dos clusters usando algoritmos de aprendizado. Para atingir esse objetivo é proposta uma metodologia para descobrir e interpretar padrões especiais de um conjunto de dados, auxiliando o usuário/especialista a organizar os dados em grupos semanticamente interpretáveis. A metodologia proposta de interpretação de clusters é uma combinação de técnicas de agrupamento e aprendizado indutivo que está estruturada em várias etapas, sendo um processo iterativo e iterativo na extração de conhecimento útil em dados não rotulados. Essa metodologia é avaliada experimentalmente usando conjuntos de dados distintos, a qual mostrou que é viável e útil sua utilização no processo de interpretação dos clusters.

Algumas perguntas podem ser respondidas como resultado da pesquisa realizada sobre dados não rotulados:

1. *Como encontrar uma descrição semântica para os dados agrupados?*

Geralmente, os algoritmos utilizados para agrupamento de dados utilizam algum critério (medida) para encontrar similaridades entre os dados. No entanto, dificilmente características conceituais são encontradas usando somente esse critério.

Assim, usando uma combinação de algoritmos de aprendizado supervisionado e não

⁹A distinção principal entre um modelo descritivo e preditivo é que a predição tem um atributo único que é central ao processo de aprendizado — a classe —, enquanto que em problemas descritivos não existe esse atributo central ao problema.

supervisionado, juntamente com o auxílio do especialista, é possível encontrar clusters que representam semanticamente os exemplos agrupados.

2. *Quais são os algoritmos de aprendizado que devem ser utilizados na metodologia proposta de interpretação de clusters?*

Qualquer algoritmo pode ser utilizado. Porém, cada algoritmo tem um *bias*, ou seja, tem uma certa “preferência” por uma solução em relação a outra. Nos experimentos realizados foi verificado que dependendo do conjunto de dados nem sempre o uso de algoritmos com *bias* distintos obtém um bom resultado.

3. *Como os clusters podem ser avaliados?*

Após os clusters serem encontrados, os dados podem ser rotulados com os clusters aos quais pertencem e regras simbólicas relacionadas aos clusters podem ser induzidas. Ainda, várias outras informações relacionadas aos principais atributos utilizados, bem como a sobreposição de clusters, podem ser obtidas. As regras e as informações relacionadas aos clusters são analisadas e avaliadas pelo especialista do domínio.

O segundo objetivo deste trabalho, relacionado a dados não estruturados, está direcionado à etapa de pré-processamento de textos usando a abordagem *bag-of-words*. Para atingir esse objetivo foi implementada uma ferramenta computacional denominada PRETEXT para realizar o pré-processamento de dados não estruturados. PRETEXT é uma ferramenta de fácil interação e possui diversas funcionalidades, entre elas, transformação dos textos em uma representação estruturada, redução da dimensionalidade dos dados e indução construtiva¹⁰. Assim, usando PRETEXT em dados não estruturados é possível analisar como a representação dos textos usando a abordagem *bag-of-words* influencia o desempenho dos algoritmos de aprendizado utilizados.

Algumas perguntas podem ser respondidas como resultado da pesquisa realizada sobre dados não estruturados:

1. *Por que PRETEXT foi implementado?*

Como a maioria dos dados disponíveis estão em formato textual, é necessário realizar o pré-processamento automático desses textos para serem submetidos a algoritmos de aprendizado. Assim, apesar de existirem outras ferramentas disponíveis que realizam o pré-processamento de textos, PRETEXT tem diversas facilidades e fornece uma gama muito grande de informações relacionadas aos textos, características nem

¹⁰A indução construtiva pode ser caracterizada pela construção de um novo atributo que é uma generalização de dois ou mais atributos.

sempre presentes nas outras ferramentas e que consideramos necessárias. Além disso, a ferramenta pode ser facilmente integrada ao projeto DISCOVER.

2. *Quais facilidades e informações usando PRETEXT ajudam no processo de mineração?*

Além de transformar os textos escritos em três línguas diferentes para um formato estruturado, adequado à maioria dos algoritmos de aprendizado, estão implementadas algumas técnicas para reduzir a dimensão dos dados, uso de indução construtiva, gráficos, informações relacionadas à frequência dos atributos para cada texto, bem como para todo o conjunto, entre outros.

3. *Como avaliar o desempenho do processo de descoberta de conhecimento em textos?*

No PRETEXT é fácil obter diversas representações com tamanhos (dimensão) distintos para um mesmo conjunto de textos. É possível, então, analisar e avaliar o desempenho dos algoritmos de aprendizado para cada representação definida pelo usuário e gerada por PRETEXT.

4. *O tamanho dos textos tem influência no resultado esperado?*

Sim, os textos de tamanho muito pequeno quando convertidos a um formato estruturado podem não ter uma boa representação usando a abordagem *bag-of-words* influenciando, portanto, no desempenho do algoritmo de aprendizado. Nos experimentos realizados foi possível observar que para textos com tamanho muito pequeno e assunto muito específico o aprendizado se torna uma tarefa árdua necessitando, talvez, de uma outra abordagem de representação, o que não ocorre com textos de tamanhos maiores.

1.5 Organização da Tese

Esta tese está organizada da seguinte forma:

Capítulo 2: Aprendizado de Máquina Nesse capítulo são introduzidos os conceitos relacionados ao aprendizado de máquina em Inteligência Artificial, com ênfase no aprendizado indutivo por exemplos, mais especificamente, o aprendizado indutivo supervisionado e não supervisionado, que são os dois tipos de aprendizado utilizados neste trabalho.

Capítulo 3: Mineração de Dados Textuais Nesse capítulo é realizado um estudo sobre o processo de mineração em dados não estruturados — textos. São apresen-

tadas diversas características relacionadas a esse processo, principalmente na etapa de pré-processamento dos textos, cujo desafio consiste em obter uma boa representação dos textos em uma forma estruturada.

Capítulo 4: Interpretação de Clusters Usando Aprendizado Indutivo Nesse capítulo é proposta uma metodologia para auxiliar na descoberta e composição de clusters encontrados em conjuntos de dados que não possuem um rótulo associado, com o objetivo de encontrar alguma interpretação semântica nesses agrupamentos. A metodologia é implementada usando tanto algoritmos de aprendizado não supervisionado, para encontrar os clusters, quanto algoritmos simbólicos de aprendizado supervisionado para auxiliar o especialista na interpretação conceitual desses clusters.

Capítulo 5: PRETEXT: uma Ferramenta para Pré-processamento de Dados Textuais Nesse capítulo é descrita a ferramenta computacional implementada para realizar pré-processamento de textos utilizando a abordagem *bag-of-words*. A ferramenta possui diversos recursos que permitem ao usuário/especialista interagir facilmente com a ferramenta, analisar a representação estruturada obtida a partir dos textos e repetir o processo, caso necessário.

Capítulo 6: Experimentos Realizados Usando a Metodologia para Interpretação de Clusters Nesse capítulo, a metodologia proposta para auxiliar na interpretação de clusters é avaliada experimentalmente usando dois conjuntos de dados. São discutidos os resultados obtidos e as diversas informações que podem ser adquiridas para auxiliar o especialista na interpretação dos dados agrupados em cada cluster.

Capítulo 7: Experimentos em Mineração de Textos Usando PRETEXT Nesse capítulo são analisados diversos experimentos realizados com vários conjuntos de dados textuais utilizando PRETEXT. É mostrado como as facilidades da ferramenta PRETEXT podem ser utilizadas conjuntamente com a avaliação dos resultados obtidos usando diferentes algoritmos de aprendizado supervisionado. A metodologia de interpretação de clusters usando PRETEXT em conjuntos de dados textuais é também ilustrada.

Capítulo 8: Conclusão Nesse capítulo são apresentadas as conclusões deste trabalho e propostas para trabalhos futuros.

Apêndice A: Algoritmos de *Stemming* Nesse apêndice são mostradas as regras dos algoritmos de *stemming* para a língua inglesa, portuguesa e espanhola, utilizadas

neste trabalho. Essas regras foram adaptadas e implementadas na ferramenta PRETEXT.

Apêndice B: Formato Padrão de Dados do Discover Nesse apêndice é apresentada a sintaxe padrão de dados definida no projeto DISCOVER e adotada na implementação da ferramenta PRETEXT.

Aprendizado de Máquina

A capacidade de aprender é essencial para um comportamento inteligente. De fato, a habilidade dos seres humanos de fazer generalizações a partir de alguns fatos, ou descobrir padrões em coleções de observações, é realizada por meio do aprendizado. Aprendizado, geralmente, é uma forma de inferência lógica que permite que conclusões sejam obtidas a partir de premissas ou exemplos particulares. Aprendizado de máquina é uma área de pesquisa em Inteligência Artificial cujo objetivo é o desenvolvimento de métodos computacionais que possam adquirir novos conhecimentos, novas habilidades e novos meios de organizar o conhecimento já existente (Mitchell, 1997). Várias abordagens de aprendizado podem ser utilizadas para que um sistema computacional possa aprender e melhorar seu desempenho por meio de observações. Entre elas, o aprendizado por indução. Neste capítulo são apresentados alguns conceitos básicos relacionados ao aprendizado de máquina indutivo, os quais são utilizados neste trabalho.

2.1 Aprendizado de Máquina Indutivo

Uma das estratégias mais utilizadas para aprender novos conceitos é o aprendizado por indução. Indução permite que conclusões genéricas sejam obtidas a partir de conjuntos de fatos ou observações (exemplos) particulares. Ela é caracterizada como o raciocínio que parte do específico para o geral, do particular para o universal, da parte para o todo. O conhecimento generalizado extrapola aquele contido nos fatos e ele pode ou não preservar a verdade. Na realidade, a conclusão obtida pode exceder a informação contida nas premissas, sendo a inferência indutiva um dos principais meios de criar novos conhecimentos e prever eventos futuros.

O aprendizado indutivo por exemplos, um tipo especial de aprendizado indutivo, consiste em induzir descrições gerais de conceitos utilizando exemplos específicos desses conceitos (Michalski, Carbonell, & Mitchell, 1983). Diversos algoritmos de AM utilizam o aprendizado indutivo para realizar generalizações e especializações na indução de uma hipótese, ou conhecimento, a partir de exemplos. Nesse caso, o algoritmo de aprendizado é um sistema computacional que toma decisões baseados em experiências acumuladas contidas em casos resolvidos com sucesso (Weiss & Kulikowski, 1991).

Basicamente, o aprendizado de máquina indutivo por exemplos pode ser dividido em aprendizado *supervisionado* e aprendizado *não supervisionado*, como ilustrado na Figura 2.1. A escolha de qual tipo de aprendizado utilizar, supervisionado ou não supervisionado, depende da tarefa e dos exemplos estarem ou não rotulados com o atributo classe. O escopo deste trabalho abrange tanto tarefas de aprendizado supervisionado quanto não supervisionado, mais especificamente, os nós na figura que estão em destaque com uma linha tracejada.

Figura 2.1: Hierarquia do aprendizado

No aprendizado supervisionado, o objetivo é induzir conceitos a partir de exemplos que estão pré-classificados, ou seja, os exemplos estão rotulados com uma classe conhecida, ou conceito. Se as classes possuem valores discretos o problema é conhecido como *classificação*. Caso as classes possuam valores contínuos o problema é conhecido como *regressão*. Já no aprendizado não supervisionado os exemplos não possuem uma classe correspondente. Nesse caso, as tarefas podem estar relacionadas com o *agrupamento* dos exemplos, com uma descrição compacta de um subconjunto de dados, denominado *sumarização*, ou com a caracterização, por meio de *regras de associação*, do quanto a presença de um conjunto de atributos implica na presença de algum outro conjunto distinto de atributos nos mesmos exemplos.

No entanto, em qualquer uma dessas tarefas de aprendizado é necessário definir um formalismo de representação, ou linguagem de descrição, a ser utilizada para descrever os exemplos, as hipóteses e o conhecimento do domínio.

Linguagem de descrição de exemplos — $\mathcal{L}_E$ — descreve um objeto (exemplo) do mundo. Na maioria dos algoritmos de AM os exemplos são descritos por um vetor de valores de atributos. Um atributo é uma característica ou uma informação que visa descrever o exemplo, o qual pode ter ou não um rótulo associado. Esse rótulo é a classe do exemplo e representa um atributo especial que descreve uma instância do fenômeno de interesse, que é o conceito que se deseja induzir em tarefas de classificação.

Assim, os exemplos utilizados em um processo de AM podem ser representados como um conjunto de dados de treinamento $E = \{E_1, E_2, \dots, E_N\}$ pertencentes a um dado domínio com uma distribuição D fixa, desconhecida e arbitrária. Cada exemplo E_i é, tipicamente, um vetor da forma $\langle a_{i1}, a_{i2}, \dots, a_{iM}, y_i \rangle$, com valores discretos e/ou contínuos. A Tabela 2.1 ilustra o formato de uma tabela *atributo-valor*, um dos formatos mais utilizados pelos algoritmos de aprendizado. Nessa tabela está representado um conjunto de dados com N exemplos e M atributos. Nesse formato, as colunas $(A_1, \dots, A_M)$ representam os diferentes atributos e as linhas $(E_1, \dots, E_N)$ os diferentes exemplos.

Tabela 2.1: Conjunto de dados no formato atributo-valor

	A_1	A_2	$\dots$	A_M	Y
E_1	a_{11}	a_{12}	$\dots$	a_{1M}	y_1
E_2	a_{21}	a_{22}	$\dots$	a_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_N	a_{N1}	a_{N2}	$\dots$	a_{NM}	y_N

A linha i na tabela refere-se ao i -ésimo exemplo e cada valor a_{ij} refere-se ao valor do j -ésimo atributo A_j do exemplo E_i . O atributo Y é o atributo que assume os valores do atributo classe de cada exemplo E_i , para o aprendizado supervisionado. Apesar dessa representação ser a mais utilizada por algoritmos de AM, outras representações mais complexas contendo relações entre atributos ou conjuntos de atributos do exemplo, podem ser utilizadas (Basili, Moschitti, & Pazienza, 2000; Montes-y-Gómez, Gelbukh, & López-López, 2002).

Linguagem de descrição de hipóteses — $\mathcal{L}_h$ — descreve a hipótese, ou classificador, induzida por um sistema de aprendizado supervisionado. A hipótese pode ser descrita utilizando, entre outras, as seguintes linguagens de descrição: lógica de pred-

icados, redes semânticas, árvores de decisão ou mesmo funções matemáticas. Por exemplo, uma árvore de decisão é uma árvore na qual cada nó não terminal representa um teste ou decisão no item considerado. Dependendo do resultado do teste é escolhida uma ramificação. Para classificar um item particular, inicia-se no nó raiz e vai percorrendo a árvore até encontrar um nó terminal. Quando o nó terminal é encontrado, uma decisão é feita. Árvores de decisão podem ser interpretadas como uma forma especial de um conjunto de regras disjuntas.

Linguagem de descrição do conhecimento do domínio — $\mathcal{L}_D$ — descreve o conhecimento do domínio, tais como, conceitos aprendidos anteriormente, restrições do domínio, hipóteses candidatas, metas a serem inferidas, métodos para avaliar as hipóteses candidatas. A linguagem $\mathcal{L}_D$ utilizada, muitas vezes, é semelhante a linguagem $\mathcal{L}_h$ para descrever hipóteses.

A escolha do tipo de descrição para cada uma das linguagens de representação é dependente de fatores relacionados aos dados de entrada (exemplos), à tarefa a ser realizada e ao objetivo desejado. Neste trabalho, é adotada a tabela atributo-valor como $\mathcal{L}_E$ e, principalmente, estruturas simbólicas como $\mathcal{L}_h$, ou seja, o interesse principal desta tese está na indução de hipóteses simbólicas — Seção 2.2.4 na página 19 —, como árvores ou regras de decisão. Como mencionado, o escopo deste trabalho abrange tanto o aprendizado supervisionado quanto o aprendizado não supervisionado¹. Cada uma dessas abordagens é apresentada em maiores detalhes nas próximas seções.

2.2 Aprendizado de Máquina Supervisionado

O processo de aprendizado supervisionado é caracterizado pela apresentação do conjunto de dados de treinamento $E = \{E_1, E_2, \dots, E_N\}$ a um algoritmo de aprendizado denominado indutor. Cada exemplo $E_i \in E$ possui uma classe y_i associada, podendo ser representado como uma tupla

$$E_i = (\vec{a}_i, y_i)$$

O vetor $\vec{a}_i$ corresponde aos valores dos atributos para o exemplo E_i e y_i é um dos possíveis valores do atributo Y associado a E_i . Para tarefas de classificação, o atributo Y assume um conjunto de valores discretos $\{C_1, C_2, \dots, C_{Ncl}\}$. O objetivo do aprendizado supervisionado é descobrir um mapeamento geral dos vetores $\vec{a}$ para valores de y . Os

¹A partir de agora, assumiremos que [AM](#) supervisionado está se referindo a tarefas de classificação e [AM](#) não supervisionado a tarefas de agrupamento.

algoritmos de aprendizado supervisionado induzem uma hipótese $\mathbf{h}$, que aproxima a verdadeira, mas desconhecida, função $y = f(\vec{a})$, ou seja, $\mathbf{h}(\vec{a}) \approx f(\vec{a})$. Em outras palavras, assumindo que os vetores $\vec{a}$ correspondem a pontos em um espaço n -dimensional $\mathbb{R}^n$, a meta do algoritmo de aprendizado é encontrar a hipótese $\mathbf{h}$ que aproxima a função $f : \mathbb{R} \rightarrow Y$ e, nesse caso, $\mathbf{h}$ é denominado *classificador* (Mitchell, 1997).

Após a indução do classificador $\mathbf{h}$, induzido a partir dos exemplos, a função do classificador é *prever* o valor correspondente y para exemplos previamente não vistos. Assim, tarefas de predição, como classificação, consistem na generalização de exemplos, que tenham uma classe conhecida, em uma linguagem capaz de reconhecer a classe de um novo exemplo ainda não rotulado. Entretanto, deve-se ter o cuidado ao induzir uma hipótese para que ela não seja excessivamente especializada aos exemplos utilizados para criá-la, um problema chamado de *overfitting*.

2.2.1 *Overfitting* e *Underfitting*

Usualmente, um conjunto de dados é dividido em dois subconjuntos disjuntos: o conjunto de treinamento usado para o aprendizado do conceito e o conjunto de teste usado para medir o grau de efetividade do conceito aprendido. Quando um classificador é induzido, é possível que ele seja muito específico para o conjunto de treinamento, apresentando um alto grau de precisão para os exemplos de treinamento e uma alta taxa de erro para o conjunto de teste. Nesse caso, pode-se dizer que o classificador “decorou” os exemplos, não conseguindo generalizar o conceito. Essa situação é conhecida por *overfitting*.

Por outro lado, também é possível que o conjunto de treinamento seja composto por poucos exemplos representativos, ou que o usuário predefina o tamanho do classificador como muito pequeno, por exemplo, usando um alto fator de poda² para uma árvore de decisão. Nesse caso, pode-se dizer que o classificador não conseguiu abstrair o conceito, apresentando baixo desempenho no conjunto de treinamento e no conjunto de teste. Essa situação é conhecida como *underfitting*.

2.2.2 Erro e Precisão

Para avaliar o classificador, os conjuntos de exemplos de treinamento e teste são normalmente disjuntos para assegurar que as medidas obtidas, utilizando o conjunto de

²Poda em árvores de decisão consiste na exclusão de algumas ramificações da árvore, tornando-a mais “simples”.

teste, sejam de um conjunto diferente do usado para realizar o aprendizado, tornando a medida estatisticamente válida. O classificador também pode ser avaliado considerando a precisão, compreensibilidade, grau de interesse, velocidade de aprendizado, requisitos de armazenamento, grau de compactação ou qualquer outra propriedade desejável que determine quão bom e apropriado ele é para a tarefa em questão (Salzberg, 1995; Dietterich, 1997; Freitas, 1999; Gomes, 2002).

Como resultado do aprendizado, a hipótese é a compreensão de um conceito pelo sistema. Tal conceito é obtido pela observação dos exemplos no conjunto de treinamento. Entretanto, a principal meta do aprendizado supervisionado é generalizar esse conceito de forma a predizê-lo em exemplos não utilizados pelo indutor, ou seja, aqueles que não fazem parte do conjunto de treinamento. Logo, um critério importante para o sucesso do aprendizado é a precisão da hipótese em exemplos não vistos, ou seja, nos exemplos do conjunto de teste. É possível calcular o erro, ou a precisão, de uma hipótese utilizando o próprio conjunto de treinamento. Tais medidas são denominadas erro, ou precisão, *aparente*, mas elas não representam uma boa estimativa do erro para novos exemplos.

Uma medida muito utilizada em problemas de classificação é a *taxa de erro* do classificador $\mathbf{h}$, denotada por $err(\mathbf{h})$. A taxa de erro é obtida utilizando a Equação 2.1, a qual compara o rótulo real de cada exemplo com o rótulo predito pelo classificador. A expressão $\| y_i \neq \mathbf{h}(\vec{a}_i) \|$ retorna 1 caso $y_i \neq \mathbf{h}(\vec{a}_i)$ e zero caso contrário; N é o número de exemplos. O complemento da taxa de erro, denominado *acurácia* do classificador, denotado por $acc(\mathbf{h})$ é dado pela Equação 2.2.

$$err(\mathbf{h}) = \frac{1}{N} \sum_{i=1}^N \| y_i \neq \mathbf{h}(\vec{a}_i) \| \quad (2.1)$$

$$acc(\mathbf{h}) = 1 - err(\mathbf{h}) \quad (2.2)$$

Freqüentemente, o cálculo do erro e da acurácia de uma hipótese é realizado utilizando técnicas de amostragem³. Técnicas de amostragem caracterizam-se por dividir o conjunto de dados em várias partições, a serem utilizadas como conjuntos de treinamento e teste. A seguir são descritas algumas dessas técnicas.

Amostragem aleatória: nessa técnica de amostragem, diferentes partições de treinamento-teste são escolhidas de forma aleatória. Para cada partição de treinamento é induzida uma hipótese que tem então seu erro calculado a partir da partição de teste

³ *Resampling*.

correspondente. O erro final é a média dos erros de todas as hipóteses induzidas.

k-fold cross-validation: nesse caso os exemplos são aleatoriamente divididos em k partições mutuamente exclusivas de tamanho aproximadamente igual a $\frac{N}{k}$ exemplos, sendo N o número total de exemplos. Utilizam-se então $(k - 1)$ partições para treinamento, e a hipótese induzida é testada na partição restante. Esse processo é repetido k vezes, cada vez considerando uma partição diferente para teste. O erro final é a média dos erros calculados com cada uma das k partições.

Leave-one-out: esta técnica é um caso especial de *k-fold cross-validation*. Nesse caso, o valor de k é igual ao número total de exemplos N . Assim, a cada iteração, um dos exemplos é utilizado para testar a hipótese induzida a partir dos $(N - 1)$ exemplos restantes. O erro final é a soma dos erros em cada teste dividido por N . A técnica *leave-one-out* é usualmente utilizada no caso do conjunto de dados ser pequeno.

2.2.3 Completude e Consistência

Uma vez induzida, uma hipótese pode também ser avaliada pela sua *completude*, se ela classifica todos os exemplos, e pela sua *consistência*, se ela classifica os exemplos corretamente. Assim, dada uma hipótese, ela poderá ser completa e consistente, incompleta e consistente, completa e inconsistente ou incompleta e inconsistente. Na Figura 2.2 são ilustrados esses quatro casos considerando dois atributos X_1 e X_2 e três classes ($\circ, +, *$). As regiões representadas por linhas sólidas representam a predição da hipótese para a classe $\circ$, regiões representadas por linhas pontilhadas para a classe $+$ e a região tracejada para a classe $*$.

Devido ao fato de quase sempre existir um número grande de hipóteses consistentes com os exemplos, todos os indutores possuem alguma forma de *bias*. O *bias* de um algoritmo é qualquer preferência do indutor sobre uma hipótese em relação a outra, que seja consistente com os exemplos. De fato, aprendizado sem *bias* é impossível (Mitchell, 1982).

2.2.4 Algoritmos de Classificação

A escolha do algoritmo de aprendizado também está relacionada ao fato da linguagem de descrição da hipótese — $\mathcal{L}_{\mathbf{h}}$ — ser de fácil compreensão ao ser humano. Assim, no aprendizado supervisionado, a hipótese $\mathbf{h}$ pode ser vista de duas formas básicas. A primeira é uma visão genérica na qual a hipótese induzida $\mathbf{h}$ representa o conhecimento

Figura 2.2: Classificação da hipótese quanto à sua completude e consistência: (a) completa e consistente, (b) incompleta e consistente, (c) completa e inconsistente, (d) incompleta e inconsistente (Monard & Baranauskas, 2003a).

adquirido visto como uma “caixa preta”. Nesse caso, a hipótese $\mathbf{h}$ pode estar representada com uma linguagem de difícil interpretação pelo ser humano. Esse é o caso das redes neurais que são construções matemáticas relativamente simples inspiradas no modelo biológico do sistema nervoso. A hipótese $\mathbf{h}$ gerada pelas redes neurais, geralmente não é compreensível pelos seres humanos, apesar de obter muitas vezes uma boa precisão. Por esse motivo, há interesse em encontrar outras formas de conhecimento de redes neurais (Milaré, 2003; Milaré, Carvalho, & Monard, 2002).

A segunda é uma visão mais apurada, na qual a “caixa preta” pode ser aberta e interpretada. Nesse caso, é levado em conta o paradigma de aprendizado e a linguagem de descrição na qual o conhecimento adquirido é expresso. Quando o conceito descrito por $\mathbf{h}$ é facilmente interpretável por seres humanos, geralmente, $\mathbf{h}$ pode ser transformado em conjuntos de regras *if-then*. Nesse caso, $\mathbf{h}$ consiste de um conjunto de regras com N_R regras, ou seja, $\mathbf{h} = \{R_1, \dots, R_{N_R}\}$, e é denominada de hipótese simbólica. Uma hipótese simbólica deve ser legível e interpretável por humanos. Regras e árvores de decisão são representações simbólicas amplamente usadas como $\mathcal{L}_h$ no aprendizado indutivo. Árvores de decisão são robustas a erros na classificação dos exemplos de treinamento e nos valores dos atributos que descrevem esses exemplos. O ponto forte da indução de regras é

sua compreensibilidade e pouco espaço de armazenamento. Entretanto, o processo para induzir regras é, em geral, mais lento que o para induzir árvore de decisão (Monard & Baranauskas, 2003b).

2.3 Aprendizado de Máquina não Supervisionado

No aprendizado não supervisionado é fornecido ao sistema de aprendizado um conjunto de dados $E = \{E_1, E_2, \dots, E_N\}$ em que cada exemplo E_i consiste somente de vetores $\vec{a}$, não incluindo a informação sobre a classe y . O objetivo desse tipo de aprendizado é construir um modelo que procura por regularidades nos exemplos, formando *agrupamentos* de exemplos que tenham características similares. Ou seja, os exemplos são *descritos* em agrupamentos, ou clusters, de acordo com alguma medida de similaridade. Diferentemente de tarefas preditivas, como classificação, tarefas de descrição consistem em identificar comportamentos intrínsecos no conjunto de dados, que não possuem uma classe especificada.

Apesar do grande avanço na área, a tarefa de agrupamento de exemplos tem ainda vários problemas em aberto. Existem algumas questões importantes a serem levadas em consideração nesse tipo de tarefa, muitas ainda sem uma resposta precisa, tais como:

- como os dados deveriam ser normalizados?
- qual medida de similaridade é apropriada para uma dada situação?
- como deveria ser utilizado o conhecimento do domínio?
- como agrupar eficientemente um grande conjunto de dados?
- como avaliar os agrupamentos encontrados pelos diversos algoritmos de aprendizado?

Dessa forma, a seleção de um algoritmo de agrupamento adequado a um problema específico, muitas vezes requer a utilização de alguns critérios. Um conjunto de critérios de admissibilidade podem ser usados para comparar algoritmos de agrupamento. Estes critérios são baseados na maneira como os clusters são formados, na estrutura dos dados e na “sensibilidade” da técnica de agrupamento para mudanças que não afetem a estrutura dos dados (Fisher & Van Ness, 1971). A seguir são apresentadas algumas técnicas de agrupamento em geral utilizadas em AM não supervisionado.

2.3.1 Técnicas de Agrupamento

Não existe nenhuma técnica de agrupamento universalmente aplicável a qualquer conjunto de dados. É necessário, portanto, avaliar as diferentes técnicas existentes para escolher a que melhor se ajusta a uma aplicação particular. As diferentes técnicas de agrupamento são classificadas de acordo com os resultados obtidos pelos algoritmos. Na Figura 2.3 é ilustrada a classificação dessas técnicas, as quais são descritas sucintamente a seguir (Witten & Frank, 2000; Chiara, 2003).

Figura 2.3: Técnicas de agrupamento: (a) Otimização (b) *Clumping* (c) Probabilística (d) Hierárquica

2.3.1.1 Técnicas de Otimização

Visam formar uma k -partição ótima sobre os exemplos de treinamento. O conjunto de dados é dividido em k clusters mutuamente exclusivos, sendo o valor de k dado pelo usuário/especialista. As técnicas de otimização são computacionalmente caras, pois fazem uma busca exaustiva pela k -partição ótima. Por isso, seu uso é restrito a pequenas quantidades de exemplos e/ou pequenos valores de k .

Um algoritmo conhecido que utiliza a técnica de otimização é o k -means (MacQueen, 1967). Esse algoritmo inicialmente escolhe k pontos aleatórios que serão os centros dos

clusters iniciais — centróides. O próximo passo é determinar, para cada exemplo do conjunto de dados, o cluster ao qual ele pertence, calculando a distância entre o exemplo e o centro de cada cluster. O exemplo irá pertencer ao cluster do qual ele estiver mais perto do centro. Após, é calculado um novo centróide para cada cluster, no qual os pontos iniciais não são mais os centros dos clusters, que eram apenas uma tentativa inicial. O centróide calculado, para cada cluster, passa a ser o novo centro. O processo se repete até que os centros dos clusters se estabilizem, ou seja, o mesmo ponto é escolhido como centro durante algumas iterações.

Um problema relacionado com essa técnica é que os centros dos clusters finais nem sempre representam uma solução ótima. Isso porque diferentes configurações de clusters podem ser obtidas a partir de diferentes escolhas para os centros dos clusters iniciais. Para encontrar os melhores agrupamentos, ou não convergir para um mínimo local, é necessário executar o algoritmo várias vezes, com diferentes centros iniciais, e escolher o melhor resultado. Há alguns estudos no intuito de se encontrar métodos para a seleção dos centros iniciais dos clusters, e não mais escolhê-los aleatoriamente ([Bradley & Fayyad, 1998](#); [Hamerly & Elkan, 2002](#)).

2.3.1.2 Técnicas Probabilísticas

As técnicas probabilísticas associam, para cada exemplo, a probabilidade dele pertencer a cada um dos clusters. A necessidade de se especificar a probabilidade dos exemplos de pertencer a um cluster ou outro advém do fato de, normalmente, não haver evidências suficientes sobre os dados de forma a se tomar uma decisão determinística sobre qual exemplo deve pertencer a qual cluster. O algoritmo *Expectation-Maximization* ([EM](#)) é um exemplo de algoritmo que utiliza a técnica probabilística.

O algoritmo [EM](#), assim como a maioria dos algoritmos que usam técnicas probabilísticas, baseia-se em um modelo estatístico chamado “misturas finitas”. Uma mistura finita é um conjunto de k distribuições de probabilidades, representando k clusters, que governam os valores dos atributos dos exemplos naquele cluster. Em outras palavras, cada distribuição determina a probabilidade de um determinado exemplo possuir um certo conjunto de valores de atributos dado que esse exemplo pertence a esse cluster. Cada cluster tem uma distribuição diferente sendo que, na realidade, cada um dos exemplos pertence a um e somente um desses clusters, mas não se sabe a qual deles. O problema é que não se sabe nem de quais clusters vieram os exemplos e nem os parâmetros do modelo de mistura. O que o algoritmo [EM](#) faz é utilizar o mesmo procedimento utilizado no algoritmo k -means: faz-se uma tentativa inicial para estimar os parâmetros do modelo,

utilizam-se esses parâmetros para calcular as probabilidades dos exemplos pertencerem a cada cluster (*expectation*), reestimam-se os parâmetros com base nas probabilidades calculadas (*maximization*) e repete-se o processo.

O algoritmo *k*-means finaliza quando os centros dos clusters se estabilizam, já o algoritmo EM tem um problema: ele converge para um valor fixo, mas nunca o atinge. Para contornar esse problema, o algoritmo calcula um valor chamado probabilidade geral. Esse valor aumenta a cada iteração e mede a qualidade dos agrupamentos. Com isso, o processo de iteração se repete até que o aumento da probabilidade geral se torne desprezível. Deve ser ressaltado que o algoritmo EM não calcula os agrupamentos ótimos, sendo que todo o processo deve ser repetido algumas vezes para se obter o melhor resultado. McLachlan & Krishnan (1997) explicam com maiores detalhes o algoritmo EM e algumas extensões propostas.

2.3.1.3 Técnicas Hierárquicas

As técnicas hierárquicas criam árvores denominadas de dendrogramas. As folhas da árvore representam exemplos individuais e os nós internos representam grupos (clusters) de exemplos. Esses métodos podem ser subdivididos em aglomerativos e divisivos. Os métodos aglomerativos constroem a árvore de baixo para cima (das folhas, ou exemplos, para a raiz), enquanto os divisivos constroem a árvore de cima para baixo (da raiz para as folhas). As técnicas hierárquicas tem um custo computacional menor do que as técnicas de otimização.

Geralmente, os algoritmos incrementais utilizam essa técnica trabalhando individualmente com cada exemplo do conjunto de dados para formar uma árvore, na qual cada folha é um exemplo e a raiz representa todo o conjunto de dados. Inicialmente, a árvore é formada apenas pela raiz. Os exemplos são adicionados, um de cada vez, formando um novo cluster ou juntando-se a um cluster já existente. O algoritmo também realiza reestruturações maiores na árvore. Para decidir onde cada exemplo será colocado, um valor denominado “utilidade da categoria” é calculado. Esse valor mede a qualidade de um agrupamento. Cada novo exemplo é processado tentando colocá-lo em cada um dos nós filhos da raiz calculando a utilidade da categoria para cada possibilidade de agrupamento. Se não for oportuno, o exemplo é colocado como um novo filho da raiz. Mas, se a utilidade da categoria for boa, então o exemplo forma um novo cluster com aquele nó filho. Caso esse nó já seja um cluster, o algoritmo é chamado recursivamente com esse cluster sendo a raiz da sub-árvore.

Um problema que ocorre com esse algoritmo é que ele cria uma árvore de hierarquia

sobrecarregada, com cada exemplo do conjunto de dados em suas folhas. Para evitar isso, um parâmetro de corte é dado para evitar o crescimento da árvore. O corte é especificado em termos da função de utilidade da categoria. Sempre que o aumento na utilidade da categoria, causado pela adição de um novo nó for muito pequeno, esse nó não é considerado.

2.3.1.4 Técnicas de Grupo (*clumping*)

As técnicas de grupo retornam clusters nos quais seus elementos (exemplos) podem se sobrepor. O problema com essas técnicas é que um mesmo exemplo pode pertencer a vários clusters. Embora nas técnicas probabilísticas as probabilidades relativas do exemplo pertencer a cada um dos clusters são dadas, cada exemplo pertence a apenas um deles. Por outro lado, a técnica probabilística pode ser vista como uma técnica de grupo, caso sejam usadas as probabilidades relativas dos exemplos, como uma forma de pertencer a vários clusters.

Deve ser observado que esta não é a única maneira de classificar métodos de agrupamento. Outras classificações podem ser encontradas em (Michalski & Stepp, 1987; Jain, Murty, & Flynn, 1999).

2.3.2 Medidas de Similaridade

Na abordagem clássica de agrupamento, os clusters são determinados exclusivamente com base em uma medida de similaridade pré-definida. Após determinar os atributos relevantes para descrever os exemplos, vetores de valores desses atributos são utilizados para descrever todo o conjunto de dados. Considerando os atributos como dimensões de um espaço multidimensional, a descrição de cada exemplo corresponde a um ponto no espaço, então, a similaridade entre os exemplos pode ser medida usando uma função que calcula a distância entre os pontos no espaço de descrição dos exemplos (Michalski & Stepp, 1987). Por exemplo, assumindo que os vetores de descrição dos exemplos E_1 e E_2 são $E_1 = \vec{a}_1 = (a_{11}, a_{12}, \dots, a_{1M})$ e $E_2 = \vec{a}_2 = (a_{21}, a_{22}, \dots, a_{2M})$ respectivamente, onde $a_{i1}, a_{i2}, \dots, a_{iM}$ são os valores dos atributos que descrevem esses exemplos, uma medida simples de distância é dada pela Equação 2.3.

$$D(\vec{a}_1, \vec{a}_2) = \sum_{j=1}^M |a_{1j} - a_{2j}| \quad (2.3)$$

Como a distância é uma função que envolve somente os atributos relativos a dois

exemplos, o processo de agrupamento baseado em similaridade pode ser feito de forma relativamente simples e sem necessitar de nenhum tipo de conhecimento sobre o assunto tratado. Essa abordagem baseada em similaridade tem produzido algoritmos de agrupamento eficientes, que têm sido úteis em várias aplicações (McCallum, Nigam, & Ungar, 2000; Jain, Murty, & Flynn, 1999; Guha, Rastogi, & Shim, 1998; Huang, 1997; Michalski & Stepp, 1987).

Porém, devido a variedade de tipos e escalas dos valores dos atributos, a medida de similaridade deve ser escolhida cuidadosamente. As características utilizadas para definir similaridade são especificadas e combinadas em uma medida a ser calculada para todos os pares de exemplos. Similaridade entre exemplos pode ser medida de diversas formas, porém, as medidas mais comuns são as medidas de distância entre dois exemplos, que satisfazem as seguintes condições:

1. $D(\vec{a}_1, \vec{a}_2) \geq 0$. A distância entre dois pontos (exemplos) não pode ser negativa;
2. $D(\vec{a}_1, \vec{a}_2) = 0$, se e somente se $\vec{a}_1 = \vec{a}_2$. A distância entre qualquer ponto e ele mesmo é zero;
3. $D(\vec{a}_1, \vec{a}_2) = D(\vec{a}_2, \vec{a}_1)$. Propriedade reflexiva;
4. $D(\vec{a}_1, \vec{a}_2) \leq D(\vec{a}_1, \vec{a}_3) + D(\vec{a}_3, \vec{a}_2)$. Desigualdade triangular.

Na Tabela 2.2 são mostradas as medidas de distância mais utilizadas. Medidas de distância são, na realidade, medidas de dissimilaridade pois, quanto maior o valor da medida de distância, menor a similaridade.

Dentre as medidas mostradas na Tabela 2.2, a medida Euclidiana é provavelmente a mais utilizada. A medida Minkowsky estabelece uma forma genérica para calcular a distância entre dois pontos de acordo com o valor de r utilizado. Quando $r = 1$ é a distância Manhattan e $r = 2$ é a distância Euclidiana. Na medida Mahalanobis V é a matriz de covariância dos atributos $A_1, A_2, \dots, A_M$ e $\det V$ é o determinante dessa matriz. A medida Chebychev é apropriada para casos nos quais dois exemplos são considerados diferentes caso possuam qualquer atributo diferente. Na medida Chi-quadrado, sum_i é a soma de todos os valores do atributo i do conjunto de dados e $size_{a_1}$ e $size_{a_2}$ é a soma de todos os valores dos atributos $\vec{a}_1$ e $\vec{a}_2$ respectivamente, baseada no teste Chi-quadrado de igualdade para dois conjuntos de frequências.

A medida de correlação é um outro tipo de medida de similaridade entre exemplos. Essa medida é o coeficiente de correlação entre um par de exemplos, o qual é medido

Tabela 2.2: Medidas de distância

Medida	Fórmula
Euclideana	$D(\vec{a}_1, \vec{a}_2) = \sqrt{\sum_{i=1}^M (a_{1i} - a_{2i})^2}$
Manhattan	$D(\vec{a}_1, \vec{a}_2) = \sum_{i=1}^M a_{1i} - a_{2i} $
Minkowsky	$D(\vec{a}_1, \vec{a}_2) = \left(\sum_{i=1}^M a_{1i} - a_{2i} ^r \right)^{\frac{1}{r}}$
Mahalanobis	$D(\vec{a}_1, \vec{a}_2) = [\det V]^{\frac{1}{M}} (\vec{a}_1 - \vec{a}_2)^T V^{-1} (\vec{a}_1 - \vec{a}_2)$
Camberra	$D(\vec{a}_1, \vec{a}_2) = \sum_{i=1}^M \frac{ a_{1i} - a_{2i} }{ a_{1i} + a_{2i} }$
Chebychev	$D(\vec{a}_1, \vec{a}_2) = \max_{i=1}^M a_{1i} - a_{2i} $
Chi-quadrado	$D(\vec{a}_1, \vec{a}_2) = \sum_{i=1}^M \frac{1}{sum_i} \left(\frac{a_{1i}}{size_{a_1}} - \frac{a_{2i}}{size_{a_2}} \right)^2$

utilizando diversos atributos. Medidas de correlação representam similaridade pela correspondência de exemplos por meio de suas características. A medida de correlação é definida pela Equação 2.4.

$$D(\vec{a}_1, \vec{a}_2) = \frac{\sum_{i=1}^M (x_{1i} - \bar{a}_{1i})(a_{2i} - \bar{a}_{2i})}{\sqrt{\sum_{i=1}^M (x_{1i} - \bar{a}_{1i})^2 \sum_{i=1}^M (a_{2i} - \bar{a}_{2i})^2}} \quad (2.4)$$

Ainda, um outro tipo de medida de similaridade é a medida de associação. Medidas de associação são utilizadas para comparar exemplos cujas características são medidas apenas para atributos com valores discretos. Por exemplo, para valores de atributos que poderiam ser “sim” ou “não” uma medida de associação poderia verificar o grau de concordância entre cada par de exemplos. A forma mais simples de medida de associação seria considerar a porcentagem de vezes que ocorre uma concordância, ambos casos respondem “sim” ou ambos respondem “não”, para valores dos atributos. Extensões desse coeficiente simples de concordância têm sido desenvolvidos para acomodar atributos nominais multicategóricos e medidas ordinais niveladas. Uma revisão de vários tipos de medidas de associação pode ser encontrada em (Everitt, 1980; Sneath & Sokal, 1973).

2.3.3 Softwares de Agrupamento

Na Tabela 2.3 são apresentadas informações de alguns softwares freqüentemente citados na literatura que realizam, entre outros, tarefas de agrupamento (Halembek, 2001). Vários desses softwares possuem ferramentas gráficas de visualização.

Tabela 2.3: Alguns algoritmos de agrupamento

Sistema	Técnica	Software Livre	Linguagem
Autoclass	probabilística http://www.ic-www.arc.nasa.gov/ic//projects/bayes-group/autoclass	sim	Lisp/C
Clustan Graphics	hierárquica http://www.clustan.com/clustangraphics_features.html	não	C
CViz	probabilística http://www.alphaworks.ibm.com/tech/cviz	sim	Java
DataMiner	otimização http://www.data-miner.com	não	Java
EMclust	hierárquica http://www.stat.washington.edu/fraley/mclust/soft.shtml	sim	Fortran
Mineset	otimização http://www.sgi.com/software/mineset	não	C++
Normix	probabilística http://www.alumnus.caltech.edu/~wolfe/normix.htm	sim	Fortran
OC	hierárquica http://www.barton.ebi.ac.uk/new/software.html	sim	C
SnoB	probabilística http://www.case.monash.edu.au/~dld/Snob.html	sim	Fortran

Os softwares listados nessa tabela não são de forma alguma exaustivos, mas eles servem para ilustrar a técnica utilizada por alguns softwares de agrupamento. Nesta tese, o software de agrupamento utilizado foi **AutoClass**, o qual é apresentado na Seção 4.1.2.1 na página 58.

2.4 Considerações Finais

No aprendizado supervisionado hipóteses são induzidas utilizando um conjunto de dados de treinamento rotulados com o atributo classe. Uma vez induzida, uma hipótese pode ter sua precisão avaliada utilizando-se os exemplos do conjunto de teste. A hipótese ou classificador é uma generalização dos exemplos de treinamento e é utilizada para prever a classe de um exemplo não rotulado desconhecido.

No aprendizado não supervisionado, também conhecido como aprendizado por observação e descoberta, a tarefa do algoritmo é agrupar exemplos similares que não estão

rotulados com o atributo classe. Nesse caso, é possível utilizar algoritmos de aprendizado para descobrir padrões nos dados a partir de alguma caracterização de regularidade, sendo esses padrões denominados clusters. Exemplos contidos em um mesmo cluster são mais similares, segundo alguma medida de similaridade, do que aqueles contidos em clusters diferentes.

Os algoritmos de aprendizado de máquina, supervisionado e não supervisionado, são freqüentemente usados no processo de mineração de dados. Os dados, no entanto, podem estar ou não em um formato estruturado adequado a esses algoritmos. Assim, antes dos algoritmos de aprendizado serem utilizados por um processo de mineração de dados, os dados necessitam ser pré-processados para serem submetidos aos algoritmos de aprendizado. No próximo capítulo, é apresentado o processo de mineração de dados textuais, que utiliza algoritmos de aprendizado, cujos dados de entrada a serem ‘minerados’ estão em formato textual não estruturado.

Mineração de Dados Textuais

Mineração de dados textuais, ou simplesmente mineração de textos, é o processo utilizado para descobrir padrões interessantes e úteis em um conjunto de dados textuais. Apesar de similar ao processo de mineração de dados, que trabalha com dados estruturados, o processo de mineração de textos difere, principalmente, por trabalhar com dados não estruturados em formato textual. Assim, para que esses dados textuais possam ser submetidos a algoritmos de aprendizado de máquina empregados em tarefas de mineração dos dados, é necessário um tratamento diferenciado na etapa de pré-processamento de dados textuais. Neste capítulo é apresentada uma visão geral do processo de mineração de textos bem como as principais tarefas e etapas relacionadas a esse processo. É dada ênfase especial à etapa de pré-processamento de textos, por ser objeto de estudo desta tese.

3.1 Etapas do Processo de Mineração de Textos

Pode-se considerar que o formato textual é a forma mais natural de armazenar informações. De fato, um estudo indica que 80% da informação das empresas estão contidas em documentos textuais (Tan, 1999). Apesar disso, e da grande quantidade de informação na internet estar disponível em conjuntos de dados não estruturados, as pesquisas relacionadas à área de mineração de textos são relativamente poucas se comparadas à área de mineração de dados. Embora, MT tenha um potencial comercial maior do que o próprio processo de MD, é um processo muito mais complexo que MD pois trabalha com dados que são textos, ou documentos, que são inerentemente não estruturados e que, muitas vezes, possuem ambigüidade. A complexidade das tarefas de MT estão relacionadas a

vários fatores, tais como:

- a língua na qual o documento está escrito. Os algoritmos que manipulam textos são, geralmente, dependentes da língua;
- o estilo no qual o documento está escrito. Por exemplo, documentos mais formais são mais fáceis de serem processados do que documentos informais;
- a natureza do conteúdo do documento. Frequentemente, documentos grandes que contêm bastante informação irrelevante são difíceis de serem processados, assim como documentos que contêm informação não textual, como figuras;
- a especificação da tarefa a ser realizada, a qual depende do nível de detalhe e natureza da informação buscada, bem como a adequação das estruturas internas escolhidas para representar a informação.

Portanto, a escolha do tipo de tarefa a ser realizada no processo de [MT](#) depende desses fatores. Dentre as principais tarefas relacionadas ao processo de [MT](#), destacam-se:

Categorização: semelhante à tarefa de classificação em mineração de dados. A categorização consiste em induzir um classificador que possa determinar se o documento pertence ou não a uma categoria pré-definida. A categorização de documentos é uma tarefa bastante utilizada no caso de existir um conjunto de documentos previamente classificados. Muitas das técnicas de extração de padrões utilizadas em categorização de documentos são semelhantes às utilizadas em [MD](#) ([Sebastiani, 2002](#); [Joachims, 2002](#)).

Agrupamento: ou *clustering* de documentos, consiste em agrupar os documentos em um conjunto finito de clusters. Técnicas de agrupamento de textos levam em conta as palavras que aparecem nos documentos para definir a função de similaridade a fim de determinar o agrupamento final ([Hatzivassiloglou, Gravano, & Maganti, 2000](#); [Yang & Wang, 2003](#)).

Sumarização: a sumarização consiste em reduzir o tamanho de um documento preservando o seu significado. O sumário de um documento é derivado de um texto fonte condensado pela seleção e/ou generalização em palavras ou frases chaves presentes em textos o qual mantém o conteúdo informativo do documento original ([Mani, 2001](#); [Pardo, Rino, & Nunes, 2003](#)).

Como se pode perceber, nas tarefas de categorização e agrupamento de textos, é comum a utilização de algoritmos de aprendizado supervisionado e não supervisionado, respectivamente. No entanto, na maioria dos algoritmos de aprendizado, os dados de entrada necessitam estar em um formato estruturado. Os documentos, portanto, devem ser transformados no formato adequado para serem submetidos a esses algoritmos. Uma boa representação dos textos nessa etapa é fundamental para o bom desempenho desses algoritmos.

Assim, independentemente da tarefa estar relacionada à categorização, sumarização ou agrupamento de textos, algumas etapas são necessárias e comuns à maioria das tarefas de MT. Essas etapas do processo de MT, ilustradas na Figura 3.1, são descritas resumidamente a seguir.

Figura 3.1: Etapas do processo de mineração de textos

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

A primeira etapa do processo, coleta de documentos, consiste em recuperar documentos relevantes ao domínio de aplicação do conhecimento a ser extraído. Diversas fontes estão disponíveis para coletar os documentos, tais como livros e, principalmente, documentos provenientes da internet. Para facilitar o acesso a esses documentos na internet, muitas ferramentas de apoio têm sido construídas usando as seguintes abordagens: Motores de Busca Baseados em Robô (*Robotic Internet Search Engines*), Mega-Indexadores (*Mega-Indexes*), Mega-Indexadores Simultâneos (*Simultaneous Mega-Indexes*), Diretórios

de Assunto (*Subject Directories*) e Motores de Busca Especializados Baseados em Robô (*Robotic Specialized Search Engines*) (Peterson, 1997). No entanto, essas cinco abordagens podem ser, de forma geral, classificadas em apenas dois grandes grupos:

- Motores de Busca (*Search Engines*): no qual estão as abordagens Motores de Busca Baseados em Robô, Mega-Indexadores, Mega-Indexadores Simultâneos e Motores de Busca Especializados Baseados em Robô. Alguns exemplos de Motores de Busca são: Altavista¹, Excite², Lycos³, The Open Text Index⁴, WebCrawler⁵, Galaxy⁶.
- Diretórios WWW (*WWW Directories*): no qual está apenas a abordagem Diretórios de Assuntos. O exemplo mais conhecido nesse grupo é o Yahoo⁷.

Trabalhos relacionados à coleta de documentos provenientes da internet podem ser encontrados na literatura (Baeza-Yates, 1998; Joachims, Freitag, & Mitchell, 1997). Muitos deles combinam técnicas de AM e Recuperação de Informação (RI) (Van Rijsbergen, 1979) para determinar o perfil do usuário visando melhorar a coleta de documentos. Após a coleta de documentos é necessário transformar os documentos em um formato propício para serem submetidos aos algoritmos de extração automática de conhecimento. Essa segunda etapa, denominada de pré-processamento, é responsável por obter uma representação estruturada dos documentos, geralmente no formato de uma tabela atributo-valor.

Essa tabela atributo-valor que representa os documentos tem como característica valores esparsos dos dados e uma alta dimensionalidade. Essas características são inerentes a problemas relacionados ao processo de MT, pois cada palavra presente nos documentos pode ser um possível elemento do conjunto de atributos dessa tabela atributo-valor. É, portanto, uma etapa bastante custosa e um cuidadoso pré-processamento dos documentos é imprescindível ao sucesso de todo o processo de MT.

Após os documentos serem representados em um formato adequado, é possível aplicar técnicas de extração de conhecimento utilizando sistemas de aprendizado. Nessa terceira etapa, o objetivo é descobrir padrões úteis e desconhecidos presentes nos documentos. Caso os documentos estejam representados no formato de uma tabela atributo-valor, geralmente, são empregados métodos de RI e de AM para extração de padrões de

¹<http://www.altavista.com>

²<http://www.excite.com>

³<http://www.lycos.com>

⁴<http://index.opentext.com>

⁵<http://webcrawler.com>

⁶<http://www.galaxy.com>

⁷<http://www.yahoo.com>

forma semelhante ao processo tradicional de MD. Alguns trabalhos relacionados à MT, referenciados na literatura, que utilizam diferentes algoritmos de aprendizado podem ser encontrados em (Aas & Eikvil, 1999; Li & Jain, 1998; Moulinier & Ganascia, 1996; Li & Yamanishi, 1999; Cohen & Hirsh, 1998; Cohen & Singer, 1996; Apté, Damerau, & Weiss, 1994; Yang & Liu, 1999; Lagus, 2000; Thomas & Sycara, 1999; Joachims, 1997).

A etapa de avaliação é a quarta e última etapa do processo de MT. Essa etapa é necessária para verificar se o objetivo foi alcançado ou se necessitam ser refeitas todas ou algumas das etapas do processo. A verificação pode ser realizada pelo usuário final, especialista do domínio e/ou analista de dados, na qual os resultados obtidos são analisados para verificar se condizem com o objetivo esperado. Diversas ferramentas de visualização podem ser usadas para identificar e remover padrões redundantes e inúteis à aplicação final (Fayyad, Grinstein, & Wierse, 2002; Rezende, Oliveira, Félix, & Rocha, 1998). Porém, abordagens (medidas) estatísticas usando as medidas de *recall*, precisão e *F-measure* são freqüentemente utilizadas para avaliar o resultado obtido pelos algoritmos de categorização. Essas medidas são apresentadas na Seção 3.3 na página 44.

A seguir é apresentada em maiores detalhes a etapa de pré-processamento de textos, um dos focos de estudo desta tese.

3.2 O Pré-processamento de Textos

Considerando que a etapa de coleta de documentos tenha sido cumprida e, portanto, os documentos estejam disponíveis, é necessário realizar o pré-processamento desses documentos. A etapa de pré-processamento tem um alto custo computacional e exige um cuidadoso planejamento e processamento para obter um bom desempenho do processo de MT.

Diversos procedimentos para a representação dos documentos podem ser adotados, desde uma representação mais simples usando a técnica *bag-of-words*, na qual cada documento é representado como um vetor composto de palavras presentes no texto, até representações mais complexas. Neste trabalho, não são utilizadas essas representações mais complexas, as quais podem ser consultadas na literatura. Entre elas podemos citar: frases ou sentenças usando técnicas de Processamento de Língua Natural (PLN) (Jacobs, 1992; Scott & Matwin, 1999; Basili, Moschitti, & Pazienza, 2000), clusters de palavras (Baker & McCallum, 1998), grafos conceituais (Montes-y-Gómez, Gelbukh, & López-López, 2002), *string kernels* (Lodhi, Shawe-Taylor, Cristianini, & Watkins, 2002) e outras estatísticas de palavras (Caropreso, Matwin, & Sebastiani, 2001).

Deve ser ressaltado que a abordagem *bag-of-words* é um dos procedimentos mais usados (Bekkerman, El-Yaniv, Tishby, & Winter, 2003). Resultados experimentais mostram que representações mais sofisticadas, às vezes, perdem em desempenho com relação a representação 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 o uso de representações mais sofisticadas tenham qualidade semântica superior, a qualidade estatística é inferior em relação a representações baseadas em palavras simples. Assim, de acordo com Joachims (2002), a abordagem *bag-of-words* é uma boa relação entre expressividade e complexidade do modelo. Enquanto representações mais expressivas capturam melhor o significado do documento, sua complexidade é maior e degrada a qualidade de modelos estatísticos. Considerando que a abordagem *bag-of-words* representa os documentos de forma acessível à maioria dos algoritmos de aprendizado, foi decidido adotar essa abordagem no desenvolvimento deste trabalho.

Uma outra questão a ser considerada na representação dos documentos é que, para a maioria dos algoritmos de aprendizado, o número de documentos necessários para assegurar um bom resultado, está relacionado com a quantidade de atributos que descreve cada documento. Em tarefas de categorização de documentos, no entanto, essa é uma situação paradoxal pois, geralmente, o número de documentos (exemplos) de treinamento disponível é menor do que o número de possíveis atributos. Dessa forma, a etapa de pré-processamento em MT consiste em, além de transformar os documentos em um formato estruturado, reduzir a dimensionalidade dessa representação. Pois, uma alta dimensionalidade pode tornar o custo de armazenamento e processamento algumas vezes impraticável. É necessário, portanto, desenvolver e utilizar métodos que auxiliem nessa etapa com o objetivo de obter um melhor desempenho na próxima etapa de extração de padrões (Forman, 2003) — Figura 3.1 na página 33.

A seguir são apresentadas a representação de documentos usando a abordagem *bag-of-words*, diversas medidas para atribuição de valores aos atributos e algumas das técnicas utilizadas para reduzir a dimensionalidade do conjunto de atributos.

3.2.1 Representação de Documentos

Dado um conjunto de N documentos $D = \{d_1, d_2, \dots, d_N\}$ e um conjunto de N_{cl} categorias $C = \{c_1, c_2, \dots, c_{N_{cl}}\}$ associado aos documentos, é possível realizar tarefas de categorização de documentos. Cada documento d_i é representado como um vetor de M termos (palavras) que ocorrem no documento. Cada termo pode ser representado por

uma palavra simples (*1-gram*) ou composta ($2, 3, \dots, n$ -*gram*) que ocorre no documento⁸.

Os documentos podem então ser representados usando o mesmo formato da tabela atributo-valor apresentado na Tabela 2.1 na página 15. Nesse caso, cada documento d_i corresponde a um exemplo E_i e cada termo t_j corresponde a um atributo A_j na Tabela 2.1. A tabela atributo-valor, correspondente à representação de documentos, é ilustrada na Tabela 3.1.

Tabela 3.1: Representação de documentos

	t_1	t_2	$\dots$	t_M	Y
d_1	a_{11}	a_{12}	$\dots$	a_{1M}	y_1
d_2	a_{21}	a_{22}	$\dots$	a_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
d_N	a_{N1}	a_{N2}	$\dots$	a_{NM}	y_N

Mais especificamente, na Tabela 3.1 estão N documentos (exemplos) representados por M termos, e cada documento d_i é um vetor na forma

$$d_i = (\vec{a}_i, y_i)$$

no qual $\vec{a}_i = (a_{i1}, a_{i2}, \dots, a_{iM})$. O valor a_{ij} representa a influência relativa do atributo t_j no documento d_i do conjunto de documentos. Para calcular cada um dos valores de a_{ij} , usualmente, são utilizados os seguintes três componentes (Salton & Buckley, 1988):

1. o componente do *documento*, relacionado à estatística de um termo estar presente em um documento particular;
2. o componente da *coleção*, relacionado à estatística de um termo estar presente no conjunto de documentos;
3. o componente de *normalização*, relacionado ao ajuste de pesos para que documentos grandes e pequenos possam ser comparados na mesma escala.

Assim, para calcular o valor de cada termo t_j em cada documento d_i , várias combinações podem ser utilizadas na definição das medidas considerando esses componentes. A mais simples é a representação binária. Na representação binária, o valor a_{ij} é igual a

⁸O termo *n-gram* é usado na literatura de duas formas distintas. A primeira forma, adotada neste trabalho, é usada para indicar o conjunto de n **palavras** que ocorrem sequencialmente no texto. A segunda forma é usada para indicar um conjunto de n **caracteres** que ocorrem sequencialmente em um texto. Ainda, a palavra *gram* deve ser usada sempre no singular.

1 se o termo t_j ocorre no documento d_i e igual a 0 caso contrário, para $j \in \{1, \dots, M\}$ e $i \in \{1, \dots, N\}$ — Equação 3.1.

$$a_{ij} = \begin{cases} 1, & \text{se } t_j \text{ ocorre em } d_i \\ 0, & \text{caso contrário.} \end{cases} \quad (3.1)$$

A atribuição de valores para os termos usando a representação binária nem sempre é adequada. Assim, o componente do documento é empregado levando em consideração a frequência que um termo aparece no documento. A medida *term frequency* — *tf* — é uma medida que utiliza o número de ocorrências de t_j em d_i . A idéia é que os termos que mais ocorrem no documento são mais relevantes que os termos menos frequentes. Nesse caso, é atribuído a a_{ij} o valor $tf(t_j, d_i)$, o qual representa o número de vezes que o termo t_j ocorre no documento d_i — Equação 3.2.

$$a_{ij} = tf(t_j, d_i) \quad (3.2)$$

No entanto, um termo muito frequente também pode ocorrer em quase todo o conjunto de documentos. Quando isso ocorre, esses termos não são úteis para uma boa discriminação das categorias. O componente da coleção é usado para dar um peso menor para tais termos usando a medida *inverse document frequency* — *idf* — definida pela Equação 3.3.

$$idf = \log \frac{N}{x} \quad (3.3)$$

A medida *idf* favorece termos que aparecem em poucos documentos do conjunto. *idf* varia inversamente ao número de documentos x que contêm o termo t_j em um conjunto de documentos. Assim, as medidas *tf* e *idf* podem ser combinadas em uma nova medida denominada *tfidf*. O valor de a_{ij} pode então ser calculado pela Equação 3.4.

$$a_{ij} = tfidf(t_j, d_i) = tf(t_j, d_i) \times \log \frac{N}{x} \quad (3.4)$$

O componente de normalização é utilizado principalmente para ajustar os pesos dos atributos para que tanto documentos pequenos quanto documentos maiores possam ser comparados na mesma escala. Em muitas situações, documentos pequenos são representados por poucos termos, enquanto que os documentos maiores, geralmente, por muitos termos. Quando uma grande quantidade de termos é usada na representação de documentos, a probabilidade do termo pertencer a um documento é alta e, assim, documentos

maiores tem melhores chances de serem relevantes do que documentos menores. Normalmente, todos os documentos relevantes à tarefa deveriam ser tratados com a mesma importância independente do seu tamanho. Um fator de normalização, nesse caso, deve ser incorporado. Os valores a_{ij} podem então ser normalizados de diversas formas, tais como a definida pela Equação 3.5, aqui denominada de *tfidf**n* para a medida *tfidf*.

$$a_{ij} = \text{tfidf}n(t_j, d_i) = \frac{\text{tfidf}(t_j, d_i)}{\sqrt{\sum_{s=1}^N (\text{tfidf}(t_s, d_i))^2}} \quad (3.5)$$

O comportamento dessas medidas em um conjunto de documentos é ilustrado na Seção 5.2 na página 74. As medidas apresentadas são bastante utilizadas para atribuir valores aos termos presentes nos documentos. No entanto, na criação da tabela atributo-valor, cada termo é um possível elemento do conjunto de atributos, devendo ser levado em consideração, além da medida a ser utilizada, um critério de redução de dimensionalidade dos atributos. A seguir são apresentadas algumas técnicas para reduzir a dimensão do conjunto de atributos para dados textuais.

3.2.2 Redução da Dimensionalidade dos Atributos

A alta dimensionalidade do conjunto de atributos é uma das características do processo de MT. De fato, facilmente o espaço de representação pode ter dimensão 30000 ou maior. É notório, portanto, a necessidade de reduzir o espaço de atributos para uma melhor representatividade dos dados e melhor desempenho do processo de MT. Essa redução é também chamada de seleção de atributos relevantes.

A remoção de atributos irrelevantes e inapropriados ajuda a minimizar problemas como *overfitting* (Joachims, 2002) — Seção 2.2.1 na página 17. Além disso, melhora a eficiência computacional, considerando que muitos algoritmos de aprendizado não trabalham com espaço de representação muito grande. Vários métodos podem ser utilizados a fim de reduzir a quantidade de atributos em categorização de textos. Entre outros, a eliminação de *stopwords*, que são palavras irrelevantes na discriminação do conteúdo de documentos em uma dada língua, a transformação de cada termo para o radical que o originou e algumas técnicas estatísticas. A seguir, são apresentados alguns dos métodos mais utilizados em MT para reduzir a dimensionalidade dos atributos.

3.2.2.1 Eliminação de *Stopwords* e de Termos Pouco Frequentes

A eliminação de *stopwords* é um método aplicado exclusivamente a textos. Termos (palavras) que são preposições, artigos, conjunções, podem ser consideradas irrelevantes para a tarefa independentemente da tarefa a ser realizada. Porém, os termos que são *stopwords* são dependentes da língua. As *stopwords* são termos muito comuns e ocorrem com alta frequência na língua considerada. Dessa forma, o fato de eliminá-las do vetor de atributos faz com que haja uma drástica redução no conjunto de atributos.

De outra maneira, diferente da eliminação de *stopwords* que são termos muito comuns, um limiar de frequência de documentos (DF⁹) (Yang & Pedersen, 1997) elimina termos pouco frequentes. Todos os termos que ocorrem em poucos documentos, dado um certo limiar, são eliminados. DF é baseado na hipótese que termos pouco frequentes não são informativos ou não têm influência suficiente para contribuir com informação útil devido a sua baixa ocorrência (Apté, Damerau, & Weiss, 1994).

3.2.2.2 Algoritmos de *Stemming*

Algoritmos de *stemming* consistem em uma normalização lingüística, na qual as formas variantes de um termo são reduzidas a uma forma comum denominada *stem*¹⁰. Na maioria dos casos essas variantes têm interpretações semânticas similares e podem ser tratadas como equivalentes em aplicações de recuperação de informação e mineração de textos. A consequência da aplicação de algoritmos de *stemming* consiste na remoção de prefixos ou sufixos de um termo, ou mesmo na transformação de um verbo para sua forma no infinitivo. Assim, os seguintes termos:

representar
representado
representando
representação
representações

são termos com significados similares que podem ser reduzidos a um único termo, o *stem represent*. Os algoritmos de *stemming* mais conhecidos são os algoritmos do Porter (1980) e o algoritmo do Lovins (1968) que removem sufixos de termos. O algoritmo do Porter remove sufixos de termos em inglês e tem sido amplamente usado, referenciado e adaptado nos últimos vinte anos. Diversas implementações do algoritmo estão disponibi-

⁹ Document Frequency thresholding.

¹⁰ Neste trabalho, *stem* refere-se ao resíduo de um algoritmo de *stemming* aplicado a uma palavra e não necessariamente à raiz etimológica da qual a palavra deriva.

lizadas na *Web*, entre elas a página oficial escrita e mantida pelo autor para distribuição do seu algoritmo (<http://www.tartarus.org/~martin/PorterStemmer>).

Apesar dos algoritmos de *stemming* aplicados a termos de textos reduzirem significativamente a dimensionalidade do conjunto de atributos, alguns problemas inerentes ao algoritmo, e à própria natureza das palavras, podem ocorrer. A maioria dos algoritmos de *stemming* não consideram um léxico associado e, portanto, ignoram o significado dos termos conduzindo a alguns erros, tais como, termos com significados diferentes são reduzidos a um mesmo *stem*, nesse caso ocorre um erro de *over-stemming*, por exemplo, as palavras **barato** e **barata** podem ser reduzidas a um mesmo *stem* **barat**. Já termos com significados similares quando reduzidos a *stems* diferentes conduzem ao erro de *under-stemming*. Por exemplo, os termos **viagem** e **viajar** podem ter aos *stems* diferentes **viag** e **viaj**, respectivamente.

Um outro tipo de erro, também muito comum, é denominado de *mis-stemming*. Nesse caso, o erro consiste em retirar o final de uma palavra, considerando que é o sufixo da palavra. Por exemplo, a palavra **lápiz** poderia ser reduzida ao *stem* **lapi**, dependendo de como o plural das palavras são tratados pelo algoritmo de *stemming*. Trabalhos relacionados a avaliação de erros obtidos por algoritmos de *stemming* podem ser encontrados em (Paice, 1996).

Apesar de algoritmos de *stemming* serem aplicados com o objetivo de tentar melhorar o desempenho de sistemas de **RI** e **MT**, inicialmente, vários estudos mostraram que, praticamente, não havia melhora significativa pelo uso de *stemming* aplicado a um conjunto de textos (Harman, 1991; Lennon, Pierce, Tarry, & Willett, 1981). No entanto, o trabalho de Krovetz (1995) mostrou melhoramentos, que embora algumas vezes pequenos, são significativos usando *stem* em um amplo conjunto de textos. Na realidade, foi apresentado que o grau de melhoramento varia consideravelmente entre conjuntos diferentes de documentos e, ainda, a medida que o algoritmo se torna mais específico na tentativa de minimizar a quantidade de *stems* diferentes para palavras com um mesmo radical, a eficiência do algoritmo degrada (Porter, 1980; Imamura, 2001). E, sem nenhuma dúvida, se algoritmos de *stemming* não melhoram o desempenho do classificador, com certeza diminuem o espaço de representação dos dados.

No Apêndice A na página 155 é apresentado em maiores detalhes o algoritmo do Porter para o inglês, português e espanhol, utilizados neste trabalho.

3.2.2.3 A Lei de Zipf e os Cortes de Luhn

Um outro método usado para reduzir a dimensionalidade dos atributos, na busca por termos que são mais representativos na discriminação dos documentos, utiliza a Lei de Zipf. A Lei de Zipf descreve uma maneira de descobrir termos considerados pouco representativos em um determinado conjunto de documentos. A lei, formulada por George Kingsley Zipf professor de lingüística em Harvard (1902-1950), declara que a frequência de ocorrência de alguns eventos está relacionada a uma função de ordenação. Zipf mostrou que a frequência de palavras em língua natural, populações das cidades, distribuição de dados em um banco de dados e muitos outros fenômenos humanos e naturais, seguem uma distribuição similar, a qual denominou de “*Principle of Least Effort*” (Zipf, 1949).

A Lei de Zipf aplicada a um conjunto de documentos é um procedimento simples que consiste em pegar todos os termos no conjunto de textos e contar o número de vezes que cada termo aparece. Se o histograma resultante for ordenado de forma decrescente, ou seja, o termo que ocorre mais frequentemente aparece primeiro, então a forma da curva é a “curva de Zipf” para aquele conjunto de documentos. A curva de Zipf, se for plotada em uma escala logarítmica, aparece como uma reta com inclinação -1, embora Gelbukh & Sidorov (2001) tenham mostrado que a curva de Zipf é dependente da língua e do estilo. A Lei de Zipf para documentos de língua natural pode ser aplicada não apenas aos termos mas, também, a frases e sentenças.

Na realidade, a lei de Zipf é uma observação empírica que se aplica em diversos domínios apresentando a seguinte distribuição:

$$p_1 = \frac{c}{1}, p_2 = \frac{c}{2}, \dots, p_n = \frac{c}{n}, \quad (3.6)$$

na qual $c = \frac{1}{H_n}$ e $H_n = \sum_{i=1}^n \frac{1}{i}$ (Knuth, 1973). Ou seja, considerando um conjunto de documentos escritos em língua natural, foi observado que o i -ésimo termo mais comum ocorre com frequência inversamente proporcional a i . Por exemplo, para um determinado conjunto de documentos, se a palavra com maior frequência ocorre 12000 vezes ($\frac{12000}{1}$), a segunda palavra com maior frequência ocorrerá aproximadamente 6000 vezes ($\frac{12000}{2}$), a terceira ocorrerá aproximadamente 4000 vezes ($\frac{12000}{3}$), e assim por diante, até a palavra na n -ésima posição que ocorrerá aproximadamente ($\frac{12000}{n}$) vezes.

Enquanto Zipf verificou sua lei utilizando jornais escritos em inglês, Luhn (1958) usou a lei como uma hipótese nula para especificar dois pontos de corte, os quais denominou de superior e inferior, para excluir termos não relevantes. Os termos que excedem o corte superior são os mais frequentes e são considerados comuns por aparecer 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 significativamente na discriminação dos documentos. Na Figura 3.2 é ilustrada¹¹ a curva de Zipf (I) e os cortes de Luhn aplicados à curva de Zipf (II), no qual o eixo cartesiano f representa a frequência das palavras e o eixo cartesiano r as palavras ordenadas pela frequência.

Figura 3.2: A curva de Zipf e os cortes de Luhn

Assim, Luhn propôs uma técnica para encontrar termos relevantes, assumindo que os termos mais significativos 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 termos mas, também, pode ser aplicada a palavras ou sentenças dos documentos.

3.2.2.4 Outras Técnicas de Seleção de Atributos

Na redução da dimensionalidade de atributos pode também ser utilizado um *thesaurus* para agrupar palavras em categorias semânticas, como por exemplo WordNet (Miller, Beckwith, Fellbaum, Gross, & Miller, 1990). No entanto, é freqüente a utilização de várias outras técnicas estatísticas que visam a redução da dimensionalidade dos atributos, tais como: *latent semantic indexing* (Deerwester, Dumais, Landauer, Furnas, & Harshman, 1990), informação mútua (Cover & Thomas, 1991; Quinlan, 1986), testes χ^2 (Mood, Graybill, & Boes, 1974), *odds ratio* (Rijsbergen, Harper, & Porter, 1981), agrupamento de

¹¹A curva de Zipf quando aplicada a um conjunto de documentos, geralmente, é uma curva hiperbólica mais acentuada.

termos¹² (Cheeseman, Stutz, Self, Kelly, Taylor, & Freeman, 1988), e outros (Pila, 2000; Lee, 2000). Maiores detalhes dessas técnicas podem ser encontrados em (Joachims, 2002; Yang & Pedersen, 1997; Aas & Eikvil, 1999; Mladenic, 1998).

3.3 Medidas de Avaliação

No caso de utilizar algoritmos de AM na etapa de extração de padrões, no processo de mineração de textos — Figura 3.1 na página 33 —, uma questão importante é como avaliar e comparar os classificadores induzidos por diferentes algoritmos de aprendizado, ou pelo mesmo algoritmo mas com diferentes parâmetros de entrada, dado um conjunto de documentos. Essa questão é ainda uma questão difícil, primeiro porque não existe um grande repositório de dados consolidados como o disponível na University of California at Irvine (UCI)¹³, para realização de experimentos em dados textuais. Segundo, porque alguns conjuntos de dados textuais disponíveis à comunidade, como o conjunto de textos Reuters-21578¹⁴, tem pelo menos cinco versões diferentes. Assim, nem sempre está claro qual versão e, conseqüentemente, qual conjunto de documentos foi utilizado nos resultados publicados na literatura. Esse problema foi ilustrado por Yang (1999), que realizou um estudo comparativo dos resultados obtidos utilizando as diversas versões do conjunto de textos Reuters.

Outra questão importante na avaliação de resultados de experimentos é a comparação entre diferentes medidas de desempenho usadas em experimentos individuais. Medidas como *recall*, precisão, acurácia e *F-measure*, são bastante citadas na literatura para avaliação de classificadores de textos. Essas medidas fornecem diversas informações as quais, muitas vezes, são complementares entre si.

Pode-se definir essas medidas usando estimadores baseados na matriz de confusão do classificador, que possui nas linhas os resultados esperados de um classificador e nas colunas os resultados preditos. Na Tabela 3.2 é ilustrada uma matriz de confusão de um classificador binário cujas classes (categorias) são positiva C_+ e negativa C_- , do conceito.

Tabela 3.2: Matriz de confusão de um classificador binário

Classe	Predita C_+	Predita C_-
Verdadeira C_+	T_P	F_N
Verdadeira C_-	F_P	T_N

¹² *Term clustering*.

¹³ <http://www.ics.uci.edu/~mllearn/MLRepository.html>.

¹⁴ The Reuters-21578, 1.0, <http://www.research.att.com/~lewis/reuters21578.html>.

Na tabela são representados os quatro possíveis resultados preditos pelo classificador (hipótese **h**) induzido pelo algoritmo de aprendizado, para cada categoria $y \in \{C_+, C_-\}$ do conceito. Os valores T_P e T_N representam o número de predições corretas enquanto que os valores F_P e F_N representam as predições erradas. Mais especificamente:

- T_P é o número de documentos *corretamente designado* para a categoria;
- F_P é o número de documentos *incorretamente designado* para a categoria;
- F_N é o número de documentos *incorretamente rejeitado* para a categoria;
- T_N é o número de documentos *corretamente rejeitado* para a categoria.

A partir desses valores da matriz de confusão do classificador, as medidas de desempenho *recall* (r), *precisão* (p), *fallout* (f), *acurácia* (acc) e *erro* (err), podem ser definidas pelas Equações 3.7, 3.8, 3.9, 3.10 e 3.11, respectivamente.

$$r = \frac{T_P}{T_P + F_N} \quad (3.7)$$

$$p = \frac{T_P}{T_P + F_P} \quad (3.8)$$

$$f = \frac{F_P}{F_P + T_N} \quad (3.9)$$

$$acc = \frac{T_P + T_N}{T_P + F_P + F_N + T_N} \quad (3.10)$$

$$err = \frac{F_P + F_N}{T_P + F_P + F_N + T_N} = 1 - acc \quad (3.11)$$

Para avaliar o desempenho dos classificadores geralmente são empregados dois métodos denominados *média-macro*¹⁵ e *média-micro*¹⁶. Para facilitar o entendimento do cálculo dessas medidas, considere um conjunto de 100 documentos sendo que cada documento d_i pertence a uma das quatro categorias (classes) $C = \{C_{\clubsuit}, C_{\heartsuit}, C_{\spadesuit}, C_{\diamondsuit}\}$. A categoria $C_{\clubsuit}$ contém 10 exemplos, a categoria $C_{\heartsuit}$ contém 20 exemplos, a categoria $C_{\spadesuit}$ contém 45 exemplos e a categoria $C_{\diamondsuit}$ contém 25 exemplos. Uma matriz de confusão para as quatro categorias é representada na Tabela 3.3.

¹⁵ *Macro-averaging*.

¹⁶ *Micro-averaging*.

Tabela 3.3: Matriz de confusão para as categorias $C_{\clubsuit}, C_{\heartsuit}, C_{\spadesuit}, C_{\diamondsuit}$

Classe		Predita			
		$C_{\clubsuit}$	$C_{\heartsuit}$	$C_{\spadesuit}$	$C_{\diamondsuit}$
Verdadeira	$C_{\clubsuit}$	10	0	0	0
	$C_{\heartsuit}$	1	17	2	0
	$C_{\spadesuit}$	2	0	43	0
	$C_{\diamondsuit}$	0	0	0	25

Os métodos média-macro e média-micro utilizam os valores da matriz de confusão para avaliar o desempenho geral do classificador. Para calcular o desempenho média-macro, primeiramente devem ser calculadas as medidas de desempenho separadamente para cada categoria, usando os valores da matriz de confusão, semelhantes à Tabela 3.2 na página 44. Por exemplo, para a categoria $C_{\clubsuit}$ a matriz de confusão, obtida a partir da Tabela 3.3, é mostrada na Tabela 3.4.

Tabela 3.4: Matriz de confusão para a categoria $C_{\clubsuit}$

Classe	Predita $C_{+\clubsuit}$	Predita $C_{-\clubsuit}$
Verdadeira $C_{+\clubsuit}$	10	0
Verdadeira $C_{-\clubsuit}$	3	87

Dessa forma, é possível calcular o valor de cada uma das medidas *recall*, precisão, *fallout*, acurácia erro para a categoria $\clubsuit$.

$$r = \frac{10}{10} = 1 \qquad p = \frac{10}{13} = 0,77 \qquad f = \frac{3}{90} = 0,03$$

$$acc = \frac{97}{100} = 0,97 \qquad err = 1 - 0,97 = 0,03$$

De forma semelhante, esses valores podem ser calculados para as outras categorias considerando a matriz de confusão para cada uma das categorias $\heartsuit, \spadesuit, \diamondsuit$ — Tabelas 3.5, 3.6 e 3.7.

Tabela 3.5: Matriz de confusão para a categoria $C_{\heartsuit}$

Classe	Predita $C_{+\heartsuit}$	Predita $C_{-\heartsuit}$
Verdadeira $C_{+\heartsuit}$	17	3
Verdadeira $C_{-\heartsuit}$	0	80

Tabela 3.6: Matriz de confusão para a categoria $C_{\spadesuit}$

Classe	Predita $C_{+\spadesuit}$	Predita $C_{-\spadesuit}$
Verdadeira $C_{+\spadesuit}$	43	2
Verdadeira $C_{-\spadesuit}$	2	53

Tabela 3.7: Matriz de confusão para a categoria $C_{\diamond}$

Classe	Predita $C_{+\diamond}$	Predita $C_{-\diamond}$
Verdadeira $C_{+\diamond}$	25	0
Verdadeira $C_{-\diamond}$	0	75

As medidas de desempenho para cada uma das quatro categorias, calculadas a partir das matrizes de confusão correspondentes dessas categorias são mostradas na Tabela 3.8.

Tabela 3.8: Medidas de desempenho para cada categoria: $C_{\clubsuit}, C_{\heartsuit}, C_{\spadesuit}, C_{\diamond}$

	<i>recall</i>	precisão	<i>fallout</i>	acurácia	erro
$C_{\clubsuit}$	1	0,77	0,03	0,97	0,03
$C_{\heartsuit}$	0,85	1	0	0,97	0,03
$C_{\spadesuit}$	0,96	0,96	0,04	0,96	0,04
$C_{\diamond}$	1	1	0	1	0

Com esses valores das medidas obtidos por categoria, calcula-se o desempenho média-macro com a média aritmética desses valores. Os resultados finais são mostrados na Tabela 3.9. Como pode ser observado, a média-macro refere-se à média por categoria considerando o mesmo peso para cada categoria, independentemente do número de documentos por categoria.

Tabela 3.9: Medidas de desempenho média-macro do classificador

	<i>recall</i>	precisão	<i>fallout</i>	acurácia	erro
Média-macro	0,95	0,93	0,02	0,98	0,02

O desempenho média-micro considera peso igual para todos os documentos. Para calcular o desempenho média-micro, primeiramente contabilizam-se todos os valores T_P , F_N , F_P , T_N de cada categoria e somam-se os valores correspondentes obtendo uma matriz de confusão geral, como mostrado na Tabela 3.10.

Tabela 3.10: Matriz de confusão geral

Classe	Predita C_+	Predita C_-
Verdadeira C_+	$10 + 17 + 43 + 25 = 95$	$0 + 3 + 2 + 0 = 5$
Verdadeira C_-	$3 + 0 + 2 + 0 = 5$	$87 + 80 + 53 + 75 = 295$

Após, cada medida de desempenho média-micro é calculada usando os valores dessa matriz de confusão geral. Os resultados obtidos são apresentados na Tabela 3.11.

Tabela 3.11: Medidas de desempenho média-micro do classificador

	<i>recall</i>	precisão	<i>fallout</i>	acurácia	erro
Média-micro	0,95	0,95	0,02	0,98	0,02

As medidas média-micro e média-macro podem obter resultados muito diferentes, especialmente se as categorias não são balanceadas. De acordo com [Sebastiani \(2002\)](#), apesar da maioria dos pesquisadores terem uma certa tendência em utilizar a média-micro, não existe um consenso entre os pesquisadores a respeito de qual medida é a mais adequada.

No entanto, essas medidas quando usadas isoladamente, às vezes, não refletem de forma adequada o desempenho do classificador. Usualmente, o classificador apresenta um compromisso entre *recall* e precisão quando um limiar de decisão no classificador é ajustado pois para obter um alto *recall*, usualmente, sacrifica-se a precisão e vice-versa. Se o *recall* e precisão são ajustados para terem o mesmo valor, então esse valor é denominado de *ponto de equilíbrio* (BEP¹⁷), também utilizado na avaliação de classificadores de textos.

Outro critério de avaliação que combina *recall* e precisão é a medida *F-measure*, definida pela Equação 3.12, na qual β é um parâmetro que permite diferenciar os pesos de *recall* e precisão, representando a importância relativa de cada um. Se o valor de β é maior que 1, isso indica uma importância maior para a precisão, enquanto um valor menor que 1 indica uma ênfase maior para *recall*.

$$F_\beta = \frac{(\beta^2 + 1) \times p \times r}{\beta^2 \times p + r} \quad (3.12)$$

Quando o valor de β é igual a 1, a medida é denominada F_1 — Equação 3.13. A medida F_1 , definida por [Van Rijsbergen \(1979\)](#), equilibra *recall* e precisão de maneira que tenham peso igual e, geralmente, é usada como um critério de otimização. Seu valor é

¹⁷ Break-even point.

maximizado quando os valores de *recall* e precisão são iguais ou próximos. Caso contrário, o menor valor entre *recall* e precisão domina o valor de F_1 .

$$F_1 = \frac{2 \times p \times r}{p + r} \quad (3.13)$$

Para o exemplo usando os valores obtidos com o método média-macro — Tabela 3.9 — o valor de F_1 é 0,94. Já para os valores obtidos com o método média-micro — Tabela 3.11 — o valor de F_1 é 0,95. É interessante observar que o BEP é apenas um valor específico para a medida F_1 . Quando $r = p$, pela definição de BEP, $F_1 = \frac{2r^2}{2r} = r = p$. Assim, o valor de BEP é sempre igual ou menor do que o valor ótimo de F_1 . Isso restringe o tipo de comparação entre o resultado de um classificador usando a medida F_1 e um classificador usando BEP. Pode-se dizer que um classificador usando BEP tem melhor desempenho do que um classificador usando a medida F_1 , se o valor de BEP é maior do que o valor de F_1 , porém, o inverso não é necessariamente verdade (Yang, 1999).

3.4 Softwares de Mineração de Textos

Alguns softwares utilizados em aplicações de MT são apresentados na Tabela 3.12. Os softwares estão descritos principalmente em funções de análise de textos, como recuperação de informação, categorização e sumarização.

Tabela 3.12: Alguns sistemas de mineração de textos

Sistema	Pré-processamento	Tarefas	Software Livre
iMiner	<i>stemming</i> dicionário http://www-3.ibm.com/software/data/iminer/	agrupamento categorização sumarização	não
TextAnalyst	<i>stemming</i> <i>stopwords</i> dicionário http://www.megaputer.com	categorização sumarização agrupamento	sim
TextSmart	<i>stopwords</i> <i>stemming</i> dicionário http://www.spss.com/textsmart/	categorização	sim
Inxight	<i>stemming</i> recursos lingüísticos http://www.inxight.com/	categorização sumarização	não
Rainbow	<i>stemming</i> <i>stopwords</i> <i>n-gram</i> http://www-2.cs.cmu.edu/~mccallum/bow/	categorização agrupamento	sim

A Tabela 3.12 não é de forma alguma exaustiva, mas ela serve para ilustrar o tipo de pré-processamento e as tarefas realizadas pelos softwares utilizados em MT. Com relação aos softwares livres, pode-se perceber que o TextAnalyst é um sistema que envolve diversas tarefas: agrupamento, categorização e sumarização. Enquanto o TextSmart, por exemplo, envolve a tarefa de categorização. No entanto, ambos os softwares, na etapa de pré-processamento, utilizam técnicas de *stemming*, eliminação de *stopwords* e um dicionário. O software Rainbow utiliza, além das técnicas de *stemming* e eliminação de *stopwords*, a concatenação de palavras em *n-gram*. Esse software é utilizado em alguns experimentos do Capítulo 7 dessa tese.

3.5 Considerações Finais

Uma questão primordial em um processo de mineração de textos é a representação dos documentos, tratada neste capítulo. A representação binária para atribuição de valores aos atributos, apesar de simples e bastante utilizada, nem sempre é adequada. Entretanto, em documentos pequenos as palavras são improváveis de se repetir, tornando nesses casos a representação binária quase tão informativa quanto a frequência (Forman, 2003). Outras possibilidades para representar documentos incluem a frequência de um termo ocorrer no documento, a frequência inversa da palavra nos documentos e a frequência do termo normalizada pelo tamanho do documento. Em situações nas quais o tamanho do documento tem uma grande variação, pode ser importante realizar essa normalização.

As características relacionadas ao processo de MT consistem, principalmente, em transformar os documentos em uma boa representação estruturada e reduzir a alta dimensionalidade do espaço de representação dos atributos. A eliminação de palavras irrelevantes, como as *stopwords*, e aplicação de algoritmos de *stemming*, reduzem significativamente a quantidade de atributos. No entanto, o espaço de representação, ainda assim, é alto. Como ilustração desse problema, considere uma versão inicial desta tese sendo considerada como um conjunto de documentos, na qual cada capítulo é um documento. Após a retirada de *stopwords* e aplicação do algoritmo de *stemming* para o português, a cardinalidade do conjunto total de atributos que representa esse conjunto é de 2299. Em outras palavras, com somente esse pré-processamento simples, são necessários 2299 termos (*stems*) para representar uma versão preliminar desta tese. O gráfico com a curva de Zipf correspondente à frequência dos *stems* é mostrado na Figura 3.3¹⁸.

¹⁸Os *stems* gerados e as informações para criação do gráfico, foram obtidos utilizando a ferramenta computacional PRETEXT, descrita no Capítulo 5 na página 71.

Figura 3.3: A curva de Zipf obtida com os *stems* desta tese

Interpretação de Clusters Usando Aprendizado Indutivo

Os primeiros trabalhos relacionados a agrupamento de dados foram desenvolvidos principalmente na área de taxonomia numérica e análise de cluster ([Anderberg, 1973](#); [Stepp III & Michalski, 1986](#)). Esses métodos, basicamente, utilizam uma medida matemática de similaridade para agrupar os dados em clusters e são denominados métodos clássicos. As medidas utilizadas, geralmente, não levam em consideração qualquer propriedade global ou conceitos que caracterizam os clusters. Conseqüentemente, os clusters obtidos podem ser de difícil interpretação, pois não apresentam descrições conceituais simples. Assim, o objetivo deste capítulo é a investigação dos problemas relacionados a tarefas de agrupamento. A idéia é utilizar uma combinação de métodos de agrupamento clássico e algoritmos de aprendizado indutivo simbólico com o objetivo de descobrir e interpretar, com o auxílio do especialista, padrões especiais de um conjunto de dados não rotulados.

4.1 Metodologia de Interpretação de Clusters

Geralmente, um método de agrupamento pode ser visto como um processo subjetivo que agrupa exemplos baseado em alguma medida de similaridade. Um mesmo conjunto de dados, muitas vezes, é agrupado diferentemente por diversos algoritmos ou aplicações, ou até mesmo pelo próprio algoritmo. Essa subjetividade faz o processo de agrupamento difícil, pois um determinado algoritmo não é adequado para solucionar todo e qualquer problema relacionado a agrupamento de dados. Assim, para obter um método adequado a uma aplicação particular, é necessário verificar cuidadosamente as diferentes técnicas e

o domínio do problema a ser explorado (Halkidi, Vazirgiannis, & Batistakis, 2000).

Por outro lado, uma abordagem denominada agrupamento conceitual foi introduzida como uma nova forma de agrupar dados de forma que os clusters encontrados pudessem ser compreendidos como um grupo de exemplos (dados) que juntos representem um conceito. O agrupamento conceitual produz não apenas clusters, mas também descrições de conceitos caracterizando os clusters (Kubat, Bratko, & Michalski, 1998). Porém, o processo de descrição conceitual dos clusters utilizando essa abordagem não é uma tarefa fácil. De fato, pode ser mais difícil do que a própria geração dos clusters. Isso porque induzir descrições de categorias a partir dos exemplos é uma tarefa inferencial complexa.

Neste trabalho é proposta uma metodologia que utiliza algoritmos de aprendizado simbólico juntamente com métodos clássicos de agrupamento para exploração de dados não rotulados, visando a descoberta e interpretação de clusters com o auxílio do usuário/especialista.

4.1.1 A Metodologia Proposta

A metodologia proposta para auxiliar a tarefa de interpretação de clusters de forma semi-automática, ilustrada na Figura 4.1, é composta basicamente pelas seguintes quatro etapas:

Figura 4.1: Metodologia proposta

1. um conjunto de dados não rotulados, no formato atributo-valor — Tabela 2.1 na página 15, é submetido a um processamento a ser realizado por algum algoritmo de AM não supervisionado. Esse algoritmo é o responsável por descobrir clusters presentes no conjunto de dados;
2. o resultado obtido, os clusters encontrados, é processado para que os dados do conjunto original, ou um subconjunto desses dados, sejam rotulados com o cluster ao qual pertencem. Assim, é gerado um conjunto de dados com uma dimensão adicional, no qual o cluster é considerado o atributo classe desses dados;

3. o novo conjunto de dados, gerado na etapa anterior, possui as características necessárias para serem utilizados como entrada para algoritmos de AM supervisionado. Como o interesse é tentar explicar para o usuário/especialista os clusters previamente encontrados, a linguagem de descrição de conceitos (hipóteses) utilizada pelo algoritmo de AM supervisionado deve ser uma linguagem simbólica, tal como regras ou árvores de decisão. Assim, os clusters podem ser descritos, simbolicamente, por meio dessas linguagens;
4. finalmente, o conhecimento do especialista do domínio é de fundamental importância ao se tentar dar uma interpretação conceitual aos clusters, agora descritos utilizando um formalismo simbólico. Com a interpretação do especialista, é possível ter uma compreensão e uma “explicação” para os dados pertencentes a cada cluster encontrado.

Dependendo do resultado obtido na última etapa, o usuário/especialista pode decidir repetir o processo nas diferentes etapas — Figura 4.1 na página oposta — como:

- usar outro algoritmo de AM não supervisionado ou apenas mudar seus parâmetros de entrada — item (a);
- agrupar um ou mais clusters distintos — item (b);
- usar outro algoritmo de AM supervisionado ou apenas mudar seus parâmetros de entrada — item (c).

É importante observar que um algoritmo de agrupamento quase sempre encontrará clusters, quaisquer que sejam, em um conjunto de dados. No entanto, a qualidade de cada cluster nem sempre reflete um agrupamento adequado ou está relacionada com os objetivos do usuário. Ainda, os clusters não representam, necessariamente, uma descrição conceitual relacionada a uma “classe”. Dessa forma, dois ou mais clusters podem agrupar exemplos que se referem a um mesmo conceito. Isso explica a idéia de agrupar os clusters originais — item (b) Figura 4.1 —, encontrados por um algoritmo de AM não supervisionado, em um único cluster.

Por exemplo, considere que na etapa 1, o algoritmo de agrupamento encontrou quatro clusters C_1 , C_2 , C_3 e C_4 , como ilustrado na Figura 4.2-(a). Os exemplos de treinamento são vistos pelo algoritmo de agrupamento apenas como pontos no espaço de busca, os quais foram agrupados de acordo com algum critério de similaridade. A etapa 2, consiste em rotular cada exemplo com o cluster ao qual pertence. Após, na etapa 3, este

Figura 4.2: Explicação do processo de clusters

novo conjunto de treinamento é submetido a um algoritmo de AM supervisionado e, na etapa 4, o usuário deve analisar o conceito induzido.

Considerando que após a interpretação o usuário considera que os clusters C_1 e C_4 referem-se ao mesmo conceito positivo (+), e os clusters C_2 e C_3 ao mesmo conceito negativo (-) como apresentado na Figura 4.2-(b), o usuário pode então decidir juntá-los em apenas dois clusters denominados, por exemplo, $C_{(1,4)}$ e $C_{(2,3)}$, repetindo o processo a partir da etapa 2, Figura 4.1 item (b), usando agora como clusters os novos rótulos $C_{(1,4)}$ e $C_{(2,3)}$.

Como pode ser observado, o usuário participa do processo de interpretação após a construção dos clusters, ou previamente, fixando o número de clusters a serem encontrados pelo algoritmo de AM não supervisionado. Trabalhos realizados usando essa metodologia podem ser encontrados em (Martins & Monard, 2000; Martins, Monard, Haedo, & Matsudo, 2001; Martins, Monard, & Halembeck, 2002a,b; Martins, Godoy, Monard, Matsubara, & Amandi, 2003; Martins, Matsubara, Monard, & Lopes, 2003).

4.1.2 Implementação da Metodologia Proposta

Nesta seção é apresentada uma implementação particular da abordagem proposta, usando **AutoClass** e **See5** como algoritmos de AM supervisionado e não supervisionado, respectivamente — Figura 4.3.

Os exemplos não rotulados são dados de entrada para **AutoClass**, que procura a melhor classificação que possa encontrar e que seja altamente provável com os dados e modelos especificados pelo usuário. Uma classificação para **AutoClass** consiste em descobrir

Figura 4.3: Processo proposto para explicação de clusters

um conjunto de clusters, descrevendo qual a porcentagem provável dos exemplos estarem em cada cluster e uma denominação probabilística dos exemplos para esses clusters, ou seja, a probabilidade relativa do exemplo pertencer a cada cluster. A saída de **AutoClass** consiste de vários relatórios contidos em arquivos distintos — Seção 4.1.2.1 na próxima página.

Após descobertos os clusters, os exemplos são automaticamente rotulados, usando a ferramenta computacional **INCLASS** (Seção 4.2.1), com o cluster ao qual pertencem com maior probabilidade. **INCLASS** está programado para processar *strings* por meio de expressões regulares, que é a característica principal da linguagem Perl (Wall, Christiansen, & Schwartz, 1996). A entrada de dados para **INCLASS** é o conjunto de dados original e um dos relatórios gerados por **AutoClass**. A saída consiste de um novo conjunto de dados que contém os exemplos originais rotulados com o cluster ao qual pertencem.

O novo conjunto de dados, no qual cada exemplo tem uma “classe” associada, é então utilizado pelo **See5** a fim de encontrar uma descrição, expressa como regras **if-then**, para os exemplos pertencentes a um mesmo cluster. Finalmente, o conhecimento induzido por **See5** pode ser interpretado pelo usuário que pode tomar algumas decisões antes de repetir o processo em qualquer das quatro etapas, como explicado anteriormente — Figura 4.1 na página 54.

Deve ser salientado que quaisquer outros algoritmos de aprendizado podem ser utilizados. Neste trabalho, é explorado o algoritmo **AutoClass** como **AM** não supervisionado e **See5** como **AM** supervisionado, por serem algoritmos robustos e aceitos pela comunidade científica. Ambos algoritmos, **AutoClass** e **See5**, utilizam um formato similar atributo-valor

para representar os exemplos. No entanto, é importante ressaltar que os *bias* utilizados pelos dois algoritmos são muito diferentes. Enquanto o **AutoClass** é um algoritmo de aprendizado estatístico, o algoritmo de aprendizado utilizado por **See5** induz regras que separam o espaço de representação dos exemplos por meio de hiperplanos. Dessa forma, algoritmos que têm *bias* muito diferentes, quando utilizados de forma complementares, podem ou não obter o resultado esperado. Na realidade, espera-se que algoritmos com *bias* semelhantes tenham um melhor desempenho. Esse é um tema a ser investigado posteriormente ao trabalho realizado nesta tese.

A seguir são apresentados resumidamente as características dos algoritmos **AutoClass** e **See5**. Algumas ferramentas desenvolvidas para auxiliar o usuário na interpretação dos clusters são apresentadas nas próximas seções.

4.1.2.1 AutoClass

AutoClass¹ é um algoritmo de aprendizado não supervisionado baseado na teoria Bayesiana, desenvolvido pelo grupo de Bayes no Ames Research Center. **AutoClass** é um software robusto, de domínio público, bastante usado e citado na literatura. A teoria probabilística que suporta a implementação de **AutoClass** encontra-se bem documentada em (Hanson, Stutz, & Cheeseman, 1991; Cheeseman, Stutz, Self, Kelly, Taylor, & Freeman, 1988; Cheeseman & Stutz, 1990).

Basicamente, **AutoClass** descreve clusters a partir da distribuição probabilística sobre os atributos dos exemplos, considerando que existe independência condicional nos dados. A abordagem Bayesiana busca em um espaço do modelo a “melhor” descrição dos clusters, e não apenas particiona os exemplos como a maioria das técnicas de agrupamento fazem. Uma boa classificação busca o ponto entre precisão preditiva *versus* complexidade dos clusters sem se ajustar² demais aos dados. Os clusters descobertos são também *nebulosos*³, no sentido que cada exemplo tem uma probabilidade de pertencer a cada um dos diferentes clusters — técnica probabilística.

AutoClass tem várias características tais como: determinar o número de clusters automaticamente; usar dados com valores discretos e reais em um mesmo conjunto de dados; manipular valores ausentes; utilizar tempo de processamento, aproximadamente, linear à quantidade de exemplos. Ele possui três modos de execução, ilustrados nas Figuras 4.4 e 4.5, e descritos a seguir:

¹<http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass-c-program.html>

² *Overfitting*.

³ *Fuzzy*.

Figura 4.4: Esquema para busca **search** e relatório **report** Figura 4.5: Esquema de predição **predict**

Busca (search) consiste em agrupar os exemplos colocando os resultados em três arquivos distintos com extensões `.search`, `.results[-bin]` e `.log`, como ilustrado na parte central na Figura 4.4. O arquivo `.search` contém registros das tentativas da busca, enquanto que o arquivo `.results[-bin]` contém as “melhores” classificações encontradas por **AutoClass**. Já o arquivo `.log` contém a informação do que foi impresso na tela durante a execução.

Relatórios (reports) os arquivos de saída do modo de busca `.search` e `.results[-`

`bin]` são necessários para a execução de **AutoClass** no modo de relatórios. Nesse modo, por sua vez, são criados mais quatro arquivos `.class-text-n`, `.case-text-n`, `.influ-o-text-n` e `.rlog`, como ilustrado a direita da Figura 4.4. Os arquivos `.class-text-n` e `.case-text-n` são relatórios descrevendo os exemplos nos clusters descobertos. O arquivo `.influ-o-text-n` mostra a influência relativa que cada atributo teve na descoberta dos clusters e o arquivo `.rlog` é o arquivo de log de execução do **reports**.

Predição (**predict**) este modo consiste na predição de possíveis clusters para um novo conjunto de dados. São utilizados os arquivos `.search` e `.results[-bin]`, criados no modo de busca (**search**), os quais possuem informações de como os exemplos foram agrupados — Figura 4.5. A saída desse modo são os exemplos agrupados nos arquivos `.class-text-n` e `.case-text-n`.

Os detalhes dos arquivos e parâmetros de execução de **AutoClass** estão descritos em <http://www-2.cs.cmu.edu/~TextLearning/eriks-code/autoclass-info/>. Entretanto, essa descrição não está bem estruturada. Assim, foi desenvolvido um relatório (Matsubara, Martins, & Monard, 2002) no qual são melhor explicados esses arquivos e parâmetros.

4.1.2.2 See5

See5⁴ é um sistema comercial para plataforma Windows™ que inclui melhorias dos algoritmos C4.5 e C4.5rules (Quinlan, 1988), os quais têm sido usados, frequentemente, para comparar seu bom desempenho com outros algoritmos de AM. O sistema **See5** foi projetado para trabalhar com conjunto de dados relativamente grandes. Como seus precursores, manipula atributos com valores discretos ou contínuos, induzindo conceitos expressos como árvores de decisão ou conjuntos de regras **if-then** não ordenadas (Baranauskas & Monard, 2000c).

Um algoritmo de aprendizado de árvore de decisão depende do critério usado para selecionar o atributo de decisão em cada ramificação da árvore. A maioria dos algoritmos de aprendizado de árvore de decisão usam alguma medida para estimar e selecionar o melhor atributo. Especificamente, **See5** utiliza a medida de entropia da teoria da informação.

Os arquivos de entrada para **See5** são dois, um contendo os dados no formato atributo-valor e o outro a descrição do tipo dos atributos, ambos com o mesmo nome, porém com extensões `.data` e `.names`, respectivamente. A saída do algoritmo pode ser

⁴<http://www.rulequest.com/see5-info.html>

uma árvore ou regras de decisão, dependendo dos parâmetros especificados pelo usuário. O `See5` e o `C4.5rules` são os principais algoritmos de [AM](#) supervisionado utilizados neste trabalho.

4.2 Ferramentas de Apoio Desenvolvidas

Para cada algoritmo utilizado, várias informações podem ser obtidas. Assim, para confrontar algumas dessas informações, na implementação do processo proposto na Figura 4.3 na página 57, foram desenvolvidas várias ferramentas de apoio, na forma de *scripts* Perl, para auxiliar tanto o processo de automação da metodologia, quanto auxiliar o especialista nesse processo de interpretação dos clusters. As principais ferramentas implementadas são descritas a seguir.

4.2.1 InClass

INCLASS é um *script* implementado para rotular os exemplos originais, não rotulados, com o cluster ao qual pertencem de acordo com `AutoClass`. A entrada para o *script* é o conjunto de dados original e o relatório `.case-text-n`, gerado por `AutoClass`.

O relatório `.case-text-n` contém uma lista na qual cada exemplo está associado ao cluster ao qual pertence com a maior probabilidade. As informações dos clusters encontrados, contidas nesse relatório, estão no formato simplificado apresentado na Tabela 4.1. Nessa tabela estão representados 128 exemplos agrupados em 22 (0...21) clusters distintos. O cabeçalho que o relatório contém foi omitido.

Tabela 4.1: Relatório `.case-text-n` gerado por `AutoClass`

Case #	Class	Prob	Case #	Class	Prob	Case #	Class	Prob
1	5	0.999	47	7	0.986	93	2	0.992
2	5	0.996	48	21	1.000	94	0	1.000
3	2	0.996	49	4	1.000	95	10	1.000
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
46	1	1.000	92	4	0.998	128	0	0.997

O relatório original é subdividido em 3 colunas e cada coluna representando a tripla

< Case #, Class, Prob >

na qual **Case #** é um número inteiro que identifica o exemplo; **Class** é o cluster ao qual esse exemplo pertence; **Prob** é a probabilidade relativa do exemplo pertencer ao cluster.

Na Tabela 4.2 é mostrado o arquivo final com o novo conjunto de dados gerado pelo INCLASS, no qual A_j refere-se ao atributo j dos exemplos e a_{ij} refere-se ao valor do atributo j do exemplo i .

Tabela 4.2: Arquivo final gerado pelo INCLASS

Exemplo	A_1	A_2	A_3	$\dots$	A_M	Cluster
1	a_{11}	a_{12}	a_{13}	$\dots$	a_{1M}	5
2	a_{21}	a_{22}	a_{23}	$\dots$	a_{2M}	5
3	a_{31}	a_{32}	a_{33}	$\dots$	a_{3M}	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
128	a_{1281}	a_{1282}	a_{1283}	$\dots$	a_{128M}	0

Na construção desse novo conjunto de dados, o INCLASS pode considerar todos os exemplos ou apenas exemplos que pertencem aos clusters encontrados com uma dada probabilidade, dependendo dos parâmetros especificados pelo usuário. O algoritmo INCLASS é apresentado no Algoritmo 1.

Algoritmo 1 INCLASS

Requisito: *Conjunto de dados*; {conjunto de *Exemplos*, i.e. conjunto de dados original}
Relatório; {conjunto de *Tuplas* $\langle$ *Exemplo*,*Cluster*,*Probabilidade* $\rangle$ criado por *AutoClass* usando *Conjunto de dados*}
MinProbabilidade; {probabilidade mínima do exemplo pertencer à classe}

- 1: **procedimento** INCLASS (*Conjunto de dados*,*Relatório*,*MinProbabilidade*)
- 2: $Final := \emptyset$
- 3: **para todo** *Tupla* **em** *Relatório* **faça**
- 4: **se** *Probabilidade* $\geq$ *MinProbabilidade* **então**
- 5: rotula *Exemplo* com *Cluster*
- 6: $Final := Final \cup \{Exemplo\}$ {inclui *Exemplo* rotulado}
- 7: **fim-se**
- 8: **fim-para**
- 9: **return** *Final*
- 10: **end** INCLASS

4.2.2 Discover $\times$ AutoClass $\times$ Discover — DAD

Caso os exemplos estejam no formato padrão do DISCOVER, o *script* DISCOVER $\times$ AutoClass $\times$ DISCOVER — DAD — é uma importante ferramenta de apoio que automatiza a conversão dos exemplos na sintaxe padrão do DISCOVER para AutoClass, faz a chamada para execução de AutoClass, realiza o rotulamento dos exemplos, e converte os exemplos rotulados novamente para a sintaxe do DISCOVER.

O DAD é um *script* orientado a objeto e utiliza duas classes para formatar os arquivos, a classe `DISC2AUTO` e a classe `OOINCLASS`. A entrada do *script* consiste de dois arquivos `.data` e `.names` na sintaxe padrão do sistema DISCOVER. A saída do DAD consiste praticamente dos mesmos arquivos de entrada, porém, com os exemplos rotulados com o atributo ‘classe’. A estrutura do *script* DAD está ilustrada na Figura 4.6.

Figura 4.6: Estrutura de execução interna do DAD

O primeiro arquivo analisado pelo DAD é o de declaração de atributos `.names`. O segundo arquivo analisado é o arquivo de exemplos `.data`. Ao ler o arquivo `.names` a classe `DISC2AUTO` é instanciada. Dependendo do tipo do atributo, real ou discreto, são instanciadas classes que contêm os métodos específicos para o tipo de atributo. O DAD, usando as instâncias dessa classe, ao ler o arquivo `.data`, executa os métodos específicos para o tipo de atributo. Esses métodos extraem as informações necessárias para criar os quatro arquivos de entrada de `AutoClass` e, posteriormente, realizar a chamada para execução de `AutoClass`. Com a saída de `AutoClass`, clusters descobertos, os exemplos são automaticamente rotulados com o cluster ao qual pertencem pela instanciação da classe `OOINCLASS`, uma versão orientado a objeto do `INCLASS`. O DAD, por meio da instanciação do `OOINCLASS`, gera novamente os arquivos `.data` e `.names`, com o atributo adicional representando a classe, no formato padrão do sistema DISCOVER.

4.2.3 AutoClass × Prolog

O *script* `Class2Prolog`, também orientado a objeto, converte o arquivo de saída `.class-text-n` de `AutoClass` em uma base de dados Prolog. Com a base em formato Prolog é possível realizar vários testes, de forma simples e rápida, auxiliando o especialista/usuário na interpretação dos clusters.

O arquivo `.class-text-1` — Figura 4.6 — possui informações sobre a probabilidade do exemplo pertencer a um ou mais clusters. Na Tabela 4.3 é apresentada uma versão simplificada de um arquivo `.class-text-1` contendo o relatório correspondente. Nesse arquivo o cluster 0 (`CLASS = 0`) contém os exemplos 1, 2, 5 e 9, no qual o exemplo 1 tem probabilidade 0.993 de pertencer ao cluster 0, probabilidade 0.005 de pertencer ao cluster 1 e probabilidade 0.002 de pertencer ao cluster 3. O exemplo 2 tem probabilidade 0.999 de pertencer ao cluster 0 e probabilidade 0.001 de pertencer ao cluster 3, e assim por diante.

Tabela 4.3: Relatório `.class-text-n` gerado por AutoClass

CLASS = 0					
1	0.993	1	0.005	3	0.002
2	0.999	3	0.001		
5	0.996	1	0.002	2	0.002
9	1.000				
⋮					

O *script* armazena todos os exemplos em uma tabela *hash*, na qual estão contidos o índice do exemplo e as probabilidades referentes à cada cluster. Com essa estrutura em memória torna-se fácil listar os exemplos em qualquer formato desejado. Assim, o `Class2Prolog` grava os exemplos usando uma estrutura de representação de uma base de dados Prolog, na forma

`ex(Exemplo,ListaProbabilidade).`

na qual `Exemplo` identifica o índice do exemplo e `ListaProbabilidade = [(C1,P1) ... (CNcl,PNcl)]` representa as probabilidades P₁ ... P_{Ncl} do `Exemplo` pertencer a cada um dos clusters (C₁ ... C_{Ncl}) — Figura 4.7.

```
ex(1,[(0,0.993),(1,0.005),(3,0.002)]).
ex(2,[(0,0.999),(3,0.001)]).
.
ex(5,[(0,0.996),(1,0.002),(2,0.002)]).
.
ex(9,[(0,1.000)]).
```

Figura 4.7: Arquivo de saída do *script* `Class2Prolog`

Utilizando essa base de dados Prolog, é possível obter informações, por meio de simples interrogações a Prolog, tais como, quais os exemplos que pertencem a mais de

um cluster, quais os exemplos que pertencem a um cluster específico, quais os clusters disjuntos, quantos exemplos há em um determinado cluster, e outros.

4.2.4 Cluster × Classe

Caso as classes de um conjunto de dados sejam conhecidas pelo usuário/especialista, é possível analisar a correspondência entre essas classes conhecidas e os clusters encontrados por **AutoClass**. Um outro *script* implementado refere-se a essa relação clusters × classes. Assim, a entrada para o *script* são os dois conjuntos de dados, um com os exemplos rotulados com a classe conhecida e outro com os exemplos rotulados com o cluster respectivo.

Para ilustrar a saída do *script*, considere um conjunto com 248 exemplos e 3600 atributos, cujas classes verdadeiras (conhecidas) são: **economia** (63 exemplos), **esporte** (59 exemplos), **informatica** (66 exemplos) e **politica** (60 exemplos). Considere que o algoritmo de agrupamento encontra apenas dois clusters **cluster_0** e **cluster_1**. Assim, um possível resultado da saída do *script* é mostrada na Figura 4.8.

```
Numero total de exemplos:  248
Numero total de atributos: 3600

Numero total de exemplos por classe:
  economia : 63 (25.40%)
  esporte  : 59 (23.79%)
  informatica : 66 (26.61%)
  politica  : 60 (24.19%)

Numero total de exemplos - cluster X classe:
  cluster_0 : 199 (80.24%)
    politica : 60 (100%)
    economia : 63 (100%)
    informatica : 17 (25.76%)
    esporte  : 59 (100%)

  cluster_1 : 49 (19.76%)
    informatica : 49 (74.24%)
```

Figura 4.8: Arquivo de saída do *script* cluster×classe

Primeiramente, no arquivo de saída do *script*, é retornado o número total de exemplos e atributos no conjunto de dados rotulados e a quantidade de exemplos em cada uma das classes verdadeiras, com a porcentagem respectiva. Após, é dada a informação correspondente a relação cluster × classe. Como ilustrado na Figura 4.8, o **cluster_0**

contém 199 exemplos e o `cluster_1` contém 49 exemplos, o que corresponde a 80,24% e 19,76% do número total de exemplos. Nesse caso, é possível observar que os dois clusters são disjuntos, *i.e.* $199 + 49 = 248$.

Pode ser observado que os exemplos contidos no `cluster_0`, pertencem as quatro classes verdadeiras `economia`, `esporte`, `informatica` e `politica`. Todos os exemplos da classe `politica`, `economia` e `esporte` foram agrupados no `cluster_0` juntamente com 17, dos 66 exemplos (25,76%), da classe `informatica`, enquanto que o restante dos exemplos da classe `informatica` estão agrupados no `cluster_1`.

No caso da porcentagem de número de exemplos por classe em cada cluster, ela é calculada com relação ao número total de exemplos na classe e não com relação ao número de exemplos no cluster.

4.2.5 Influência dos Atributos — AutoClass $\times$ See5

O relatório `.influ-o-text-n` gerado por AutoClass — Figura 4.4 na página 59 — contém informações relacionadas à influência relativa dos atributos na descoberta dos clusters. Assim, é possível confrontar os atributos que aparecem nas regras induzidas por See5 e a influência desses atributos para AutoClass, possibilitando a verificação da influência dos atributos para ambos algoritmos.

O *script* desenvolvido que verifica essa influência tem como saída um relatório com a seguinte representação

```
< #_At : InflAuto : R_See5 : Atributo >
```

na qual `#_At` é um número que identifica o atributo, dessa forma é possível verificar quantos atributos diferentes aparecem nas regras induzidas por See5; `InfluAuto` indica a influência relativa, para AutoClass, desse atributo na geração dos clusters; `R_See5` indica em quantas regras, do conjunto de regras induzidas por See5, esse atributo comparece; `Atributo` indica o nome do atributo.

A saída do *script* é ilustrada na Figura 4.9. Por exemplo, o primeiro atributo, `estad`, tem uma influência de 0.896 nos clusters gerados por AutoClass e aparece em apenas uma das regras induzidas por See5, enquanto que o atributo `represent` (`#_At` 9), aparece em 2 das regras de See5 e tem uma influência de 0.502 na geração dos clusters. Essa influência é uma medida heurística da influência relativa de cada atributo na diferenciação dos clusters de todo o conjunto de dados. E não tem uma correspondência direta com a probabilidade do exemplo pertencer a cada um dos clusters, apresentados nos relatórios

#_At	InfluAuto	R_See5	Atributo
1	0.896	1	"estad"
2	0.872	1	"construc"
3	0.925	1	"computador"
4	0.751	1	"relaca"
5	0.854	1	"usuar"
6	0.758	1	"desenvol"
7	0.911	1	"consult"
8	0.732	2	"soluc"
9	0.502	2	"represent"
10	0.770	3	"process"

Figura 4.9: Arquivo de saída do *script* `AutoClass` $\times$ `See5`: atributos presentes nas regras geradas por `See5` e a influência desses atributos para `AutoClass` na descoberta dos clusters

gerados por `AutoClass`— Tabela 4.1 na página 61 e Tabela 4.3 na página 64.

É importante salientar que os paradigmas de aprendizado usado por `AutoClass` e `See5` são diferentes. Assim, a influência dos atributos para o aprendizado realizado por esses algoritmos pode também ser diferente. A partir desse relatório é possível analisar se os atributos que apareceram nas regras do `See5`, ou seja, os atributos relevantes para a indução da hipótese, são os que mais influenciaram `AutoClass` na geração dos clusters. Essa informação pode ser importante para o usuário/especialista na tomada de decisão.

4.3 Integração ao Projeto Discover

Como já mencionado, o projeto DISCOVER utiliza um formato padrão para representar os exemplos e as hipóteses induzidas por algoritmos de AM simbólico. Dessa forma, é necessário que todos os módulos a serem integrados ao DISCOVER obedeçam a uma padronização. No caso de algoritmos de agrupamento, é simples utilizar o formato padrão para representar exemplos do DISCOVER, pois há uma correspondência quase direta entre esse formato e o formato dos exemplos utilizado pelos diversos algoritmos de agrupamento. O problema concentra-se na definição de um formato padrão para a saída dos diversos algoritmos de agrupamento, pois, como mencionado na Seção 2.3.1 na página 22, a saída desses algoritmos é fortemente dependente da técnica de agrupamento utilizada.

É esperado, portanto, que as diferentes técnicas de agrupamento (otimização, hierárquicas, *clumping*, probabilísticas), apresentem formatos de saída bastante diversificados, sendo assim, é praticamente impossível chegar a um formato padrão para qualquer

saída de algoritmos de agrupamento. Ou seja, diferentemente dos algoritmos de aprendizado simbólico supervisionado, nem toda saída dos algoritmos de agrupamento poderá ser “traduzida” para um mesmo formato padrão, como foi possível para o caso de regras **if-then** de algoritmos de aprendizado simbólico supervisionado.

Entretanto, consideramos que será possível, no futuro, descobrir uma sintaxe padrão para representar a saída dos algoritmos que utilizem as diferentes técnicas de agrupamento. Na Figura 4.10 está ilustrada essa idéia.

Figura 4.10: Biblioteca de conversão para algoritmos de agrupamento

A metodologia proposta neste capítulo, bem como as implementações realizadas, estão sendo integradas ao ambiente DISCOVER. Por exemplo, o *script* DAD realiza a conversão dos dados a partir do formato padrão do DISCOVER para o formato de **AutoClass** e também transforma a saída de **AutoClass** para o formato do DISCOVER. No entanto, caso diferentes técnicas sejam futuramente utilizadas, será necessário projetar e desenvolver novas bibliotecas de conversão no DISCOVER que possam, pelo menos, atingir um formato padrão para cada uma das diferentes técnicas de agrupamento, como ilustrado na Figura 4.10.

4.4 Considerações Finais

O problema de agrupamento tem sido bastante estudado em vários contextos e por diferentes pesquisadores, refletindo sua importância e utilidade como um dos passos na análise exploratória de dados. Entretanto, agrupamento é um problema combinatório difícil e algumas divergências em hipóteses e contextos em comunidades científicas diferentes, fez com que o surgimento de conceitos e metodologias gerais que possam ser úteis, tivesse um desenvolvimento um pouco lento ([Jain, Murty, & Flynn, 1999](#)).

Dentro do projeto DISCOVER está, também, contemplado o uso de [AM](#) não supervisionado, relacionado ao processo que auxilia a interpretação de clusters, proposta neste trabalho. Assim, o trabalho descrito neste capítulo consiste na apresentação de uma metodologia para descoberta e interpretação de cluster. Essa metodologia utiliza uma combinação de algoritmos de agrupamento e aprendizado indutivo, em conjunto de dados não rotulados. O objetivo é auxiliar o especialista na interpretação dos clusters encontrados bem como agrupar clusters diferentes mas que se referem a um mesmo conceito. Essa metodologia foi utilizada neste trabalho tanto em conjunto de dados estruturados, Capítulo 6, quanto em dados não estruturados relacionados ao processo de mineração de textos, Capítulo 7. No entanto, para que possa ser aplicado em [MT](#), os textos devem ser pré-processados, em um formato estruturado, para serem submetidos a algoritmos de aprendizado. No próximo capítulo é apresentada uma ferramenta computacional, denominada PRETEXT, que realiza esse pré-processamento de dados textuais.

PRETEXT: uma Ferramenta para Pré-processamento de Dados Textuais

A representação de documentos textuais em um formato estruturado para o processo de mineração de textos tem uma influência fundamental em quão bem um algoritmo de aprendizado poderá generalizar. A abordagem *bag-of-words* é uma das representações estruturadas mais simples, mais utilizada e que tem obtido um bom desempenho no processo de mineração de textos. No entanto, essa abordagem é caracterizada pela alta dimensionalidade e por valores esparsos na representação dos textos, visto que cada palavra é um possível atributo nessa representação. São necessárias, portanto, ferramentas computacionais que possam realizar de forma automática a transformação dos documentos em uma representação estruturada e que, ao mesmo tempo, auxiliem na redução da dimensionalidade dessa representação. Neste capítulo é descrita em detalhes a ferramenta PRETEXT que tem essas características, bem como diversas outras funcionalidades que a distinguem de outras ferramentas existentes.

5.1 A Ferramenta PRETEXT

PRETEXT ([Matsubara, Martins, & Monard, 2003](#)) é uma ferramenta computacional implementada em Perl ([Wall, Christiansen, & Schwartz, 1996](#)), desenvolvida com o objetivo de realizar o pré-processamento de um conjunto de documentos utilizando a abordagem *bag-of-words*. Uma de suas principais funcionalidades é transformar palavras presentes em documentos escritos em inglês, português ou espanhol, em *stems*. O algoritmo de *stemming* implementado na ferramenta é baseado no algoritmo do [Porter](#)

(1980) para a língua inglesa e adaptada para o português e o espanhol — Apêndice A na página 155.

Algumas ferramentas que realizam pré-processamento de textos são descritas nos sistemas Rainbow (McCallum, 1996) e Ngram (Banerjee & Pedersen, 2003). Entretanto, foi decidido implementar a ferramenta PRETEXT pois essas ferramentas não possuem algumas funcionalidades que consideramos necessárias. Essas funcionalidades estão relacionadas à quantidade de informações geradas, bem como a facilidade que os arquivos gerados possam ser submetidos a diversos algoritmos de aprendizado que utilizam o formato atributo-valor para os dados de entrada. Ainda, um outro motivo é a facilidade de integração do PRETEXT ao projeto DISCOVER, já que a sua implementação contempla os requisitos necessários para integrá-lo facilmente nesse ambiente, o que não acontece com as ferramentas disponíveis.

PRETEXT utiliza a abordagem *bag-of-words* para transformar os documentos no formato de tabela atributo-valor do DISCOVER e foi implementado em dois módulos: **stem.pl** e **report.pl**, como ilustrado na Figura 5.1.

Figura 5.1: A ferramenta PRETEXT

O módulo **stem.pl** da ferramenta é responsável pela transformação das palavras em *stems*. Os documentos a serem processados devem estar em um diretório, como ilustrado na Figura 5.1, ou em uma hierarquia de subdiretórios. Para a execução do módulo **stem.pl** é necessário um arquivo contendo algumas especificações dos parâmetros de execução (**parameters**), bem como um diretório contendo uma ou mais listas de *stopwords* (**StopList**), que são palavras pouco significativas como artigos, preposições e conjunções que pouco caracterizam os documentos. A saída desse módulo consiste em diversos arquivos intermediários, denominados de **stembase**, que contêm os *stems* correspondentes às palavras

contidas nos documentos e informações úteis relacionadas a cada um dos *stems*.

É importante ressaltar que, para um mesmo conjunto de documentos e uma mesma lista de *stopwords*, o módulo **stem.pl** gera a mesma **stembase**. Assim, uma vez transformados os documentos em um conjunto de *stems*, esse módulo não precisa ser executado novamente, e o usuário pode gerar diferentes tabelas atributo-valor executando apenas o módulo **report.pl** com diferentes parâmetros.

O módulo **report.pl**, a partir de alguns arquivos gerados pelo módulo **stem.pl** e dos parâmetros especificados pelo usuário, retorna informações para geração de gráficos e criação da tabela atributo-valor. A tabela é gerada no formato padrão do DISCOVER representados pelo arquivos de dados (**.data**) e pelo arquivo de atributos (**.names**) — Figura 5.12 na página 91. Para calcular os valores dos atributos na tabela, a ferramenta utiliza qualquer uma das medidas descritas na Seção 3.2.1 na página 36, bem como as medidas propostas na Seção 5.2 na próxima página, de acordo com o parâmetro especificado pelo usuário. Além disso, a ferramenta apresenta facilidades para reduzir a dimensionalidade do conjunto de atributos usando a lei de Zipf e os cortes de Luhn — Seção 3.2.2.3 na página 42.

Uma das características da ferramenta é a construção de *stems* usando mais de um *gram*. No PRETEXT, 1-*gram* se refere a um *stem* simples, enquanto 2 e 3-*gram* referem-se a 2 ou 3 *stems*, cujas palavras ocorrem sequencialmente no documento. A ferramenta permite a concatenação de até 3 *stems*, ou seja, 3-*gram*. Porém, os *gram* são formados a partir dos *stems* gerados e, portanto, termos que são *stopwords*, como por exemplo o termo ‘de’, não comparecem na concatenação de *stems*. É possível trabalhar com a combinação de *gram* a fim de obter uma melhor representação dos documentos. A utilização de mais de um *gram* permite que palavras que aparecem sequencialmente no documento como “inteligência artificial”, “aprendizado de máquina” e “mineração de textos”, que são mais representativas conceitualmente quando utilizadas juntas, possam ser utilizadas no PRETEXT.

Uma outra característica da ferramenta é a facilidade de usar indução construtiva — criação de novos atributos na tabela atributo-valor que são generalizações de dois ou mais atributos. A indução construtiva é realizada utilizando um arquivo contendo um conjunto de taxonomias (**Taxonomy**) definidas pelo usuário, o qual, quando disponível, é interpretado pelo módulo **report.pl**.

A ferramenta PRETEXT está implementada usando idéias aceitas e difundidas na comunidade científica. No entanto, PRETEXT possui diversas facilidades implementadas e, quando executada usando um conjunto de documentos, retorna uma grande quantidade

de informações relacionadas a esses documentos. O grande diferencial da ferramenta consiste nessa diversidade de informações geradas, uso de taxonomias e o uso de várias medidas que podem ser utilizadas para auxiliar no processo de mineração de textos.

5.2 As Medidas Implementadas para Atribuição de Valores aos Atributos

As medidas de atribuição de valores aos atributos (*stems*) implementadas no PRETEXT são, além das citadas na literatura — binária, *tf*, *tfidf*, *tfidf_n*, Seção 3.2.1 na página 36 —, algumas outras medidas propostas na implementação da ferramenta.

A representação binária — Equação 3.1 na página 38 — usa os valores 0 e 1 para indicar que o atributo está ou não está presente no documento. A medida *tf* — Equação 3.2 na página 38 — usa a frequência que o termo aparece. A medida *tfidf* — Equação 3.4 na página 38 — tenta discriminar melhor os documentos utilizando, juntamente com a medida *tf*, um fator de ponderação¹ que é a medida *idf* — Equação 3.3 na página 38. O fator de ponderação pode ser utilizado quando os termos que aparecem na maioria dos documentos possam ter uma “força” de representação menor. Esse fator varia entre 0 e $\log N$. Assim, quando o termo aparece em todos os documentos o fator de ponderação é igual a 0, quando aparece em apenas 1 documento ele é $\log N$.

Para a implementação do PRETEXT, foi proposta uma nova medida, semelhante à *tfidf*, denominada *tflinear*. Essa medida, definida pela Equação 5.1, utiliza um fator de ponderação linear, definido pela Equação 5.2², diferente do fator logarítmico usado na medida *tfidf*. Desse modo, o fator de ponderação *linear* varia entre 0 e 1.

$$tflinear(t_j, d_i) = tf(t_j, d_i) \times linear(t_j) \quad (5.1)$$

$$linear(t_j) = 1 - \frac{x}{N} \quad (5.2)$$

Por exemplo, suponha que em um conjunto de 100 documentos ($N = 100$) o termo **estudar**, que corresponde ao *stem* **estud**, apareça em 90 documentos ($x = 90$), ou seja, o termo aparece em quase todos os documentos. O fator de ponderação linear para esse termo é

¹O fator de ponderação corresponde ao componente do documento, 3.2.1 na página 36.

²O x representa o número de documentos em que o termo aparece.

$$\text{linear}(\text{estudar}) = 1 - \frac{90}{100} = 0,1$$

que é um valor muito próximo de zero. Os valores de a_{ij} , para $t_j = \text{estudar}$, serão sempre multiplicados por 0,1, ou seja, todos os valores que compõem a coluna do termo **estudar** na tabela atributo-valor — Tabela 3.1 na página 37 — são multiplicados por 0,1. Assim, o termo **estudar** fica com menos “força” do que os termos que aparecem em apenas um documento, para os quais tem-se

$$\text{linear}(\text{estudar}) = 1 - \frac{1}{100} = 0,99$$

O fator linear para um conjunto com 1000 documentos e um termo com frequência 50 aparecendo em apenas 1 documento, é dado por $(1 - \frac{1}{1000}) = 0,999$ que é muito próximo a 1. O fator logarítmico enfatiza, principalmente, termos que aparecem em poucos documentos em um grande conjunto de documentos. O valor do fator de ponderação logarítmico, para esse mesmo conjunto de documentos é $\log \frac{1000}{1} = 3$. Assim, o valor de a_{ij} usando *tfidf* para esse termo corresponde ao valor da sua frequência triplicada (50 vai para 150). Portanto, para esses casos, o fator linear não enfatiza o termo tanto quanto o fator logarítmico.

A Figura 5.2 ilustra o comportamento desses dois fatores de ponderação para um conjunto com 100 documentos. O fator *idf* para termos que aparecem em apenas um documento é $\log \frac{100}{1} = 2$. Como pode ser observado, conforme o termo está presente em uma quantidade maior de documentos, sua “força” diminui. Assim, o fator *idf* para termos que aparecem em todos os documentos é $\log \frac{100}{100} = 0$.

Dessa forma, para coleções de documentos com temas semelhantes, é possível que alguns termos apareçam em todos os documentos do conjunto de textos. Nesse caso, tanto os fatores de ponderação *idf* quanto *linear* são nulos. Uma possível solução é não permitir que o fator de ponderação seja zero, pois um termo pode ser relevante na discriminação do conteúdo mesmo que apareça em todos os documentos. Assim, no PRETEXT é proposto um outro fator denominado *smooth*.

O *smooth* é ativado quando o fator de ponderação é igual a zero, ou seja, quando um termo aparece em todos os documentos. Quando ativado, ele aumenta temporariamente em 10% a variável N que contém o número de documentos do conjunto de textos, não permitindo que o fator seja zero.

Suponha que o valor de *idf* para um termo que apareça em todos os documentos é $\log \frac{100}{100} = 0$. Com o fator de ponderação igual a zero, o *smooth* é ativado. O valor de N é aumentado temporariamente para 110 obtendo assim $\log \frac{110}{100} = 0,04$. Desse modo, os valores da coluna correspondente a esse termo serão multiplicados por 0,04 ao invés de 0.

Figura 5.2: Fator de ponderação $idf \times linear$

Um outro aspecto que também deve ser levado em conta é o tamanho dos documentos no conjunto de textos. Frequentemente, o tamanho desses documentos é muito diferente e essa diferença de tamanho deveria estar melhor refletida nas medidas utilizadas. Por exemplo, considere dois documentos que pertencem à mesma categoria com tamanhos de 1 Kbyte e 100 Kbytes respectivamente. Nesse caso, existirá uma grande diferença na frequência dos termos em ambos os documentos. Uma possível solução para esse problema é normalizar os valores da tabela atributo-valor — Tabela 3.1 na página 37. Essa normalização é frequentemente realizada usando a normalização linear, definida pela Equação 5.3, ou a normalização quadrática³, definida pela Equação 5.4.

$$NormLinear(t_j, d_i) = \frac{a_{ij}}{MAX_{i=1..N}(a_{ij})} \quad (5.3)$$

$$NormQuadratic(t_j, d_i) = \frac{a_{ij}}{\sqrt{\sum_{i=1}^N (a_{ij})^2}} \quad (5.4)$$

Para ilustrar os valores dessas medidas em um conjunto de documentos, foram criados 7 (sete) documentos — Tabela 5.1 — para mostrar como as medidas se comportam. Assim, o primeiro documento (`texto1.txt`) é composto somente pelo termo `cas` com frequência 1, o segundo documento (`texto2.txt`) é composto por dois termos, `cas` e `filh`, também ambos com frequência 1 e assim por diante, até que os três últimos documentos

³A normalização quadrática é utilizada na medida *tfidf*, Equação 3.5 na página 39.

são compostos por 5 termos com frequências variadas.

Tabela 5.1: Representação de documentos utilizando *bag-of-words*

documento	amig	ciudad	trabalh	filh	cas
texto1.txt	0	0	0	0	1
texto2.txt	0	0	0	1	1
texto3.txt	0	0	1	1	1
texto4.txt	0	1	1	1	1
texto5.txt	5	5	5	5	5
texto6.txt	5	8	13	28	28
texto7.txt	5	6	10	8	8

O objetivo dos quatro primeiros documentos é apenas mostrar a variação das medidas em função da quantidade de documentos nos quais um mesmo termo aparece. Dessa forma, verifica-se que o termo **amig** aparece em 3 documentos, o termo **ciudad** aparece em 4 documentos, o termo **trabalh** aparece em 5 documentos, e assim por diante até que o último termo **cas** aparece em todos os documentos. O fator de ponderação é calculado de acordo com o número de documentos em que o termo aparece. Portanto, os quatro primeiros documentos, **texto1.txt**, **texto2.txt**, **texto3.txt** e o **texto4.txt** que contêm um, dois, três e quatro termos com valores não nulos, respectivamente, foram criados estrategicamente para mostrar as variações das medidas. Em outras palavras, esses valores foram criados para melhor ilustrar a “força” que o fator de ponderação exerce sobre as medidas.

Para verificar o comportamento das medidas, observe os diferentes valores dos termos para os documentos **texto5.txt**, **texto6.txt** e **texto7.txt**. Todos os cinco termos que aparecem no conjunto de textos estão presentes no **texto5.txt** com frequência 5. Desse modo, o fator de ponderação altera a “força” de cada termo de acordo com o número de documentos em que o termo aparece, ou seja, quanto maior o número de documentos nos quais o termo aparece, menor é a sua força de representação, ilustrado pelos valores decrescentes na Figura 5.3. A medida *tf*, que não usa fator de ponderação, considera todos os termos com frequência constante 5.

Os valores das medidas *tfidf* e a *tfidf* usando *smooth* (*tfidfS*) são os mesmos para os primeiros quatro termos, sobrepostas na figura, diferindo apenas para o termo **cas**, que possui o fator de ponderação dado por $idf(cas) = \log \frac{7}{7} = 0$. Quando o valor do fator de ponderação é zero, como nesse caso, o *smooth* altera temporariamente a variável que contém o número de documentos do conjunto, não permitindo que o fator de ponderação seja zero, como no caso dos valores do termo **cas** para a medida *tfidfS*. Como pode ser observado, para as medidas *tfidf* e *tflinear* que não utilizam o *smooth*, o termo **cas** que aparece em todos os documentos tem seu valor zerado.

Figura 5.3: Documento `texto5.txt` representado por diferentes medidas

Os valores dos termos para o documento `texto6.txt` foram criados para ilustrar o comportamento da medida *tfidf* quando atinge valores praticamente constantes para os termos, como ilustrado na Figura 5.4. Observe que nessa medida os valores da frequência (*tf*) são fortemente influenciados pelo fator de ponderação e que o último termo, `cas`, tem valores diferentes para todas as medidas.

Figura 5.4: Documento `texto6.txt` representado por diferentes medidas

Ainda, como pode ser observado, os valores dos termos `amig` e `cas`, cujas frequências são bem diferentes (5 e 28 respectivamente), quando os fatores de ponderação *idf* e *linear*

são utilizados nas medidas *tfidf* e *tflinear*, os valores desses dois termos ficam semelhantes, pois, a frequência do termo **amig** é 5, porém esse termo aparece em apenas três dos sete documentos do conjunto, enquanto que a frequência do termo **cas** é 28, porém esse termo aparece em todos documentos do conjunto. Finalmente, a Figura 5.5 ilustra essas medidas para o documento `texto7.txt`.

Figura 5.5: Documento `texto7.txt` representado por diferentes medidas

O fator de ponderação é fortemente dependente do número de documentos em que o termo aparece, podendo mudar significativamente os valores dos termos na representação dos documentos. Em geral, o fator de ponderação diminui a frequência dos termos. De fato, usando o fator de ponderação *linear*, o valor da frequência será sempre menor ou igual a própria frequência. Para o fator de ponderação *idf*, o valor será menor caso o termo apareça em muitos documentos. O *smooth*, quando utilizado, atua apenas nos termos que aparecem em todos os documentos do conjunto para que os valores desses termos não se tornem nulos.

5.3 A Implementação de PRETEXT

A implementação foi realizada utilizando o paradigma de orientação a objetos em Perl. Assim, a ferramenta foi implementada em dois módulos: **stem.pl** e **report.pl** — Figura 5.1 na página 72. Basicamente, esses módulos representam o módulo de geração de *stems* e o módulo de geração dos arquivos do DISCOVER, respectivamente. Nas próximas seções serão apresentados mais detalhadamente cada um dos módulos, bem como o arquivo

de parâmetros utilizado pelos dois módulos e alguns arquivos auxiliares.

5.3.1 O Arquivo de Parâmetros

A principal interação entre o usuário e a ferramenta é realizada pela definição de parâmetros no arquivo de configurações — Figura 5.1 na página 72 — denominado `parameters.cfg`. Nesse arquivo são definidas todas as informações necessárias à execução dos dois módulos de PRETEXT. O arquivo de parâmetros contém a definição dos parâmetros divididos em quatro grupos, denominados respectivamente, `%global%`, `%one%`, `%two%` e `%three%`, como mostrado a seguir.

```
%global%  
%one%  
%two%  
%three%
```

Caso o arquivo de parâmetros consista somente dessas quatro linhas, então PRETEXT é executado usando os valores *default*, os quais são mostrados na Figura 5.6. No primeiro grupo desse arquivo — `%global%` — são especificados parâmetros globais, tais como a língua na qual os textos estão escritos e os diretórios que contêm os dados, entre outros. Todos os parâmetros que começam com `dir` são considerados definições de diretórios.

Nos grupos `%one%`, `%two%` e `%three%` são definidos os parâmetros para 1, 2 e/ou 3-gram. É possível habilitar um, dois ou três gram, ou qualquer combinação dessas. Desse modo, podem-se construir tabelas com 1 e 2 gram, 1 e 3 gram, 2 e 3 gram, ou com 1, 2 e 3 gram. Todos os parâmetros de execução do PRETEXT são descritos em detalhes a seguir.

- `language`: especifica a língua que os textos estão escritos. Valores possíveis: `port`, `espn` ou `ingl`. Valor *default* `port`. Consultado por `stem.pl`.
- `logFile`: especifica o nome do arquivo com o log de execuções; Valor *default* `pretex.log`. Consultado por `stem.pl` e `report.pl`.
- `dirStemBase`: especifica o nome do diretório que deve conter os arquivos intermediários (`stembase`). Valor *default* `stembase`. Consultado por `stem.pl` e `report.pl`.
- `dirTextBase`: especifica o diretório que contém o conjunto de textos. Valor *default* `textbase`. Consultado por `stem.pl` e `report.pl`.


```
1  # Esta é uma linha de comentário
2  %global%
3  language      = port
4  logFile       = pretext.log
5  dirStemBase   = stembase
6  dirTextBase   = textbase
7  dirStopList   = stoplist
8  dirDiscover   = discover
9  dirGraphics   = graphics
10 dirPredict    = predict
11 %one%
12 measure       = freq
13 normalize     = quadratic
14 smooth        = disabled
15 min           = 0
16 max           = 0
17 std_dev       = 0
18 %two%
19 measure       = freq
20 normalize     = quadratic
21 smooth        = disabled
22 min           = 0
23 max           = 0
24 std_dev       = 0
25 %three%
26 measure       = freq
27 normalize     = quadratic
28 smooth        = disabled
29 min           = 0
30 max           = 0
31 std_dev       = 0
```

Figura 5.6: Exemplo de arquivo de parâmetros

- **dirStopList**: especifica o diretório que contém a(s) lista(s) de *stopwords*. Valor *default* **stoplist**. Consultado por **stem.pl** e **report.pl**.
- **dirDiscover**: especifica o diretório no qual serão armazenados os arquivos **.data** e **.names**. Valor *default* **discover**. Consultado por **stem.pl** e **report.pl**.
- **dirGraphics**: especifica o diretório no qual serão armazenados os arquivos para criação de gráficos. Valor *default* **graphics**. Consultado por **stem.pl** e **report.pl**.
- **gram**: se **disabled** o *gram* é desabilitado para o módulo em execução. Por exemplo, pode-se habilitar os três *gram* para executar o **stem.pl** e habilitar apenas um dos *gram* para o **report.pl**. Valores possíveis: **enabled** e **disabled** Valor *default* **enabled**. Consultado por **stem.pl** e **report.pl**.

- **measure**: define a medida que é utilizada para construir a tabela atributo-valor. Valores possíveis `bool`, `freq`, `tfidf` e `tflinear`. Valor *default* `freq`. Consultado por **report.pl**.
- **normalize**: define o método de normalização a ser aplicado aos valores dos atributos da tabela atributo-valor. Valores possíveis `linear`, `quadratic` e `disabled`. Valor *default* `quadratic`. Consultado por **report.pl**.
- **smooth**: se `disabled` o *smooth* é desabilitado. Caso contrário *smooth* é habilitado. Valores possíveis: `enabled` e `disabled`. Valor *default* `disabled`. Consultado por **report.pl**.
- **min**: se definido, são carregados apenas os *stems* que tiverem frequência maior ou igual ao valor definido. Valor *default* 0 (desabilitado). Consultado por **report.pl**.
- **max**: se definido, são carregados apenas os *stems* que tiverem frequência menor ou igual ao valor definido. Valor *default* 0 (desabilitado). Consultado por **report.pl**.
- **std_dev**: se definido, é calculado o *rank* de *stems*. Com o desvio padrão desse *rank* é definido um intervalo $(\bar{x} - ks; \bar{x} + ks)$, no qual $\bar{x}$ é a média do *rank*, s é o desvio padrão desse *rank* e k é o valor definido pelo usuário. Os *stems* que estão fora desse intervalo são descartados. Valor *default* 0 (desabilitado). Consultado por **report.pl**.

Os três últimos parâmetros são usados na redução da dimensão da tabela atributo-valor, representando os pontos de corte de Luhn. Os parâmetros **max** e **min** definem manualmente os pontos de corte superior e inferior de Luhn, respectivamente — Figura 3.2 na página 43. Todos os *stems* cujas frequências, considerando todo o conjunto de documentos, encontram-se fora do intervalo definido por esses parâmetros e são ignorados na construção da tabela atributo-valor. Como mencionado, os cortes são efetuados somente quando definidos no arquivo de parâmetros, no qual o usuário pode realizar qualquer combinação com os cortes **min** e **max**. Por exemplo:

- para **min** = 3, todos os *stems* com frequência 1 e 2 são ignorados e os *stems* com frequência maiores ou igual a 3 são mantidos na tabela atributo-valor;
- para **max** = 10, todos os *stems* com frequência maior que 10 são ignorados e os *stems* com frequência igual ou menor a 10 são mantidos;
- para **min** = 3 e **max** = 10, somente os *stems* com frequência maior ou igual a 3 e menores ou igual 10 são mantidos.

O parâmetro `std_dev` faz cortes dos *stems* de acordo com o desvio padrão sobre o *rank* de *stems*. O *rank* é construído da seguinte maneira: suponha uma competição na qual é possível ter empate entre as colocações, ou seja, vários primeiro, segundo e terceiro colocados. O importante no *rank* são quantas colocações existem, neste caso do primeiro até o terceiro.

Para um melhor entendimento do cálculo do desvio padrão utilizado por PRETEXT, imagine uma corrida de *stems*, como os ilustrados na Figura 5.7 (a), no qual a velocidade é determinada pela frequência de cada *stem* e todos largam na mesma posição. Assim o primeiro colocado é o *stem* que tiver a maior frequência, o segundo colocado o *stem* que tiver a segunda maior frequência, e assim por diante, podendo haver sempre empate em todas as colocações. O *rank* de *stems* é exatamente isso, o *stem* de maior frequência fica com a primeira colocação, o de segunda maior frequência fica na segunda colocação até ser construído o *ranking*.

<i>stem</i>	freq
amig	5
linux	5
gnu	5
ciad	7
trabalh	12
filh	13
idad	13
cas	17

(a)

<i>rank</i>	freq
1	17
2	13
3	12
4	7
5	5

(b)

Figura 5.7: A frequência dos *stems* (a) gera o *rank* de *stems* (b)

Com o *rank* construído, é calculado o desvio padrão e a média dos valores de 1 a 5, obtendo-se o valor 1.58 e 3, respectivamente⁴. Supondo que seja definido no arquivo de parâmetros `std_dev = 0.8`, o intervalo do *rank* é $[3 - 1.58 \times 0.8, 3 + 1.58 \times 0.8] = [1.73, 4.26]$. Esse intervalo é sobre o *rank* e, portanto, ele é convertido para $[7, 13]$ que é um intervalo sobre a frequência. Assim, os *stems* com frequências menor que 7 e maior que 13 são ignorados.

5.3.2 O Módulo `stem.pl`

O módulo `stem.pl` é responsável por transformar as palavras dos documentos em *stems* e armazenar essas informações no diretório `stembase`. Para a execução desse módulo são necessários alguns arquivos e diretórios que são apresentados a seguir.

⁴No PRETEXT é utilizado o ponto na separação das casas decimais.

5.3.2.1 Arquivos de Entrada

Os arquivos de entrada para o módulo **stem.pl** consistem de:

- arquivo de parâmetros (**parameters.cfg**): o módulo **stem.pl** consulta principalmente os parâmetros **language**, **dirStemBase**, **dirTextBase**, **dirStopList** e os *gram* habilitados (**gram = enabled**).
- diretório de *stopwords* (**stoplist**): a **stoplist** é um diretório contendo um ou mais arquivos com as *stopwords*. Cada um desses arquivos são denominados *stopfiles*. Essas *stopfiles* contêm apenas palavras como artigos, preposições, conjunções, que são pouco significativas para caracterizar o documento. O PRETEXT aloca em memória apenas as *stopfiles* com a extensão **.enabled**. Desse modo é possível habilitar e desabilitar várias *stopfiles* apenas alterando a sua extensão. Nas *stopfiles*, cada *stopword* deve ser separada por quebra de linha como ilustrado a seguir:

```
de
porem
afinal
ou
cujo
```

Durante a execução, todas as *stopfiles* ativas, ou seja com extensão **.enabled**, são concatenadas formando uma única lista. Assim, o usuário pode criar listas específicas para cada domínio, como por exemplo, uma lista específica para economia, outra para informática e assim por diante. Desse modo, quando a base de textos contiver documentos nesses domínios, o usuário pode ativar essas listas renomeando esses arquivos com a extensão **.enabled**.

- diretório de base de textos (**textbase**): a base de textos, ou **textbase**, pode ser construída de duas formas, que determinam se os dados devem ser rotulados ou não:
 1. dados não rotulados: criar o diretório **textbase** contendo os arquivos textos do conjunto sem nenhuma divisão de subdiretórios;
 2. dados rotulados: criar o diretório **textbase** com as classes separadas por subdiretórios dentro da **textbase**.

Por exemplo, suponha que sejam fornecidos os arquivos **debian.txt**, **oracle.txt**, **economiafolha.txt** e **economiaestadao.txt**, sendo que os dois primeiros são da classe **tecnologia** e os dois últimos da classe **economia**. Para que ao final de todo o processo seja construída a tabela com os dados rotulados, deve-se construir o diretório **textbase** com a seguinte estrutura.

```
textbase/  
    tecnologia/  
        debian.txt  
        oracle.txt  
    economia/  
        economiafolha.txt  
        economiaestadao.txt
```

Desse modo, com o arquivos separados por subdiretórios, no final do processo, o PRETEXT rotula automaticamente os documentos.

5.3.2.2 Arquivos de Saída

A saída do módulo **stem.pl** consiste de 8 (oito) arquivos intermediários que compõem a **stembase**, alguns com informações necessárias ao módulo **report.pl** e outros com informações úteis para o usuário. Para cada *gram* definido pelo usuário são gerados dois arquivos: um com extensão **.all** e outro com extensão **.txt**. O arquivo com extensão **.all** contém todos os *stems* do conjunto, com as frequências respectivas, e o arquivo com extensão **.txt** contém a frequência dos *stems* para cada um dos documentos do conjunto. Assim, são gerados até seis arquivos, dois para cada *gram* e outros dois arquivos restantes contêm as palavras que formaram cada um dos *stems*.

A **stembase** fornece informações no nível da palavra, o que permite saber em qual *stem* cada palavra foi transformada e em quais documentos se encontram cada um desses *stems*. Nos arquivos com extensão **.all** é possível observar, entre outras coisas, quais são os *stems* que podem ser *stopwords* em potencial, que não foram eliminados por não estarem presentes nas *stopfiles*, observando a frequência do *stem* em cada documento e em quantos documentos o *stem* aparece. Além das *stopwords*, é possível verificar quais são os pontos de cortes **max** e **min** mais razoáveis verificando a frequência das palavras mais relevantes. Os oito arquivos gerados pelo módulo **stem.pl** são:

- **oneGram.all**: este arquivo contém todos os *stems* para 1-gram de todos os documentos do conjunto e em quantos arquivos o *stem* aparece, como ilustrado na Figura 5.8 na próxima página. Por exemplo, na linha 3 dessa figura, 12: (3/5):trabalh, informa que 12 é a frequência do *stem* trabalh e (3/5) significa que esse *stem* apareceu em 3 dos 5 arquivos do conjunto de textos.
- **oneGram.txt**: este arquivo contém os *stems* separados por documentos, como ilustrado na Figura 5.9 na página 87. Nessa figura, texto1.txt (linha 1) é composto apenas

pelo *stem cas* com frequência 6, `texto2.txt` (linha 3) é composto por dois *stems*, dos quais o *stem cas* aparece com frequência 1 e o *stem filh* com frequência 3.

- `twoGram.all`: semelhante a `oneGram.all`, porém com 2-gram.
- `twoGram.txt`: semelhante a `oneGram.txt`, porém com 2-gram.
- `threeGram.all`: semelhante a `oneGram.all`, porém com 3-gram.
- `threeGram.txt`: semelhante a `oneGram.txt`, porém com 3-gram.
- `stemWdTF.all`: este arquivo contém os *stems* e as palavras que os formaram ordenados pela frequência, como mostrado na Figura 5.10 na próxima página. Nessa figura as palavras *filha filhas filho e filhos* foram transformadas no *stem filh*. Os valores na frente dos *stems* mostram a frequência do *stem* no conjunto de documentos e em quantos documentos o *stem* apareceu. Por exemplo, na linha 16 é informado que o *stem filh* aparece 13 vezes em 4 dos 5 documentos do conjunto. Os valores na frente das palavras mostram a frequência das palavras no conjunto de documentos. Na linha 17 a palavra *filha* aparece 4 vezes no conjunto de documentos.
- `stemWdST.all`: semelhante ao `stemWdTF.all`, mas ordenado alfabeticamente.

```

1  5:(1/5):amig
2  7:(2/5):ciad
3 12:(3/5):trabalh
4 13:(4/5):filh
5 17:(5/5):cas

```

Figura 5.8: Exemplo de arquivo `oneGram.all`

5.3.2.3 Execução do Módulo `stem.pl`

Com os parâmetros, arquivos e diretórios definidos corretamente, a execução do módulo `stem.pl`, para transformar os documentos em uma base de *stems*, é realizada digitando na linha de comando `stem.pl`.

Um exemplo de execução é ilustrado na Figura 5.11 na página 88. Nesse exemplo, a partir da linha 9 até a 20 são mostrados todos os parâmetros lidos pelo `stem.pl` definidos no arquivo `parameters.cfg`. Como pode-se notar, apenas o parâmetro para um *gram* está habilitado. Portanto, são criados *stems* formados por apenas uma palavra. Nas linhas 24 e

```
1  textbase/texto1.txt
2      cas:6
3  textbase/texto2.txt
4      cas:1
5      filh:3
6  textbase/texto3.txt
7      cas:1
8      filh:1
9      trabalh:1
10 textbase/texto4.txt
11     cas:1
12     cidad:1
13     filh:1
14     trabalh:1
15 textbase/texto5.txt
16     amig:5
17     cas:8
18     cidad:6
19     filh:8
20     trabalh:10
```

Figura 5.9: Exemplo de arquivo oneGram.txt

```
1  amig : 5 (1/5)
2      amiga : 2
3      amigas : 1
4      amigo : 1
5      amigos : 1
6  cidad : 7 (2/5)
7      cidade : 4
8      cidades : 2
9      cidadoes : 1
10 trabalh : 12 (3/5)
11     trabalham : 1
12     trabalhando : 1
13     trabalhar : 2
14     trabalho : 6
15     trabalhos : 2
16 filh : 13 (4/5)
17     filha : 4
18     filhas : 3
19     filho : 3
20     filhos : 3
21 cas : 17 (5/5)
22     casada : 1
23     casado : 2
24     casara : 2
25     casaram : 5
26     casas : 2
27     casos : 2
28     casou : 3
```

Figura 5.10: Exemplo de arquivo stemWdTF.all

25 são informados que foram lidos 634 *stopwords* divididas em dois arquivos. Todas essas informações são mostradas durante a execução do módulo **stem.pl** e são armazenadas no arquivo de log (**pretex.log**).

É importante salientar que na execução desse módulo, é recomendável que sejam ativados todos os *gram*, já que computacionalmente o tempo de execução é praticamente o mesmo.

```
1  Execution started at Mon Jul  7 20:58:56 2003
2
3      PreTex: stem.pl
4
5      Implemented by LABIC
6
7
8          === PARAMETERS READ ===
9          language      = port
10         dirStemBase    = stembase
11         dirTextBase    = textbase
12         dirStopList    = stoplist
13         dirDiscover    = discover
14         dirGraphics    = graphics
15         dirPredict     = predict
16         logFile        = pretex.log
17
18         ENABLED        one gram
19         DISABLED       two gram
20         DISABLED       three gram
21
22
23         ### STOP LIST ###
24         Total StopWords 634
25         Total StopFiles 2
26
27     ....
28     Creating StemBase
29     === one ===
30     Total Stems : 5
31     Total Texts : 5
32     stembase/oneGram.all created!
33     stembase/oneGram.txt created!
34     stembase/stemWdST.all created!
35     stembase/stemWdTF.all created!
36
37
38     Stemize Success
39
40     Execution finished at Mon Jul  7 20:58:56 2003
41     Execution Time: 0 seconds
```

Figura 5.11: Exemplo de execução do módulo **stem.pl**

5.3.3 O Módulo **report.pl**

A base de *stems* (**stembase**) gerada pelo módulo **stem.pl** contém, entre outros, os arquivos com informações sobre as frequências dos *stems*, como o ilustrado na Figura 5.9 na página 87 para 1-gram. A partir desses arquivos, o PRETEXT cria a tabela atributo-valor, na qual cada documento é um exemplo, e os *stems* são os atributos dessa tabela, no formato da Tabela 5.1 na página 77.

O valor dos atributos da tabela atributo-valor pode ser calculado utilizando qualquer uma das medidas descritas na Seção 3.3 na página 44 e na Seção 5.2 na página 74. O módulo **report.pl** é também responsável, se requerido pelo usuário, pela aplicação dos cortes de Luhn e pelo uso de taxonomias. Para a execução desse módulo **report.pl** são necessários alguns arquivos apresentados a seguir.

5.3.3.1 Arquivos de Entrada

Os arquivos de entrada para o módulo **report.pl** consistem de:

- arquivo de parâmetros (**parameters.cfg**): é na execução do módulo **report.pl** que são consultados os parâmetros: **max**, **min**, **measure**, **smooth** e **normalize**.
- diretório **stembase**: a **stembase** gerada pelo módulo **stem.pl**. São consultados os arquivos **.all** e **.txt** para cada *gram* definido pelo usuário.
- arquivo de taxonomias (**taxonomy.txt**): este arquivo é opcional, ou seja, caso esteja disponível no mesmo diretório no qual foi invocado **report.pl**, PRETEXT realiza indução construtiva. A indução construtiva é caracterizada pela criação de novos atributos a partir de alguns atributos já existentes. Esses novos atributos, geralmente, são generalizações de dois ou mais atributos.

Por exemplo, considere os cinco *stems* **amig**, **cidad**, **trabalh**, **filh** e **cas** na Figura 5.9 na página 87, que são os *stems* correspondentes ao conjunto de documentos. Caso o usuário considere que os *stems* **cas** e **filh** podem ser agrupados em um único *stem*, no qual o usuário decide nomear **familia**, e os outros três *stems* **trabalh**, **cidad** e **amig** podem ser agrupados em um *stem* nomeado **outros** pelo usuário, então, o arquivo **taxonomia.txt** deve conter essa informação, como ilustrado a seguir.

```
familia: cas, filh
outros: trabalh, cidad, amig
```

Nesse caso, a tabela com as frequências dos cinco *stems* originais — Tabela 5.2 (a) — é transformada para a tabela ilustrada na Tabela 5.2 (b) com apenas dois atributos e cuja frequência é a soma das frequências dos atributos originais, como definido no arquivo `taxonomia.txt` correspondente.

Tabela 5.2: Exemplo de tabela gerada com o uso de taxonomias

documento	amig	ciudad	trabalh	filh	cas	FAMILIA	OUTROS
texto1.txt	0	0	0	0	6	6	0
texto2.txt	0	0	0	3	1	4	0
texto3.txt	0	0	1	1	1	2	1
texto4.txt	0	1	1	1	1	2	2
texto5.txt	5	6	10	8	8	16	21

(a)

(b)

É importante salientar que essa taxonomia é geralmente definida pelo usuário/especialista após a análise dos arquivos de *stems* gerado por **stem.pl**. As palavras que representam a junção de *stems*, como no exemplo, **familia** e **outros**, aparecem no arquivo de saída `.names` sempre com letras maiúsculas, para que possam ser diferenciadas dos *stems*.

5.3.3.2 Arquivos de Saída

A saída do módulo **report.pl** consiste de 2 (dois) arquivos que representam a tabela atributo-valor na sintaxe padrão do DISCOVER — Apêndice B — e arquivos com dados para geração de gráficos, descritos a seguir:

- arquivo `discover.names`: neste arquivo são declarados os atributos (*stems*) como ilustrado na Figura 5.12 (a);
- arquivo `discover.data`: este arquivo contém os valores dos atributos para cada exemplo e representa a tabela atributo-valor propriamente dita, como ilustrado na Figura 5.12 (b);

A primeira linha do arquivo `discover.names` representa o atributo da primeira coluna no `discover.data`, a segunda linha do `discover.names` representa o atributo da segunda coluna no `discover.data`, e assim por diante. O primeiro atributo, `filename`, é declarado sempre da mesma maneira para qualquer que seja o conjunto de documentos, pois ele representa o nome do arquivo no qual está armazenado o documento e, normalmente, não é utilizado pelos algoritmos de aprendizado de máquina.

- arquivos para geração de gráficos: são gerados arquivos com dados sobre o *rank* de *stems* os quais podem ser utilizados para gerar os gráficos no software que o usuário desejar. Além desses arquivos, o PRETEXT gera 3 *scripts* para **GnuPlot**TM (Crawford, 1998), `oneGnuPlot.script`, `twoGnuPlot.script` e `threeGnuPlot.script`, para cada conjunto de *gram* respectivos, os quais geram os gráficos no formato L^AT_EX (Kopla & Daly, 1999). Esses *scripts* podem ser facilmente alterados para gerar outros formatos, como `png` e `ps` alterando a primeira linha dos *scripts*.

Figura 5.12: Exemplo dos arquivos `.names` (a) e `.data` (b) na sintaxe padrão do DISCOVER

Por exemplo, a Figura 5.13, gerada pelo *script* `oneGnuPlot.script`, representa o *rank* de *stems* apresentado na Figura 5.7 na página 83. Apesar de simples, quando aplicado em conjunto de documentos maiores pode ser muito útil para auxiliar na determinação dos pontos de cortes de Luhn — Figura 3.2 na página 43.

Figura 5.13: Exemplo de gráfico gerado utilizando o *rank* de *stems*

5.3.3.3 Execução do Módulo `report.pl`

Com os parâmetros, arquivos e diretórios definidos corretamente, a execução de `report.pl` é feita invocando o módulo no mesmo diretório do arquivo de parâmetros.

Assim, para o PRETEXT criar os arquivos de dados (`.data`) e atributos (`.names`) no formato do DISCOVER, basta digitar na linha de comando **report.pl**.

Na Figura 5.14 está ilustrado um exemplo de arquivo de log de execução do **report.pl**. A partir da linha 7 até 19 são apresentados os parâmetros lidos pelo módulo. Como já mencionado, uma característica presente quando um conjunto de documentos em formato não estruturado é transformado em formato estruturado usando a abordagem *bag-of-words*, é que a tabela atributo-valor gerada é muito esparsa, ou seja, a porcentagem de valores não nulos é muito pequena. PRETEXT calcula essa porcentagem que é mostrada na linha 37 dessa figura.

```

1  Execution started at Mon Jul  7 20:52:41 2003
2
3  PreTex: report.pl
4  Implemented by LABIC
5
6
7      === PARAMETERS READ ===
8      language      = port
9      dirStemBase    = stembase
10     dirTextBase    = textbase
11     dirStopList    = stoplist
12     dirDiscover    = discover
13     dirGraphics    = graphics
14     dirPredict     = predict
15     logFile        = pretext.log
16     ##### one #####
17     measure        = freq
18     normalize      = disabled
19     smooth         = enabled
20     #####
21     Taxonomy was NOT Loaded
22
23     =====one=====
24
25  Loading stembase/oneGram.all
26
27  Number of stems loaded from stembase/oneGram.all    5
28
29  Loading stembase/oneGram.txt
30  .....
31  graphics/oneStdDev.dat created!
32  graphics/oneStdDevFF.dat created!
33  graphics/oneGnuPlot.script created!
34  === one ===
35  Total Stems : 5
36  Total Texts : 5
37  Matrix Density                                60%
38  === CREATING DISCOVER NAMES ===
39  Number of Stems                                5
40  Discover Names Created
41  === CREATING DISCOVER DATA ===
42  TotalPoints (point = text)                    5
43  .....
44  Number of Texts                                5
45  Discover Data Created
46
47
48  report Success
49
50  Execution finished at Mon Jul  7 20:52:41 2003
51  Execution time: 0 seconds

```

Figura 5.14: Exemplo de execução do módulo **report.pl**

5.3.4 Scripts Auxiliares

Foram implementados no PRETEXT vários *scripts* auxiliares, descritos a seguir:

- **checkWord.pl**: transforma uma palavra, fornecida como parâmetro, no *stem* correspondente. Esse *script* é útil para verificar o *stem* de uma palavra e para verificar se palavra é ou não uma *stopword* definida na **stoplist**. A execução desse *script* é feita invocando o commando **checkWord.pl <palavra>** no qual <palavra> é a palavra que se deseja verificar, como exemplificado na Figura 5.15:

```
1 c:\workdir>checkWord.pl algumas
2 algumas is a STOPWORD!!
3 word: algumas
4 stem: algum
```

Figura 5.15: Exemplo de execução do *script* **checkWord.pl**

Nesse exemplo, foi realizado uma consulta referente a palavra **algumas**. O *script* informa que essa palavra está atualmente declarada como *stopword*⁵ e também, informa ao usuário que **algum** é o *stem* dessa palavra.

- **predict.pl**: esse *script* é responsável pela transformação de novos documentos, no formato tabela atributo-valor, usando os mesmos parâmetros e mesmos *stems* utilizados na geração dos arquivos **discover.names** e **discover.data**. Ou seja, após gerados os arquivos **discover.names** e **discover.data**, esses arquivos podem ser submetidos a algum algoritmo de aprendizado na indução de um modelo — o conjunto de treinamento usado pelo algoritmo. Entretanto, o modelo induzido deve ser avaliado em um conjunto de dados independente — o conjunto de teste. No entanto, não é possível executar novamente os módulos **stem.pl** e **report.pl** para realizar a transformação desse novos documentos, pois, mesmo utilizando parâmetros idênticos aos realizados na transformação do conjunto de documentos usados para treinamento, o novo conjunto de documentos deve ser utilizado para testar o modelo induzido pelo algoritmo de AM, e pode ter *stems* diferentes. Assim, o *script* **predict.pl** é responsável pela transformação do novo conjunto de documentos utilizando os mesmos atributos (*stems*) do conjunto de documentos usado na indução do modelo.

⁵Um dos arquivos no diretório de *stopword* contém a palavra **algumas**.

Para a sua execução basta criar uma **textbase** com o nome “**predict**”, e não “**textbase**” — Seção 5.3.2, item **textbase** —, e executar na linha de comando **predict.pl**. A Figura 5.16 ilustra um exemplo de execução desse *script*.

```
1  Execution started at Thu Jul 17 20:26:56 2003
2
3  PreTex: predict.pl
4  Implemented by LABIC
5
6
7      === PARAMETERS READ ===
8      language      = port
9      dirStemBase    = stembase
10     dirTextBase    = textbase
11     dirStopList    = stoplist
12     dirDiscover    = discover
13     dirGraphics    = graphics
14     dirPredict     = predict
15     logFile        = pretext.log
16     ##### one #####
17     measure        = tfidf
18     normalize      = disabled
19     smooth         = enabled
20     #####
21     Taxonomy was NOT Loaded
22
23     ...
24
25     =====one=====
26
27     Loading stembase/oneGram.all
28
29     Number of stems loaded from stembase/oneGram.all    5
30
31     Loading stembase/oneGram.txt
32     .....
33     Loading TFIDF
34     Loading SMOOTH
35     === CREATING DISCOVER TEST ===
36     TotalPoints (point = text)                        3
37     ...
38     Number of Texts                                  3
39     Discover Test Created
40
41     report Success
42
43     Execution finished at Thu Jul 17 20:26:56 2003
44     Execution time: 0 seconds
```

Figura 5.16: Exemplo de execução do *script* **predict.pl**

5.4 Considerações Finais

O bom resultado em um processo de mineração usando algoritmos de aprendizado está diretamente relacionado não somente à quantidade mas, principalmente, à qualidade dos dados que descrevem uma situação do mundo real. Em um processo de mineração de textos, no qual os dados estão em um formato não estruturado textual, é necessário que esses dados sejam pré-processados de tal forma que possam ser submetidos a algoritmos de aprendizado. Neste capítulo foi descrita a ferramenta computacional PRETEXT que realiza o pré-processamento de dados textuais de forma interativa e iterativa.

PRETEXT é uma ferramenta desenvolvida usando o paradigma de orientação a objetos⁶. Dessa forma, apesar das diversas facilidades e funcionalidades já implementadas, a ferramenta está preparada para que outras funcionalidades possam facilmente ser incorporadas no futuro. As características atuais da ferramenta incluem: a transformação de documentos, usando a abordagem *bag-of-words*, em um formato estruturado de uma tabela atributo-valor, para documentos escritos em português, espanhol e inglês; a atribuição de valores aos atributos usando diversas medidas propostas na literatura e outras medidas propostas neste trabalho; a redução da dimensionalidade dos atributos; o uso de taxonomias por meio de indução construtiva; a geração de vários arquivos auxiliares com informações úteis ao usuário; a geração de arquivos para criar gráficos relacionados a frequência dos *stems*; entre outros. Essas facilidades, bem como a grande quantidade de informações geradas por PRETEXT, são o diferencial dessa ferramenta em relação a outras ferramentas existentes.

A saída de PRETEXT é um arquivo contendo os documentos representados na forma de uma tabela atributo-valor e um arquivo contendo informações sobre os atributos. Esses arquivos estão na sintaxe padrão utilizada pelo sistema DISCOVER e podem ser convertidos automaticamente para a sintaxe de dados de mais de dezesseis algoritmos de aprendizado usando o ambiente computacional DISCOVER LEARNING ENVIRONMENT - DLE (Batista & Monard, 2003), no qual PRETEXT será futuramente integrado. Como mencionado, outros recursos podem ser facilmente implementados na ferramenta PRETEXT, os quais deverão surgir naturalmente mediante o uso e a interação com pesquisadores de áreas correlatas à Tecnologia da Informação.

⁶A primeira versão da ferramenta foi desenvolvida por mim usando o paradigma estruturado. No entanto, devido ao fato do projeto DISCOVER, cuja versão inicial também utilizava o paradigma estruturado, ter sido reimplementado usando orientação a objetos, fez-se necessário que a ferramenta PRETEXT também seguisse o mesmo padrão. Dessa forma, a nova versão da ferramenta PRETEXT usando o paradigma de orientação a objetos foi implementada pelo Edson Takashi Matsubara. Na versão orientada a objetos foram inseridas novas funcionalidades as quais foram surgindo em decorrência dos experimentos realizados.

Experimentos Realizados Usando a Metodologia para Interpretação de Clusters

Neste capítulo são apresentados os experimentos realizados com o objetivo de extrair conhecimento de um conjunto de dados não rotulados, utilizando a metodologia proposta no Capítulo 4. Para analisar o desempenho e a eficácia dessa implementação da metodologia, uma série de experimentos foram realizados usando os algoritmos de aprendizado *AutoClass* e *See5* com dois conjuntos de dados diferentes. O primeiro é um conjunto de dados naturais¹, obtido do repositório da UCI. O segundo é um conjunto de dados reais, fornecido por pesquisadores da Universidade de Buenos Aires — Argentina. Para ambos conjuntos de dados são descritos os experimentos realizados e os resultados obtidos.

6.1 O Conjunto de Dados Breast-Cancer

Os experimentos, nesta seção, foram realizados usando o conhecido conjunto de dados (exemplos) naturais *Breast-cancer*. Esse conjunto de dados pertencem ao repositório da UCI (Merz & Murphy, 1998) e foram fornecidos pelo Instituto de Oncologia de Ljubljana. O conjunto de dados consiste de 285 descrições de pacientes representados por 9 atributos (5 discretos e 4 contínuos) e 2 classes predizendo a recorrência (29,47% dos exemplos) ou não recorrência (70,53%) do câncer de mama — Tabela 6.2. O conjunto de dados contém ruídos (menos de 1%) e valores ausentes nos exemplos. Na Tabela 6.2 são

¹Neste trabalho o termo *dados naturais* referem-se a dados provenientes de repositórios de dados, para distinguir de *dados reais*, que são colhidos diretamente de sistemas de armazenamento, e *dados artificiais*, que são criados artificialmente.

descritos os atributos do conjunto de dados Breast-cancer.

Tabela 6.1: Breast-cancer: descrição do conjunto de dados

# Ex	Atr (discretos, contínuos)	Classe (% dos exemplos)
285	9 (5, 4)	2 (29,5%, 70,5%)

Tabela 6.2: Breast-cancer: descrição dos atributos

Atr	Tipo	Descrição
01 age	contínuo	idade do paciente
02 age_at_meno	discreto	gt40; leq40; premenopaus
03 tumor_size	contínuo	tamanho do tumor de mama
04 involved_nodes	contínuo	número de nódulos envolvidos
05 node_capsule	discreto	yes; no
06 degree_of_malig	contínuo	grau de malignidade do tumor
07 breast	discreto	left; right
08 breast_quadrant	discreto	central; left_upper; left_lower; right_upper; right_lower
09 irradiation	discreto	yes; no
10 rec	discreto	recurrence; no_recurrence (Atributo Classe)

6.1.1 Configuração de Execução dos Experimentos

Considerando que a classe de cada exemplo é conhecida para esse conjunto de dados, os experimentos realizados foram divididos em dois grandes grupos:

- o atributo classe (atributo 10 rec na Tabela 6.2) foi ignorado na execução do **AutoClass**, ou seja, apenas os nove primeiros atributos foram usados para descobrir os clusters;
- o atributo classe foi considerado como um atributo comum na execução de **AutoClass**, ou seja, os dez atributos foram usados para descobrir os clusters. O objetivo nesse caso é verificar a influência do atributo classe no processo de agrupamento.

Os experimentos foram realizados utilizando os algoritmos de aprendizado **AutoClass** e **See5**. Praticamente, os algoritmos foram executados usando os valores *default* para seus parâmetros, exceto para o número de clusters a ser encontrado por **AutoClass** e para *10-fold cross-validation* no **See5**. No entanto, é importante salientar que os classificadores obtidos poderiam ter um melhor desempenho caso os parâmetros fossem ajustados em cada um dos experimentos realizados. Para a execução dos experimentos, portanto, foi utilizada a seguinte configuração:

1. **AutoClass**: os dados foram submetidos cinco vezes a **AutoClass** para diferentes números de clusters:
 - (a) fixando o número de clusters a ser encontrado em 2, 3, 5 e 7;
 - (b) sem fixar o número de clusters, sendo encontrados 10 clusters;
2. **INCLASS**: para cada resultado de **AutoClass**, foram criados dois novos conjuntos de dados:
 - (a) rotulando todos os exemplos do conjunto de dados original com o cluster ao qual pertencem;
 - (b) rotulando apenas os exemplos do conjunto de dados original que possuem probabilidade 1 de pertencer ao cluster;
3. **See5**: cada novo conjunto de dados com exemplos rotulados com o cluster ao qual pertencem, criado pelo **INCLASS** no passo anterior, foram submetidos ao **See5** considerando:
 - (a) todo o conjunto de dados como treinamento e teste;
 - (b) usando *10-fold cross-validation*.

Dessa forma, é possível obter e analisar vários resultados interessantes, tais como: (i) a quantidade de exemplos que pertencem a mais de um cluster considerando a probabilidade dos exemplos pertencer a cada cluster encontrado por **AutoClass**; (ii) o número de regras induzidas; (iii) os atributos mais relevantes utilizados na descrição das regras; (iv) os erros associados às regras induzidas pelo **See5**. Os resultados dos experimentos são apresentados a seguir.

6.1.2 Resultados Obtidos sem o Atributo Classe Original

AutoClass foi executado fixando o número de clusters em 2, 3, 5 e 7. Também, **AutoClass** foi executado sem fixar o número de clusters, sendo encontrado 10 clusters. Os resultados para cada experimento são apresentados nas tabelas respectivas a seguir, nas quais:

- ‘Cluster’ - identifica cada um dos clusters diferentes, denotado por $C_i(j)$, no qual i representa o número total de clusters em cada experimento, para $i \in \{2, 3, 5, 10\}$, e j identifica o cluster, para $j \in \{0, \dots, 9\}$;

- ‘P’ - parâmetro de entrada *MinProbability* do INCLASS, Algoritmo 1 na página 62. O valor 0 indica que todos os exemplos são considerados, ou seja, *MinProbability* ≥ 0 , enquanto que o valor 1 indica que apenas exemplos com probabilidade 1 de pertencer ao cluster são considerados;
- ‘# Ex’ - número de exemplos no novo conjunto de dados criado por INCLASS;
- ‘Rec’ e ‘nRec’ - número de exemplos com classes **recurrence** e **no_recurrence**, respectivamente, que pertencem ao cluster $C_i(j)$;
- ‘# ExCl’ - número de exemplos em cada cluster;
- ‘% ExCl’ - distribuição dos exemplos em cada cluster;
- ‘ErrCM %’ - porcentagem do erro da classe majoritária;
- ‘ErrA %’ - porcentagem do erro aparente de **See5**, isto é, o erro obtido quando se utiliza todo o conjunto de dados como treinamento e teste;
- ‘Err10cv %’ - porcentagem da estimativa do erro verdadeiro de **See5** obtido usando *10-fold cross-validation* e o erro padrão²;
- ‘# R’ - número de regras induzidas por **See5** utilizando todos os exemplos para treinar e testar;
- ‘# MR’ - número médio de regras induzidas por **See5** obtido usando *10-fold cross-validation* e o erro padrão.

6.1.2.1 Fixando o Número de clusters em 2

Os resultados obtidos com o número de clusters fixado em 2 são mostrados na Tabela 6.3.

Tabela 6.3: Breast-cancer: resultados - 2 clusters

Cluster	P	# Ex	Rec	nRec	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_2(0)$			19	127	146	51,2					
$C_2(1)$	1	285	65	74	139	48,8	48,8	0,0	$0,0 \pm 0,0$	2	$2,0 \pm 0,0$

Esses resultados mostram que a força preditiva do modelo induzido por **See5** para os dois clusters descobertos por **AutoClass** é muito boa, visto que o erro verdadeiro — Err10cv — é zero. Também, pode ser observado que nenhum dos exemplos pertencem a

²Do inglês *standard error*. O *standard error* é calculado usando a fórmula $\frac{\sigma}{\sqrt{n}}$ no qual σ é o desvio padrão e n é o grau de liberdade da amostra.

mais de um cluster, nem mesmo com uma pequena probabilidade. De fato, do total de 285 exemplos, todos eles pertencem a um dos clusters com probabilidade igual a 1.

Na Tabela 6.4 são mostradas as duas regras induzidas por *See5* para esses dois clusters. Para cada regra, *See5* mostra o número de exemplos cobertos pela regra e um fator de confiança da regra (o número entre 0 e 1 dentro do símbolo “[]”). Como *See5* induz regras *não ordenadas* (Baranauskas & Monard, 2000b), o mesmo exemplo pode ser coberto por mais de uma regra. Esse não é o caso para esses dois clusters, pois as regras são disjuntas, cobrindo todos os 285 exemplos usando apenas o atributo `involved_nodes`.

<p>Rule 1: (cover 146)</p> <p style="padding-left: 20px;"><code>involved_nodes <= 0</code></p> <p>-> class C2(0) [0.993]</p>	<p>Rule 2: (cover 139)</p> <p style="padding-left: 20px;"><code>involved_nodes > 0</code></p> <p>-> class C2(1) [0.993]</p>
--	---

Default class: C2(0)

Tabela 6.4: Breast-cancer: regras induzidas - 2 clusters

A interpretação da hipótese induzida é muito simples:

- exemplos com valores menores ou iguais a 0 para o atributo `involved_nodes`, pertencem ao cluster denotado C2(0) — (Rule 1);
- exemplos com valores maiores que 0 para o atributo `involved_nodes` pertencem ao cluster denotado C2(1) — (Rule 2).

6.1.2.2 Fixando o Número de Clusters em 3

O experimento foi repetido fixando em 3 o número de clusters. Na Tabela 6.5 são apresentados os resultados obtidos.

Tabela 6.5: Breast-cancer: resultados - 3 clusters

Cluster	P	# Ex	Rec	nRec	# ExCl	% ExCl	ErrCM %	ErrA %	Erro10cv %	# R	# MR
$C_3(0)$	1	285	65	74	139	48,8	51,2	0,0	0,0	3	$3,0 \pm 0,0$
$C_3(1)$			18	120	138	48,4					
$C_3(2)$			1	7	8	2,8					

Como no experimento com 2 clusters, o erro verdadeiro do classificador é zero e as regras induzidas são poucas. Cada exemplo no conjunto de 285 exemplos pertence a apenas um dos três clusters com probabilidade igual a 1. Na Tabela 6.6 são mostradas as regras induzidas por *See5* para esses três clusters. É interessante observar que a regra

Rule 1 na Tabela 6.6 é igual a regra Rule 2 na Tabela 6.4, cobrindo os mesmos 139 exemplos. Os outros 146 exemplos são cobertos por duas regras disjuntas Rule 2 e 3 — Tabela 6.6.

```

Rule 1: (cover 139)
    involved_nodes > 0
-> class C3(0) [0.993]

Rule 2: (cover 138)
    tumor_size > 0
    involved_nodes <= 0
-> class C3(1) [0.993]

Rule 3: (cover 8)
    tumor_size <= 0
-> class C3(2) [0.900]

Default class: C3(0)

```

Tabela 6.6: Breast-cancer: regras induzidas - 3 clusters

Apenas dois atributos (`involved_nodes`, `tumor_size`) foram usados para representar o modelo (conjunto de regras). Como no experimento anterior, as regras cobrem todos os exemplos e cada exemplo é coberto por apenas uma regra.

6.1.2.3 Fixando o Número de Clusters em 5

Os resultados obtidos com o número de clusters fixado em 5 são mostrados na Tabela 6.7.

Tabela 6.7: Breast-cancer: resultados - 5 clusters

Cluster	P	# Ex	Rec	nRec	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_5(0)$			18	120	138	48,4					
$C_5(1)$			17	43	60	21,1					
$C_5(2)$	1	285	40	20	60	21,1	51,6	0,0	0,0	5	$5,0 \pm 0,0$
$C_5(3)$			8	11	19	6,7					
$C_5(4)$			1	7	8	2,8					

Novamente, o erro do classificador é zero e o número de regras induzidas é pequeno (5 regras). Todos os exemplos pertencem a um cluster com probabilidade igual a 1. Na Tabela 6.8 são mostradas as regras induzidas por **See5** para esses cinco clusters. Nessa tabela, as regras Rule 1 e a Rule 5 são, respectivamente, iguais a Rule 2 e Rule 4 na Tabela 6.6, cobrindo os mesmos 138 e 8 exemplos respectivamente. Os outros 139 exemplos são cobertos por regras disjuntas Rule 2, 3 e 4.

Nesse caso, o número de atributos necessários para representar o conjunto de regras é igual a 3 (`involved_nodes`, `tumor_size`, `degree_of_malig`). Como nos experimentos anteriores, as regras cobrem todos os exemplos e cada exemplo é coberto por apenas uma regra.

```

Rule 1: (cover 138)
  tumor_size > 0
  involved_nodes <= 0
-> class C5(0) [0.993]

Rule 2: (cover 60)
  involved_nodes > 0
  degree_of_malig > 1
  degree_of_malig <= 2
-> class C5(1) [0.984]

Rule 3: (cover 60)
  involved_nodes > 0
  degree_of_malig > 2
-> class C5(2) [0.984]

Rule 4: (cover 19)
  involved_nodes > 0
  degree_of_malig <= 1
-> class C5(3) [0.952]

Rule 5: (cover 8)
  tumor_size <= 0
-> class C5(4) [0.900]

Default class: C5(0)

```

Tabela 6.8: Breast-cancer: regras induzidas - 5 clusters

6.1.2.4 Fixando o Número de Clusters em 7

Na Tabela 6.9 são apresentados os resultados obtidos com o número de clusters fixado em 7.

Tabela 6.9: Breast-cancer: resultados - 7 clusters

Cluster	P	# Ex	Rec	nRec	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_7(0)$			9	56	65	22,8					
$C_7(1)$			17	43	60	21,1					
$C_7(2)$			40	20	60	21,1					
$C_7(3)$	1	285	4	45	49	17,2	77,2	0,4	$0,4 \pm 0,4$	7	$7,0 \pm 0,0$
$C_7(4)$			5	19	24	8,4					
$C_7(5)$			8	11	19	6,7					
$C_7(6)$			1	7	8	2,8					

O conjunto de regras é composto por 7 regras e, embora o erro verdadeiro não seja zero, ele é muito pequeno ($0,4\% \pm 0,4\%$). Como anteriormente, todos os exemplos pertencem a somente um cluster com probabilidade igual a 1. As regras induzidas por *See5* são mostradas na Tabela 6.10.

Os atributos que representam o conjunto de regras (*involved_nodes*, *tumor_size*, *degree_of_malig*) são os mesmos utilizados nos experimentos anteriores com 5 clusters. O conjunto de regras cobre todos os exemplos e apenas um exemplo é coberto por mais de uma regra.

De fato, até 7 clusters existe uma hierarquia de cluster, como mostrado na Figura 6.1, na qual, em cada nível, exemplos nos clusters um nível anterior são os mesmos ou são, posteriormente, divididos em clusters disjuntos. Deve ser observado que *AutoClass* não usa a técnica hierárquica para descobrir clusters. Portanto, esse é um resultado interessante

<p>Rule 1: (cover 65) tumor_size > 0 involved_nodes <= 0 degree_of_malig > 1 degree_of_malig <= 2 -> class C7(0) [0.985]</p>	<p>Rule 2: (cover 60) involved_nodes > 0 degree_of_malig > 1 degree_of_malig <= 2 -> class C7(1) [0.984]</p>
<p>Rule 3: (cover 60) involved_nodes > 0 degree_of_malig > 2 -> class C7(2) [0.984]</p>	<p>Rule 4: (cover 49) tumor_size > 0 involved_nodes <= 0 degree_of_malig <= 1 -> class C7(3) [0.980]</p>
<p>Rule 5: (cover 25) involved_nodes <= 0 degree_of_malig > 2 -> class C7(4) [0.926]</p>	<p>Rule 6: (cover 19) involved_nodes > 0 degree_of_malig <= 1 -> class C7(5) [0.952]</p>
<p>Rule 7: (cover 8) tumor_size <= 0 -> class C7(6) [0.900]</p>	<p>Default class: C7(0)</p>

Tabela 6.10: Breast-cancer: regras induzidas - 7 clusters

que pode ser analisado pelo especialista do domínio para dar alguma significado a esses clusters.

Figura 6.1: Hierarquia de clusters

6.1.2.5 Sem Fixar o Número de Clusters

Quando o número de clusters não foi fixado, **AutoClass** encontrou 10 clusters. Na Tabela 6.11 são mostrados os resultados obtidos.

Tabela 6.11: Breast-cancer: resultados - 10 clusters

Cluster	P	# Ex	Rec	nRec	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_{10}(0)$	0	285	40	20	60	21,1	79,0	0,7	$1,0 \pm 0,5$	11	$10,9 \pm 0,1$
$C_{10}(1)$			4	45	49	17,2					
$C_{10}(2)$			17	26	43	15,1					
$C_{10}(3)$			4	33	37	13,0					
$C_{10}(4)$			5	23	28	9,8					
$C_{10}(5)$			5	19	24	8,4					
$C_{10}(6)$			0	17	17	6,0					
$C_{10}(7)$			7	5	12	4,2					
$C_{10}(8)$			1	7	8	2,8					
$C_{10}(9)$			1	6	7	2,5					
$C_{10}(0)$	1	199	40	20	60	30,2	69,9	0,0	$1,0 \pm 0,7$	7	$7,0 \pm 0,0$
$C_{10}(1)$			4	45	49	24,6					
$C_{10}(2)$			17	26	43	21,6					
$C_{10}(3)$			0	8	8	4,0					
$C_{10}(4)$			0	0	0	0,0					
$C_{10}(5)$			5	19	24	12,1					
$C_{10}(6)$			0	0	0	0,0					
$C_{10}(7)$			0	0	0	0,0					
$C_{10}(8)$			1	7	8	4,0					
$C_{10}(9)$			1	6	7	3,5					

Como pode ser observado, existem 86 exemplos ($285-199 = 86$ na Tabela 6.11) que pertencem a mais de um cluster ou pertencem a um dos clusters com probabilidade 0,999. Para **AutoClass** essa probabilidade é considerada como próxima de 1 e, portanto, para esses exemplos não são consideradas probabilidades de 0,001 dos exemplos pertencerem a um outro cluster. Assim, os clusters com o maior número de exemplos sobrepostos são:

- 13 exemplos pertencem ao clusters 2 e 6 simultaneamente, denotado por $C_{10}(2,6)$;
- 31 exemplos pertencem ao clusters 3 e 4 simultaneamente, denotado por $C_{10}(3,4)$;
- 7 exemplos pertencem ao clusters 7 e 9 simultaneamente, denotado por $C_{10}(7,9)$;

como mostrado na Tabela 6.12 e ilustrado na Figure 6.2. A força preditiva do modelo induzido por **See5** é muito boa, visto que o erro é muito pequeno ($1,0\% \pm 0,5\%$). Isso deve ocorrer porque cada um dos 86 exemplos pertencem a um cluster com alta probabilidade, e a outro cluster com uma probabilidade muito baixa.

Figura 6.2: Breast-cancer: clusters descobertos por AutoClass - 10 clusters

Tabela 6.12: Breast-cancer: clusters sobrepostos - 10 clusters

Clusters Sobrepostos	# Ex Comuns
$C_{10}(2, 6)$	13
$C_{10}(3, 4)$	31
$C_{10}(7, 9)$	7

Como mostrado na Tabela 6.11, **See5** induziu 11 (onze) regras para $P \geq 0$ e 7 (sete) regras para $P = 1$. Os atributos usados nessas regras são mostrados na Tabela 6.13, na qual o símbolo “•” refere-se a $P \geq 0$ e “o” a $P = 1$. Pode ser observado que apenas quatro atributos (**age_at_meno**, **tumor_size**, **involved_nodes**, **degree_of_malig**) são relevantes para predizer os exemplos em cada cluster.

Todas as regras induzidas por **See5** tem um alto fator de confiança, como mostrado na Tabela 6.14 para $P \geq 0$ e na Tabela 6.15 para $P = 1$.

Os atributos **age_at_meno**, **tumor_size**, **involved_nodes** e **degree_of_malig** foram usados no conjunto de regras para $P \geq 0$. O número total de exemplos cobertos pelas regras é 287. Isso significa que pelo menos dois exemplos ($287 - 285$) são cobertos por mais de uma regra, o que representa apenas 0,7% dos exemplos no conjunto de dados. O número de regras usando $P = 1$ diminuiu de 11 para 7 e apenas os atributos **tumor_size**, **involved_nodes** e **degree_of_malig** foram usados para representar o conjunto de regras induzido.

Como mencionado anteriormente, o atributo classe do conjunto de dados Breast-cancer não foi considerado nos experimentos descritos. Entretanto, os clusters descobertos por **AutoClass** em todos os experimentos descritos não estão relacionados ao conceito das classes **recurrence/no_recurrence** do conjunto de dados original. Por exemplo, pode ser observado na Tabela 6.11 que exemplos de ambas as classes pertencem ao mesmo cluster

Tabela 6.13: Breast-cancer: atributos no conjunto de regras - 10 clusters

	Atributos								
	age	age_at _meno	tumor _size	involved _nodes	node_ capsule	degree_of _malig	breast	breast_ quadrant	irradiation
$C_{10}(0)$				● ○		● ○			
$C_{10}(1)$			● ○	● ○		● ○			
$C_{10}(2)$				● ○		● ○			
$C_{10}(3)$		●	● ○	● ○		● ○			
$C_{10}(4)$		●		●		●			
$C_{10}(5)$			○	● ○		● ○			
$C_{10}(6)$				●		●			
$C_{10}(7)$				●		●			
$C_{10}(8)$			● ○						
$C_{10}(9)$				● ○		● ○			

para $P = 1$. Para $P \geq 0$, embora $C_{10}(6)$ tem apenas exemplos da classe **no_recurrence**, todos pertencem ao cluster com probabilidade menor do que 1. Isso significa que **AutoClass** agrupou os dados sem considerar o atributo classe original como altamente relevante.

A seguir, são descritos os resultados obtidos repetindo os mesmos experimentos, porém, usando o atributo classe como um atributo a mais.

6.1.3 Resultados Obtidos Usando o Atributo Classe Original

Os mesmos experimentos descritos na Seção 6.1.1 na página 98 foram realizados, fixando o número de clusters em 2, 3, 5, 7 e, também, deixando **AutoClass** determinar o número de clusters automaticamente. Neste último caso, **AutoClass** novamente encontrou 10 clusters.

Os resultados obtidos com o número de clusters fixado em 2, 3 e 5 clusters foram idênticos aos anteriores. Em outras palavras, o atributo classe nesse conjunto de dados não é considerado relevante por **AutoClass** para agrupar os exemplos. No experimento com 7 clusters, apenas o erro aparente — ErrA — diminuiu de $0,4\% \pm 0,4\%$ para $0,3\% \pm 0,3\%$. Para o experimento que **AutoClass** determina o número de clusters, os resultados são similares, e são mostrados na Tabela 6.16.

Rule 1: (cover 60) involved_nodes > 0 degree_of_malig > 2 -> class C10(0) [0.984]	Rule 2: (cover 49) tumor_size > 0 involved_nodes <= 0 degree_of_malig <= 1 -> class C10(1) [0.980]
Rule 3: (cover 43) involved_nodes > 1 degree_of_malig > 1 degree_of_malig <= 2 -> class C10(2) [0.978]	Rule 4: (cover 37) age_at_meno = premenopaus tumor_size > 0 involved_nodes <= 0 degree_of_malig > 1 degree_of_malig <= 2 -> class C10(3) [0.974]
Rule 5: (cover 28) age_at_meno = leq40 involved_nodes <= 0 degree_of_malig > 1 degree_of_malig <= 2 -> class C10(4) [0.933]	Rule 6: (cover 1) age_at_meno = gt40 involved_nodes <= 0 degree_of_malig > 1 -> class C10(4) [0.667]
Rule 7: (cover 25) involved_nodes <= 0 degree_of_malig > 2 -> class C10(5) [0.926]	Rule 8: (cover 17) involved_nodes > 0 involved_nodes <= 1 degree_of_malig > 1 degree_of_malig <= 2 -> class C10(6) [0.947]
Rule 9: (cover 12) involved_nodes > 0 involved_nodes <= 1 degree_of_malig <= 1 -> class C10(7) [0.929]	Rule 10: (cover 8) tumor_size <= 0 -> class C10(8) [0.900]
Rule 11: (cover 7) involved_nodes > 1 degree_of_malig <= 1 -> class C10(9) [0.889]	Default class: C10(0)

Tabela 6.14: Breast-cancer: regras induzidas ($P = 0$) - 10 clusters

Quando os resultados na Tabela 6.16 são comparados com os resultados na Tabela 6.11 na página 105, pode ser observado que o erro verdadeiro e o número de regras induzidas diminuiu para o experimento com o atributo classe. De um número total de 285 exemplos, apenas 43 (15%) exemplos (285 - 242) — Tabela 6.16 — pertencem ao cluster com probabilidade menor que 1. Deve ser observado que, quando o atributo classe não foi usado, Tabela 6.11, o número de exemplos que pertencem ao mesmo cluster com probabilidade menor do que 1 é 86 (285 - 199). Portanto, nesse caso, existem 50% mais exemplos que pertencem a mais de um cluster. Na Tabela 6.17 são mostrados os clusters sobrepostos e na Figura 6.3 são ilustrados graficamente os clusters encontrados.

Na Tabela 6.18 são mostrados os atributos usados por *See5* para $P \geq 0$ e $P = 1$ usados para descrever os 10 clusters.

```

Rule 1: (cover 60)
    involved_nodes > 0
    degree_of_malig > 2
-> class C10(0) [0.984]

Rule 2: (cover 49)
    tumor_size > 0
    involved_nodes <= 0
    degree_of_malig <= 1
-> class C10(1) [0.980]

Rule 3: (cover 43)
    involved_nodes > 0
    degree_of_malig > 1
    degree_of_malig <= 2
-> class C10(2) [0.978]

Rule 4: (cover 8)
    tumor_size > 0
    involved_nodes <= 0
    degree_of_malig > 1
    degree_of_malig <= 2
-> class C10(3) [0.900]

Rule 5: (cover 24)
    tumor_size > 0
    involved_nodes <= 0
    degree_of_malig > 2
-> class C10(5) [0.962]

Rule 6: (cover 8)
    tumor_size <= 0
-> class C10(8) [0.900]

Rule 7: (cover 7)
    involved_nodes > 0
    degree_of_malig <= 1
-> class C10(9) [0.889]

Default class: C10(0)

```

Tabela 6.15: Breast-cancer: regras induzidas ($P = 1$) - 10 clusters

Tabela 6.16: Breast-cancer: resultados usando o atributo classe - 10 clusters

Cluster	P	# Ex	Rec	nRec	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_{10}(0)$	0	285	9	56	65	22,8	77,2	0,4	$0,4 \pm 0,4$	10	$10,0 \pm 0,0$
$C_{10}(1)$			40	20	60	21,1					
$C_{10}(2)$			4	45	49	17,1					
$C_{10}(3)$			13	20	33	11,6					
$C_{10}(4)$			5	19	24	8,4					
$C_{10}(5)$			0	17	17	6,0					
$C_{10}(6)$			7	5	12	4,2					
$C_{10}(7)$			4	6	10	3,5					
$C_{10}(8)$			1	7	8	2,8					
$C_{10}(9)$			1	6	7	2,5					
$C_{10}(0)$	1	242	9	56	65	26,9	73,3	0,4	$0,4 \pm 0,4$	7	$7,0 \pm 0,0$
$C_{10}(1)$			39	21	60	24,8					
$C_{10}(2)$			4	45	49	20,3					
$C_{10}(3)$			13	16	29	12,0					
$C_{10}(4)$			5	19	24	9,9					
$C_{10}(5)$			0	0	0	0,0					
$C_{10}(6)$			0	0	0	0,0					
$C_{10}(7)$			0	0	0	0,0					
$C_{10}(8)$			1	7	8	3,3					
$C_{10}(9)$			1	6	7	2,9					

O conjunto de regras induzido por *See5* considerando $P \geq 0$ é mostrado na Tabela 6.19. Pode ser observado que quatro atributos (*age*, *tumor_size*, *involved_nodes*, *degree_of_malig*) foram suficientes para representar o conjunto de regras. O número de exemplos cobertos

Tabela 6.17: Breast-cancer: clusters sobrepostos usando o atributo classe - 10 clusters

Clusters Sobrepostos	# Ex Comuns
$C_{10}(3, 5)$	7
$C_{10}(3, 7)$	9
$C_{10}(5, 7)$	2
$C_{10}(6, 9)$	5

Figura 6.3: Breast-cancer: clusters sobrepostos usando o atributo classe - 10 clusters

Tabela 6.18: Breast-cancer: atributos no conjunto de regras usando o atributo classe - 10 clusters

	Atributos								
	age	age_at_meno	tumor_size	involved_nodes	node_capsule	degree_of_malig	breast	breast_quadrant	irradiation
$C_{10}(0)$			● ○	● ○		● ○			
$C_{10}(1)$				● ○		● ○			
$C_{10}(2)$			● ○	● ○		● ○			
$C_{10}(3)$	●			● ○		● ○			
$C_{10}(4)$				● ○		● ○			
$C_{10}(5)$				●		●			
$C_{10}(6)$				●		●			
$C_{10}(7)$	●			●		●			
$C_{10}(8)$			● ○						
$C_{10}(9)$				● ○		● ○			

por esse conjunto de regras é 286. Isso significa que apenas um exemplo ($286 - 285$) é coberto por mais de um regra, o que representa 0,35% dos exemplos nesse conjunto de dados.

<p>Rule 1: (cover 65) tumor_size > 0 involved_nodes <= 0 degree_of_malign > 1 degree_of_malign <= 2 -> class C10(0) [0.985]</p>	<p>Rule 2: (cover 60) involved_nodes > 0 degree_of_malign > 2 -> class C10(1) [0.984]</p>
<p>Rule 3: (cover 49) tumor_size > 0 involved_nodes <= 0 degree_of_malign <= 1 -> class C10(2) [0.980]</p>	<p>Rule 4: (cover 33) age <= 58 involved_nodes > 1 degree_of_malign > 1 degree_of_malign <= 2 -> class C10(3) [0.971]</p>
<p>Rule 5: (cover 25) involved_nodes <= 0 degree_of_malign > 2 -> class C10(4) [0.926]</p>	<p>Rule 6: (cover 17) involved_nodes > 0 involved_nodes <= 1 degree_of_malign > 1 degree_of_malign <= 2 -> class C10(5) [0.947]</p>
<p>Rule 7: (cover 12) involved_nodes > 0 involved_nodes <= 1 degree_of_malign <= 1 -> class C10(6) [0.929]</p>	<p>Rule 8: (cover 10) age > 58 involved_nodes > 1 degree_of_malign > 1 degree_of_malign <= 2 -> class C10(7) [0.917]</p>
<p>Rule 9: (cover 8) tumor_size <= 0 -> class C10(8) [0.900]</p>	<p>Rule 10: (cover 7) involved_nodes > 1 degree_of_malign <= 1 -> class C10(9) [0.889]</p>
<p>Default class: C10(0)</p>	

Tabela 6.19: Breast-cancer: regras induzidas usando o atributo classe ($P = 0$) - 10 clusters

O conjunto de regras induzidas por *See5* considerando $P = 1$ é mostrado na Tabela 6.20. O número de regras induzidas diminuiu de 10 ($P \geq 0$) para 7 ($P = 1$). Além do mais, apenas três atributos (*tumor_size*, *involved_nodes*, *degree_of_malign*) foram suficientes para representar o conjunto de regras para $P = 1$. Exceto para o último experimento (com 10 clusters), os experimentos incluindo o atributo classe descobriram os mesmos clusters que os experimentos realizados sem considerar o atributo classe.

Pode-se então concluir que para esse conjunto de dados, o atributo classe quando considerado como outro atributo qualquer, não tem uma função importante para *Auto-Class*, que busca por uma classificação de probabilidade máxima *a posteriori* nos exemplos. Informações de como os exemplos foram agrupados, quais os atributos são considerados mais relevantes, tanto pelo algoritmo não supervisionado quanto pelo supervisionado uti-

<p>Rule 1: (cover 65)</p> <pre>tumor_size > 0 involved_nodes <= 0 degree_of_malig > 1 degree_of_malig <= 2 -> class C10(0) [0.985]</pre>	<p>Rule 2: (cover 60)</p> <pre>involved_nodes > 0 degree_of_malig > 2 -> class C10(1) [0.984]</pre>
<p>Rule 3: (cover 49)</p> <pre>tumor_size > 0 involved_nodes <= 0 degree_of_malig <= 1 -> class C10(2) [0.980]</pre>	<p>Rule 4: (cover 29)</p> <pre>involved_nodes > 0 degree_of_malig > 1 degree_of_malig <= 2 -> class C10(3) [0.968]</pre>
<p>Rule 5: (cover 25)</p> <pre>involved_nodes <= 0 degree_of_malig > 2 -> class C10(4) [0.926]</pre>	<p>Rule 6: (cover 8)</p> <pre>tumor_size <= 0 -> class C10(8) [0.900]</pre>
<p>Rule 7: (cover 7)</p> <pre>involved_nodes > 0 degree_of_malig <= 1 -> class C10(9) [0.889]</pre>	<p>Default class: C10(0)</p>

Tabela 6.20: Breast-cancer: regras induzidas usando o atributo classe ($P = 1$) - 10 clusters

lizado, auxiliam o especialista a ter uma compreensão melhor dos dados, realizar uma análise minuciosa dos resultados obtidos com as diversas informações a ser utilizada na tomada de decisões, ou mesmo ajustar o(s) algoritmo(s) ao conjunto de dados. Por exemplo, nesse conjunto de dados, após o agrupamento realizado por **AutoClass**, **See5** induziu um conjunto de regras utilizando apenas quatro (dos nove) atributos, os quais o especialista pode usar para “explicar” os clusters encontrados.

6.2 O Conjunto de Dados EPH

O conjunto de dados reais utilizado neste trabalho foi fornecido pelo grupo de estatística da Universidade de Buenos Aires - Argentina. Esse conjunto de dados é grande e contém dados sigilosos. Os dados são referentes a uma Pesquisa Permanente de Domicílio (EPH³), que é um programa nacional da Argentina, juntamente com a Administração Estadual de Estatística (DPE⁴) daquele país.

A pesquisa consiste em caracterizar a população considerando informações de distribuição social, demográfica e produção de aglomerados urbanos do país. O objetivo do especialista é observar o comportamento dos dados obtidos por essa pesquisa usando

³*Encuesta Permanente de Hogares.*

⁴*Direcciones Provinciales de Estadística.*

tanto métodos estatísticos ([Matsudo, 2000](#)) quanto outros métodos, como os algoritmos de aprendizado utilizados nesse trabalho, na busca por padrões relacionados com a população do país.

O conjunto de dados inicial foi convertido em uma tabela atributo-valor contendo 185 atributos. Alguns passos críticos na preparação e na redução dos dados foram necessários. Uma limpeza no conjunto de dados original foi realizada, juntamente com o especialista, para remover exemplos com muitos valores ausentes, determinar os atributos menos relevantes, bem como combinar e discretizar vários atributos. Após esses passos, o conjunto de dados original foi reduzido a 4648 exemplos com 40 atributos (31 discretos e 9 contínuos), como mostrado na Tabela 6.21. Esse conjunto de dados não contém o atributo classe associado.

Tabela 6.21: EPH: descrição do conjunto de dados

# Ex	Atr (discretos, contínuos)
4648	40 (31, 9)

6.2.1 Configuração de Execução dos Experimentos

Os diversos experimentos, usando esse conjunto de dados, foram realizados de acordo com os parâmetros definidos na Seção 6.1.1 na página 98. No entanto, o atributo classe nesse conjunto de dados não é conhecido e, portanto, os experimentos realizados, diferentemente dos realizados com o conjunto de dados Breast-cancer, consistiram de experimentos no primeiro grupo — Seção 6.1.1. Os parâmetros para a execução de **AutoClass** foram definidos como:

1. **AutoClass**: os dados foram submetidos cinco vezes a **AutoClass** para diferentes números de clusters:
 - (a) fixando o número de clusters a ser encontrado em 2, 3, 5 e 10;
 - (b) sem fixar o número de clusters, sendo encontrados 25 clusters;

Nesse último caso, por ser um número relativamente alto de clusters, o que dificulta a indução de um bom modelo no sentido de explicar esses clusters, não foram realizados usando esse parâmetro.

6.2.2 Resultados Obtidos

O resultado de cada um dos experimentos é sintetizado em uma tabela semelhante a descrita na Seção 6.1.2 na página 99. Esses resultados são descritos a seguir.

6.2.2.1 Fixando o Número de Clusters em 2

Os resultados obtidos com o número de clusters fixado em 2 são mostrados na Tabela 6.22.

Tabela 6.22: EPH: resultados - 2 clusters

Cluster	P	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_2(0)$	0	4648	68,1	31,9	0,2	$0,4 \pm 0,2$	5	$4,8 \pm 0,1$
$C_2(1)$			31,9					
$C_2(0)$	1	4475	67,0	33,0	0,1	$0,1 \pm 0,0$	5	$4,6 \pm 0,2$
$C_2(1)$			33,0					

Como pode ser observado, o poder preditivo do modelo induzido por **See5** utilizando 2 clusters é muito bom, apresentando um erro verdadeiro muito baixo e um número pequeno de regras induzidas. Do número total de 4648 exemplos utilizados, somente 273 exemplos ($4648 - 4475$) pertencem a algum cluster com probabilidade menor que 1, o que representa somente 5,9% do total de exemplos. Em outras palavras, há poucos exemplos que pertencem a mais de um cluster.

As regras induzidas por **See5** para esse conjunto de dados são mostradas nas Tabelas 6.23 e 6.24, para os exemplos com probabilidade 0 e para os exemplos com probabilidade 1 de pertencerem aos clusters, respectivamente.

A primeira regra mostrada na Tabela 6.23 cobre mais de 50% dos exemplos e apenas três atributos (P22M, P47, P24), de um total de quarenta atributos, foram suficientes para representar todo o conjunto de regras. O número total de exemplos cobertos por essas regras é 4758 ($2929 + 236 + 1393 + 179 + 21$). Considerando que **See5** induz regras não ordenadas, isso indica que, no máximo, 110 exemplos são cobertos por mais de uma regra. Ainda, o fator de confiança das regras é alto.

É possível observar que os conjuntos de regras induzidas por **See5** são semelhantes para os dois conjuntos de regras, usando todos os exemplos ($P \geq 0$) e somente um subconjunto de dados ($P = 1$), diferindo apenas no número de exemplos cobertos e a confiança de cada regra. Como mencionado, os dados são sigilosos e, portanto, os atributos estão

Rule 1: (cover 2929) P22M <= 0 P47 = 1 -> class C2(0) [0.999]	Rule 2: (cover 236) P22M <= 0 P47 = 9 -> class C2(0) [0.992]
Rule 3: (cover 1393) P22M > 0 -> class C2(1) [0.999]	Rule 4: (cover 179) P47 = 2 -> class C2(1) [0.978]
Rule 5: (cover 23) P24 = 9 -> class C2(1) [0.960]	
Default class: C2(1)	

Tabela 6.23: EPH: regras induzidas usando o atributo classe ($P = 0$) - 2 clusters

Rule 1: (cover 2784) P22M <= 0 P47 = 1 -> class C2(0) [0.999]	Rule 2: (cover 216) P22M <= 0 P47 = 9 -> class C2(0) [0.995]
Rule 3: (cover 1393) P22M > 0 -> class C2(1) [0.999]	Rule 4: (cover 177) P47 = 2 -> class C2(1) [0.989]
Rule 5: (cover 21) P24 = 9 -> class C2(1) [0.957]	
Default class: C2(0)	

Tabela 6.24: EPH: regras induzidas usando o atributo classe ($P = 1$) - 2 clusters

codificados. Dessa forma, as regras induzidas por *See5* foram interpretadas pelo especialista.

6.2.2.2 Fixando o Número de Clusters em 3

Os resultados obtidos com o número de clusters fixado em 3 foram também considerados bons, os quais são apresentados na Tabela 6.25.

Pode ser observado que o número de exemplos que pertencem a um cluster com probabilidade 1 é ainda maior que no experimento anterior com 2 clusters. Isso significa que os exemplos são melhor separados em 3 clusters do que em 2. Ainda que 60 exemplos (4648 - 4588) não pertencem a um cluster com probabilidade 1, desse total, 57 exemplos pertencem a um dos clusters com probabilidade 0,999. Como mencionado, para *AutoClass*

Tabela 6.25: EPH: resultados - 3 clusters

Cluster	P	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_3(0)$	0	4648	45,7	54,4	0,2	$0,2 \pm 0,1$	8	$7,4 \pm 0,2$
$C_3(1)$			31,8					
$C_3(2)$			22,5					
$C_3(0)$	1	4588	45,5	54,5	0,4	$0,3 \pm 0,1$	5	$5,1 \pm 0,1$
$C_3(1)$			32,2					
$C_3(1)$			22,3					

essa probabilidade é considerada como próxima de 1 e, portanto, para esses exemplos não são consideradas probabilidades de 0,001 dos exemplos pertencerem a um outro cluster. Isso justifica a existência de 4588 exemplos com probabilidade igual a 1 e somente 3 exemplos pertencerem a mais de um cluster, como pode ser observado na Tabela 6.26.

Tabela 6.26: EPH: clusters sobrepostos - 3 clusters

Clusters Sobrepostos	# Ex Comuns
$C_3(0, 1)$	1
$C_3(1, 2)$	2

De fato, somente um exemplo pertence ao cluster 0 e 1 simultaneamente — $C_3(0, 1)$. Ou seja, com uma dessas probabilidades $< 0,9999$, e dois exemplos pertencem aos clusters 1 e 2 simultaneamente — $C_3(1, 2)$. Também foi observado que o número de regras geradas por **See5** é pequeno, com um erro verdadeiro muito baixo. As regras induzidas por **See5** são encontradas em (Martins, Monard, Haedo, & Matsudo, 2001). Na Tabela 6.27 são apresentados os atributos presentes nas regras induzidas pelo **See5** para descrever as três classes (clusters).

Tabela 6.27: EPH: atributos no conjunto de regras - 3 clusters

	Atributos				
	01	02	03	04	05
$C_3(0)$	● ○	● ○	●		
$C_3(1)$	● ○	●	● ○	●	● ○
$C_3(2)$	● ○	● ○			

Os resultados dos experimentos mostraram que o poder preditivo para 2 e 3 clusters foi muito bom, apresentando um número pequeno de regras induzidas e erros associados às regras muito baixo. Isto significa que, para 2 e 3 clusters, **AutoClass** conseguiu separar bem os exemplos nos clusters e de forma tal que **See5** consegue explicar esses clusters com um modelo simples e preciso, para esse conjunto de dados.

6.2.2.3 Fixando o Número de Clusters em 5

Os resultados obtidos com o número de clusters fixado em 5 são mostrados nas Tabelas 6.28, 6.29 e 6.30. Pode ser observado que o erro verdadeiro do modelo induzido por *See5* para 5 clusters continua baixo, mas o número de regras para exemplos com probabilidade ≥ 0 incrementa consideravelmente, o que dificulta a interpretação “semântica” desses clusters pelo especialista.

Tabela 6.28: EPH: resultados - 5 clusters

Cluster	P	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_5(0)$	0	4648	35,2	64,8	2,3	$4,3 \pm 0,4$	36	$37,3 \pm 1,2$
$C_5(1)$			24,2					
$C_5(2)$			20,3					
$C_5(3)$			10,5					
$C_5(4)$			9,9					
$C_5(0)$	1	3007	24,6	65,0	0,1	$0,6 \pm 0,1$	17	$15,4 \pm 0,6$
$C_5(1)$			35,0					
$C_5(2)$			13,6					
$C_5(3)$			12,7					
$C_5(4)$			14,1					

Tabela 6.29: EPH: clusters sobrepostos - 5 clusters

Clusters Sobrepostos	# Ex Comuns
$C_5(0,1)$	168
$C_5(0,1,2)$	88
$C_5(0,2)$	282
$C_5(1,2)$	21
$C_5(1,4)$	2
$C_5(2,3)$	215
$C_5(2,3,4)$	2
$C_5(2,4)$	11
$C_5(3,4)$	10

Vale ressaltar que a contagem de exemplos comuns foi realizada considerando separadamente os clusters. Por exemplo, a sobreposição dos clusters 0 e 1 ($C_5(0,1)$) indica que há 168 exemplos que somente pertencem ao cluster 0 e 1, enquanto que $C_5(0,1,2)$ indica que há 88 exemplos que pertencem simultaneamente a esses três clusters. Ou seja, esses 88 exemplos não foram incluídos nos 168 exemplos anteriores.

Considerando os clusters com os maiores números de exemplos em comum — $C_5(0,1)$, $C_5(0,2)$ e $C_5(2,3)$ — os quais representam uma proporção aproximada⁵ de 6,1% para

⁵Por exemplo, a proporção aproximada para $C_5(0,1)$ é obtida considerando que os clusters 0 e 1 contêm 59,4% dos exemplos — 35,2% para $C_5(0)$ + 24,2% para $C_5(1)$. Estes 59,4% representam aproximadamente 2760 exemplos dos 4648 exemplos do conjunto de treinamento. Assim, 168 exemplos sobrepostos representam, proporcionalmente, 6,1% dos exemplos dos clusters 0 e 1.

Tabela 6.30: EPH: atributos no conjunto de regras - 5 clusters

	Atributos																				
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21
$C_5(0)$	• ○	• ○	• ○		• ○	• ○	•	•	•	•	• ○	•	•							○	
$C_5(1)$	• ○	• ○	• ○		•	•			•		○	•		•	○					○	○
$C_5(2)$	• ○	• ○	• ○		•	• ○	•	•	•		• ○	•	•		• ○	•	•	•	•	•	○
$C_5(3)$	• ○	• ○	•								•				•	•	•	•	•	○	
$C_5(4)$		• ○	• ○																• ○		

$C_5(0, 1)$, 10,9% para $C_5(0, 2)$ e 15,1% para $C_5(2, 3)$, uma tentativa válida, sugerida pelo especialista, é agrupar esses clusters em um único cluster. A sobreposição dos clusters com a maior proporção de exemplos comuns, que nesse caso é o cluster 2 e 3, foram agrupados em um único cluster, denotado por $C_5(2-3)$.

Após renomear os clusters 2 e 3 como um único cluster $C_5(2-3)$, o conjunto de dados foi fornecido ao **See5**, agora com quatro clusters, obtendo os resultados mostrados nas Tabelas 6.31 e 6.32. Na Tabela 6.31 é mostrado apenas o resultado obtido para o cluster $C_5(2-3)$, considerando que os resultados relacionados aos clusters $C_5(0)$, $C_5(1)$ e $C_5(4)$ não foram alterados, — Tabela 6.28. Ainda, observe que no título das tabelas consta o rótulo ‘5 clusters’ para indicar que os resultados estão relacionados aos experimentos em que o número de clusters foi fixado em 5.

Tabela 6.31: EPH: resultados para $C_5(2-3)$ - 5 clusters

Cluster	P	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_5(2-3)$	0	4648	30,7	64,8	2,3	$3,6 \pm 0,4$	21	$26,6 \pm 1,8$

Pode ser observado que agrupando os clusters 2 e 3, **See5** utiliza um número menor de atributos para descrever as regras do modelo, como mostrado na Tabela 6.32. No caso, os atributos 15, 16, 18 e 21, não estão presentes nessas regras.

6.2.2.4 Fixando o Número de Clusters em 10

Os resultados obtidos com o número de clusters fixado em 10 são mostrados nas Tabelas 6.33, 6.34 e 6.35. O erro verdadeiro obtido do classificador induzido por **See5**, apesar de ter aumentado, ainda pode ser considerado baixo, como pode ser observado na Tabela 6.33. Porém, o número de regras aumentou significativamente.

Tabela 6.32: EPH: atributos no conjunto de regras para $C_5(2)$ e $C_5(3)$ agrupados - 5 clusters

	Atributos																				
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21
$C_5(0)$	•	•	•			•	•	•	•	•	•	•	•								
$C_5(1)$	•	•	•			•			•			•		•							
$C_5(2-3)$	•	•	•		•	•	•	•	•		•	•					•			•	
$C_5(4)$	•		•																•		

Tabela 6.33: EPH: resultados - 10 clusters

Cluster	P	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_{10}(0)$	0	4648	22,4	77,7	3,6	$7,1 \pm 0,5$	96	$90,5 \pm 2,5$
$C_{10}(1)$			12,8					
$C_{10}(2)$			9,9					
$C_{10}(3)$			9,9					
$C_{10}(4)$			9,3					
$C_{10}(5)$			8,4					
$C_{10}(6)$			8,1					
$C_{10}(7)$			7,0					
$C_{10}(8)$			6,3					
$C_{10}(9)$			5,8					
$C_{10}(0)$	1	2712	18,6	81,4	1,1	$2,7 \pm 0,5$	43	$41,2 \pm 0,8$
$C_{10}(1)$			15,3					
$C_{10}(2)$			9,6					
$C_{10}(3)$			8,5					
$C_{10}(4)$			11,0					
$C_{10}(5)$			7,6					
$C_{10}(6)$			13,2					
$C_{10}(7)$			6,9					
$C_{10}(8)$			1,1					
$C_{10}(9)$			8,2					

Nota-se, também, que a sobreposição dos clusters também aumentou — Tabela 6.34 — bem como o número de atributos utilizados por *See5* na descrição das regras, Tabela 6.35.

Tabela 6.34: EPH: Clusters sobrepostos - 10 clusters

Clusters Sobrepostos	# Ex Comuns	Clusters Sobrepostos	# Ex Comuns
$C_{10}(0, 2)$	202	$C_{10}(3, 9)$	5
$C_{10}(0, 7)$	121	$C_{10}(4, 6)$	3
$C_{10}(0, 7, 8)$	23	$C_{10}(4, 6, 9)$	1
$C_{10}(0, 8)$	99	$C_{10}(4, 9)$	9
$C_{10}(1, 2)$	122	$C_{10}(5, 7)$	105
$C_{10}(1, 7)$	12	$C_{10}(5, 7, 8)$	3
$C_{10}(0, 2, 8)$	11	$C_{10}(5, 8)$	66
$C_{10}(1, 2, 7)$	1	$C_{10}(5, 9)$	11
$C_{10}(2, 9)$	19	$C_{10}(6, 9)$	10
$C_{10}(3, 4)$	114	$C_{10}(7, 8)$	24
$C_{10}(3, 4, 9)$	4	$C_{10}(7, 8, 9)$	1

Tabela 6.35: EPH: atributos no conjunto de regras - 10 Clusters

	Atributos																													
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$C_{10}(0)$	•	•	•			○			•		•		•			•	•					•	•	•	•	•				
$C_{10}(1)$	•	•	•			•	•				•	•		•	•	•	•	•				•				•	•			
$C_{10}(2)$	•	•	•		○	○			•		•	•	•	•	•	•	•			•	•	•			•	•				
$C_{10}(3)$	•	•	•						•	•			•					•	•											
$C_{10}(4)$	•	•	•						•		•	•			•			•	•			•						•		
$C_{10}(5)$	•	•	•	•		•			•	•	•		•		•	•			•		•	•					•			
$C_{10}(6)$	•	•							•						•				○											
$C_{10}(7)$	•	•	•	•	•	•			○		•				•	•	•			•	•		•				•		•	
$C_{10}(8)$	•	•	•			•	•		•		•			•		•	•			•							•			
$C_{10}(9)$	•	•	•		○	○			•			•	•		•	•	•	•	•	•	•	•				•			•	•

De forma análoga ao experimento realizado com 5 clusters, foi considerada para 10 clusters a maior proporção aproximada dos exemplos sobrepostos (14,6% para $C_{10}(5, 7)$, 13,5% para $C_{10}(0, 2)$ e 12,7% para $C_{10}(3, 4)$) e foi realizado novo experimento cujos resultados são mostrados nas Tabelas 6.36 e 6.37. Esses clusters com a maior proporção de exemplos sobrepostos foram agrupado como $C_{10}(0-2)$, $C_{10}(3-4)$, $C_{10}(5-7)$, e considerados como classes. Com esses novos agrupamentos usados como classes, foram realizados as seguintes rotulações no conjunto de dados, os quais foram posteriormente submetidos a See5:

1. a base de dados foi rotulada considerando somente os clusters 0 e 2 como um único cluster (classe) denotado por $C_{10}(0-2)$ — total 9 clusters;
2. idem ao anterior mas considerando somente os clusters 3 e 4 como um único cluster denotado por $C_{10}(3-4)$ — total 9 clusters;
3. idem ao anterior mas considerando somente os clusters 5 e 7 como um único cluster denotado por $C_{10}(5-7)$ — total 9 clusters;
4. idem aos anteriores mas agora considerando os três clusters, $C_{10}(0-2)$, $C_{10}(3-4)$ e $C_{10}(5-7)$ — total 7 clusters;

Na Tabela 6.36 encontram-se os resultados dos quatro experimentos realizados: um experimento para cada novo agrupamento de classe e outro considerando todos os agrupamentos de classes em uma única classe — $C_{10}(0-2), C_{10}(3-4), C_{10}(5-7)$.

Tabela 6.36: EPH: resultado do agrupando em 4 conjuntos de clusters - 10 clusters

Cluster	P	# ExCl	% ExCl	ErrCM %	ErrA %	Err10cv %	# R	# MR
$C_{10}(0-2)$	0	4648	33,2	49,8%	3,2	$6,0 \pm 0,3$	91	$78,5 \pm 3,2$
$C_{10}(3-4)$			19,2		3,2	$6,4 \pm 0,3$	77	$69,8 \pm 1,6$
$C_{10}(5-7)$			15,5		3,3	$6,1 \pm 0,1$	94	$82,6 \pm 2,5$
$C_{10}(0-2),(3-4),(5-7)$			50,2		2,6	$4,9 \pm 0,4$	59	$55,0 \pm 1,8$

Os resultados obtidos mostram que no último experimento realizado, o erro verdadeiro e o número de regras diminuíram consideravelmente. Na Tabela 6.37 são mostrados os atributos que estão presentes nas regras induzidas nesse quarto experimento. Percebe-se que as novas regras induzidas utilizam três novos atributos 31, 32 e 33 que não foram usados anteriormente por *See5* — Tabela 6.35 na página anterior — enquanto que oito atributos utilizados anteriormente, especificamente os atributos 04, 05, 09, 23, 24, 25, 28 e 30, não foram necessários para descrever as regras do novo modelo.

Tabela 6.37: EPH: atributos com $C_{10}(0)$ e $C_{10}(2)$, $C_{10}(3)$ e $C_{10}(4)$, $C_{10}(5)$ e $C_{10}(7)$ agrupados - 10 clusters

	Atributos																																			
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33			
$C_{10}(0-2)$	•	•	•			•				•	•	•		•	•	•	•			•		•					•									
$C_{10}(1)$	•	•	•			•	•					•		•	•		•	•				•					•	•								
$C_{10}(3-4)$	•																																			
$C_{10}(5-7)$	•	•	•			•					•			•	•	•	•			•	•							•				•	•	•		
$C_{10}(6)$	•	•													•																					
$C_{10}(8)$	•	•	•								•				•	•	•			•								•							•	
$C_{10}(9)$	•	•	•			•						•	•		•	•	•		•	•		•					•			•			•			

Como para 5 e 10 clusters a quantidade de clusters sobrepostos aumentou significativamente e, atendendo a solicitação do especialista, novos experimentos foram realizados agrupando os clusters sobrepostos com a maior proporção de exemplos em comum, em um único cluster (classe). O conjunto de dados foi novamente rotulado com o nome dos clusters agrupados, e o processo foi repetido. Os resultados obtidos foram considerados melhores do que os anteriores, pois diminuiu o número de regras, o número de atributos utilizados nas regras, e o valor do erro dos classificadores. Maiores detalhes dos resultados obtidos com esse conjunto de dados, podem ser encontrados em (Martins, Monard, Haedo, & Matsudo, 2001).

Com esses resultados foi possível para o especialista do domínio realizar uma análise semântica, para dar uma “explicação” aos clusters encontrados. É importante ressaltar que podem ser obtidas várias informações em cada etapa do processo, usando as informações geradas, tais como apresentadas nas tabelas descritas anteriormente. Na realidade, os diversos experimentos realizados com um conjunto de dados, tem como objetivo fornecer ao especialista visões parciais diferentes desses dados, com o intuito de auxiliá-lo na re-

alização de uma interpretação mais apurada desses dados. Entretanto, a análise manual dos resultados contidos nesses relatórios, a qual deve ser realizada pelo especialista com o objetivo de “descobrir” o significado dos clusters, nem sempre é uma tarefa fácil. Assim, é importante o desenvolvimento de ferramentas computacionais para auxiliar o especialista nessa tarefa ([Baranauskas & Monard, 2000a](#)).

A metodologia proposta para realização de experimentos, usando tanto algoritmos de aprendizado supervisionado quanto não supervisionado, em um conjunto de dados não rotulados mostrou-se bastante interessante, pois oferece ao especialista uma vasta gama de informações relacionadas aos dados. Apesar dos clusters encontrados no conjunto de dados Breast-cancer não corresponderem às classes verdadeiras, isso significa que os *bias* dos algoritmos utilizados, **AutoClass** e **See5**, podem não ser os mais adequados quando combinados para esse conjunto de dados. Mesmo que o resultado obtido seja diferente do resultado que correspondem às classes, nem sempre pode ser considerado um resultado ruim. Por exemplo, para o caso do **AutoClass** que conseguiu separar bem os dados em clusters, todos com probabilidade 1, significa que existe um outro tipo de relacionamento entre os dados. Assim, os dados podem ser agrupados utilizando um outro critério, o qual pode ser útil para encontrar padrões diferentes dos esperados. Portanto, cabe ao especialista analisar e decidir se os algoritmos são adequados àquele conjunto de dados e se os padrões encontrados são válidos, mesmo que diferentes dos esperados.

No caso dos experimentos com o conjunto de dados EPH, além da classe não ser conhecida, os atributos não eram informativos, devido à codificação dos mesmos. Assim, como em qualquer processo de agrupamento, a análise dos resultados é uma tarefa extremamente complexa, visto que sem a ajuda do especialista do domínio não se tem um *feedback* “conceitual” para os clusters encontrados. No entanto, para o conjunto de dados EPH, os resultados obtidos nos experimentos foram considerados bons, pois os especialistas conseguiram observar, analisando os resultados dos diversos agrupamentos dos dados, informações úteis e ter uma melhor compreensão dos dados, observando as principais características relacionadas aos atributos utilizados, à sobreposição dos clusters, às regras induzidas, entre outros. Dessa forma, consideramos que a metodologia atingiu o objetivo proposto e pode ser uma grande aliada quando o interesse em um processo de agrupamento está relacionado à interpretação semântica de clusters de um conjunto de dados não rotulados.

6.3 Considerações Finais

Em um processo de mineração de dados, a interação com o especialista é fundamental, visto que este detém o conhecimento do domínio. No entanto, é possível automatizar alguns processos que possam auxiliar o especialista, ou usuário final, no próprio processo de mineração ou mesmo na interpretação dos resultados obtidos pelos algoritmos de aprendizado. Um dos objetivos deste trabalho consiste na proposta e implementação de uma metodologia para auxiliar o especialista no processo de descobrir conhecimento em um conjunto de dados não rotulados, usando algoritmos de aprendizado supervisionado e não supervisionado.

A metodologia proposta consiste em auxiliar o especialista a ir além de simplesmente encontrar os clusters presentes no conjunto de dados não rotulados, mas, também, em analisar sobreposição de clusters, atributos presentes nas regras induzidas, número de regras, erro associado ao classificador induzido, entre outros. Essas informações são úteis pois, conceitualmente, descrever clusters não significa necessariamente, descrever conceitos representados pelos exemplos que pertencem a esses clusters, como mostrado na Figura 4.2 na página 56 e descrito na Seção 4.1.1.

É evidente que o especialista sabe que um cluster, que contém um dado subconjunto de exemplos do conjunto original, não auxilia necessariamente na tarefa de descobrir o possível conceito embutido nesses clusters. Entretanto, o uso de algoritmos de AM que induzem regras as quais são facilmente interpretáveis por humanos utilizando o conjunto de clusters encontrado por algum algoritmo de AM não supervisionado, clusters esses que podem ser reagrupados por iniciativa do especialista utilizando a metodologia proposta, permite analisar melhor as características dos exemplos que pertencem a cada cluster, facilitando assim ao especialista encontrar uma explicação semântica, ou conceito embutido, desses clusters.

Experimentos em Mineração de Textos Usando PRETEXT

A abordagem *bag-of-words* é uma técnica bastante utilizada na representação de documentos em um processo de MT. No entanto, essa técnica é caracterizada pela alta dimensionalidade no conjunto dos possíveis atributos, como visto no Capítulo 3. Neste capítulo é apresentado uma série de experimentos usando quatro conjuntos distintos de textos, rotulados e não rotulados, cujo objetivo é induzir o melhor classificador, ou agrupamento, diminuindo a dimensionalidade dos atributos. Em outras palavras, a idéia é induzir o classificador que possui o menor erro de predição futura utilizando um subconjunto de cardinalidade bem menor que a cardinalidade do conjunto dos possíveis atributos que representam os documentos. Os textos utilizados são de diversas fontes, tais como páginas *web* e jornais. Os experimentos foram realizados usando os algoritmos See5, C4.5rules, CN2, Rainbow, SVM-Torch II e AutoClass, para o aprendizado supervisionado e não supervisionado. Para cada conjunto de textos são descritos os experimentos realizados e os resultados obtidos.

7.1 O Conjunto de Textos NILC

O conjunto de textos, ou documentos, usados nesse experimento foi fornecido pelo Núcleo Interinstitucional de Lingüística Computacional — NILC¹. Esse conjunto de textos contém mais de 4000 documentos escritos em português, divididos nos seguintes tópicos: didáticos, jornalísticos, jurídicos, literários e técnicos/científicos. Desses 4000 documentos, foram selecionados 248 documentos do tópico jornalístico, os quais estão classificados em

¹<http://www.nilc.icmc.usp.br/nilc/>.

quatro classes: Informática, Economia, Esporte e Política. Cada documento é um arquivo texto (extensão `txt`) com tamanho médio e desvio padrão de $24,84 \text{ KB} \pm 13,76 \text{ KB}$. Na Tabela 7.1 é mostrada a distribuição dos documentos por classe.

Tabela 7.1: NILC: distribuição de documentos por classe

Economia	Esporte	Informática	Política	Total
63	59	66	60	248
25,4%	23,8%	26,6%	24,2%	100%

7.1.1 O Pré-processamento

O pré-processamento dos documentos foi realizado usando a ferramenta PRETEXT. Os documentos foram transformados em 22214 *stems* sendo que mais de 9000 *stems* apareceram apenas uma vez e mais de 2900 apareceram mais de duas. De acordo com Luhn, *stems* com uma frequência muito baixa podem não ser significativos na discriminação de documentos. Assim, neste trabalho foi definido um limiar, ou corte inferior de Luhn, para que apenas os *stems* acima desse limiar de frequência fossem mantidos. Na realidade, esse é um limiar mínimo que está relacionado com a eliminação simples e direta de *stems* com frequência 1 e 2. Um outro critério adotado foi definir como limiar inferior um valor relacionado com a quantidade de exemplos na classe minoritária. Esse limiar foi definido em 10% do número de exemplos da classe minoritária.

Para esse conjunto de textos, o limiar ou corte inferior definido é o valor correspondente a 10% da classe minoritária Esporte (59 exemplos), *i.e.* *stems* com frequência acima de 6 para esse conjunto de textos. Nesse caso, de um total de 22214 *stems* permaneceram apenas 6202, cuja frequência média e desvio padrão é $392,4 \pm 415,1$. Na Figura 7.1 é ilustrada a saída gráfica usando os arquivos gerados por PRETEXT e Gnuplot (Crawford, 1998), da curva de Zipf para *stems* com frequência maior ou igual a 6 para esse conjunto de textos.

Pode ser observado que apenas um *stem* aparece 3217 vezes de um conjunto de *stems* com frequência maior ou igual a 6. No entanto, o objetivo é encontrar o menor número de atributos que possam discriminar bem um conjunto de documentos para obter um bom classificador. Assim, após a primeira tentativa com o limiar inferior estabelecido, vários outros pontos de corte podem ser utilizados buscando esse objetivo.

Dessa forma, os experimentos usando esse conjunto de textos ilustram alguns passos usados para reduzir a dimensão do conjunto de atributos, usando a ferramenta PRETEXT, com o objetivo de induzir um bom classificador, usando os algoritmos de aprendizado

Figura 7.1: NILC: frequência dos *stems*

See5 ([Rulequest-Research, 1999](#)), CN2 ([Clark & Boswell, 1989](#)) e SVMTorch II ([Collobert & Bengio, 2001](#)). *See5* e CN2 são algoritmos de aprendizado simbólicos que induzem regras de decisão. O *bias* indutivo do CN2 é diferente do *See5* ([Baranauskas & Monard, 2000c](#)).

Support Vector Machines (SVM) são técnicas de aprendizado baseadas na Teoria de Aprendizado Estatístico, proposta por [Vapnik & Chervonenkis \(1971\)](#). Essa técnica mapeia os dados de entrada para um espaço abstrato de alta dimensão, onde os exemplos podem ser eficientemente separados por um hiperplano. O SVM incorpora esse conceito usando funções denominadas *Kernels*. Essas funções permitem o acesso a espaços complexos de maneira simplificada e computacionalmente eficiente. O hiperplano ótimo nesse espaço é definido como aquele que maximiza a margem de separação entre dados pertencentes a diferentes classes. A principal vantagem do SVM é sua precisão e robustez em dados com uma alta dimensão. Entretanto, diferentemente de algoritmos de aprendizado simbólico, classificadores induzidos utilizando SVM não são diretamente interpretáveis pelo usuário.

7.1.2 Resultados Obtidos

Após o pré-processamento, os documentos transformados por PRETEXT na tabela atributo-valor, no formato padrão dos arquivos *.data* e *.names* do DISCOVER, foram submetidos aos algoritmos de aprendizado para realizar os experimentos. Nessa série de experimentos, foi possível observar o erro obtido pelos classificadores induzidos e, caso

necessário, os cortes de Luhn foram ajustados e os algoritmos executados novamente.

Os resultados obtidos pelos dois algoritmos de aprendizado simbólicos são apresentados na Tabela 7.2, na qual:

- ‘Exp’ - identifica o experimento;
- ‘Inf’ e ‘Sup’ - representam, respectivamente, os valores inferiores e superiores utilizados como cortes de Luhn;
- ‘stems’ - número de *stems* (atributos) na tabela atributo-valor;
- ‘Med’ - identifica a medida usada no experimento, *tf* ou *tfidf*;
- ‘ErrA %’ - porcentagem do erro do classificador induzido quando utiliza todo o conjunto de dados como treinamento e teste;
- ‘Err10cv %’ - porcentagem do erro do classificador induzido usando *10-fold cross-validation* e o erro padrão;
- ‘# R’ - número de regras no conjunto de regras induzido pelos algoritmos;
- ‘# AtrR’ - número de atributos diferentes presentes no conjunto de regras;
- ‘ F_1 ’ - representa o valor calculado da medida F_1 ;
- ‘% Esp’ - porcentagem de valores não nulos na tabela atributo-valor.

O símbolo $*$ nos experimentos E_{3*} e E_{5*} na Tabela 7.2 indica que o número de atributos foi reduzido não apenas usando os cortes de Luhn mas, também, pelo uso da informação contida em alguns dos arquivos criados pelo módulo **Stem.pl** da ferramenta PRETEXT (Seção 5.3.2 na página 83). Com essas informações, é possível decidir se um *stem* representa um conjunto de palavras relevantes ou se algumas palavras com *stems* idênticos são pouco significativas para discriminar os documentos. Como mencionado, palavras que não são consideradas relevantes pelo usuário podem ser incluídas em uma lista de *stopwords* específica definida pelo usuário e PRETEXT pode ser novamente executado, reduzindo a dimensão dos dados ou a frequência de *stems*.

No primeiro experimento E_1 , a tabela atributo-valor com os *stems* com frequência maior ou igual a 6 (corte inferior de Luhn) foram submetidos aos algoritmos **See5** e **CN2**. Os erros obtidos por ambos classificadores podem ser considerados similares exceto pela quantidade de regras induzidas. Os erros, quando comparados com o erro da classe majoritária, podem ser considerados bons. No experimento E_2 a idéia é encontrar um valor

Tabela 7.2: NILC: resultados com *See5* e CN2

Exp	Inf	Sup	stems	Med	ErrA %	Err10cv %		# R		# Attr		F_1	% Esp
						See5	CN2	See5	CN2	See5	CN2		
E_1	6		6202	<i>tf</i>	0,8	$8,8 \pm 2,1$	$8,5 \pm 1,8$	6	13	7	21	0,91	11,7
				<i>tfidf_n</i>	1,2	$8,8 \pm 1,5$	$8,5 \pm 1,9$	6	13	7	21	0,91	11,6
E_2	6	808	6142	<i>tf</i>	1,6	$8,4 \pm 2,7$	$7,7 \pm 1,9$	6	14	7	20	0,92	11,1
				<i>tfidf_n</i>	1,6	$6,9 \pm 2,0$	$7,2 \pm 2,4$	6	14	7	20	0,91	11,1
E_{3*}	59		1538	<i>tf</i>	0,4	$9,3 \pm 2,2$	$10,1 \pm 1,9$	7	12	8	23	0,91	31,0
				<i>tfidf_n</i>	0,4	$10,1 \pm 2,1$	$10,9 \pm 1,8$	7	12	8	23	0,90	30,9
E_4	30		2468	<i>tf</i>	0,8	$7,7 \pm 1,7$	$12,1 \pm 2,1$	6	13	7	21	0,92	23,4
				<i>tfidf_n</i>	1,2	$7,7 \pm 1,0$	$10,9 \pm 1,6$	6	13	7	21	0,92	23,3
E_{5*}	30	808	2412	<i>tf</i>	2,4	$7,6 \pm 1,5$	$8,5 \pm 2,3$	6	13	6	22	0,92	22,3
				<i>tfidf_n</i>	2,4	$8,4 \pm 1,8$	$7,6 \pm 1,5$	6	13	6	22	0,92	22,3
E_6	12		4266	<i>tf</i>	0,8	$8,4 \pm 1,7$	$10,5 \pm 1,9$	6	13	7	21	0,92	15,8
				<i>tfidf_n</i>	1,2	$8,5 \pm 2,0$	$10,5 \pm 2,3$	6	13	7	21	0,92	15,7
E_7	24		2874	<i>tf</i>	0,8	$6,9 \pm 1,1$	$9,3 \pm 1,8$	6	13	7	21	0,93	21,1
				<i>tfidf_n</i>	1,2	$8,4 \pm 1,5$	$7,3 \pm 0,8$	6	13	7	21	0,91	21,1
E_8	48		1796	<i>tf</i>	0,4	$7,7 \pm 1,5$	$12,9 \pm 2,1$	7	12	8	23	0,92	28,5
				<i>tfidf_n</i>	0,4	$8,5 \pm 1,6$	$9,7 \pm 2,3$	7	12	8	23	0,91	28,4
E_9	6	808	19	<i>tf</i>	2,0	$2,8 \pm 1,2$	$4,5 \pm 1,3$	7	12	6	15	0,97	13,5
				<i>tfidf_n</i>	2,0	$4,0 \pm 1,2$	$3,6 \pm 1,1$	7	12	6	15	0,96	13,5

apropriado para o corte superior de Luhn. O valor 808 foi escolhido porque está relacionado à média e um desvio padrão da frequência dos *stems* ($392,4 \pm 415,1$). Nesse caso, apenas atributos com frequência menor que 6 e maior que 808 não foram considerados. Os resultados obtidos mostram uma melhora especialmente para o classificador induzido pelo *See5* usando a medida *tfidf_n*.

Para tentar usar um valor de atributo que possa discriminar perfeitamente uma classe, caso exista, no experimento E_{3*} os *stems* usados possuem frequência igual ou maior que 59. Este valor representa o número de exemplos que pertencem à classe minoritária — classe Esporte, Tabela 7.1 na página 126. Essa heurística é baseada na idéia que um atributo poderia aparecer apenas uma vez em todos os documentos de uma mesma classe, discriminando perfeitamente aquela classe. Assim, um valor seguro é dado pelo número de documentos na classe minoritária. Entretanto, como pode ser observado, o erro dos classificadores, nesse caso, aumentou.

Para continuar esse processo de tentativa e erro, com o objetivo de encontrar um bom valor para o corte inferior de Luhn, os experimentos E_4 e E_{5*} , E_6 , E_7 e E_8 foram executados ajustando o valor para aproximadamente 50%, 20%, 40% e 80% do número de exemplos na classe minoritária, respectivamente. No experimento E_{5*} , foi fixado o corte superior de Luhn em 808, bem como uma nova lista de *stopwords* foi definida para reduzir a dimensão dos dados.

Considerando somente os resultados até o experimento E_8 inclusive, pode ser obser-

Figura 7.2: Erro dos classificadores

vado que os experimentos E_2 e E_4 apresentam os melhores resultados para CN2 e **See5**, respectivamente, considerando as duas medidas tf e $tfidf$. Assim, decidiu-se executar um experimento usando apenas atributos que aparecem no conjunto de regras induzido pelos classificadores nos experimentos E_2 e E_4 . Mais especificamente, os atributos presentes nas regras induzidas por esses classificadores foram selecionados para construir um novo conjunto de dados e submetidos aos algoritmos de aprendizado. O melhor resultado foi obtido nesse novo experimento, o experimento E_9 . Como pode ser observado, um conjunto de dados textuais com 22214 atributos foi reduzido, usando apenas atributos que aparecem no conjunto de regras induzido por **See5** e CN2, para um conjunto de dados com apenas 19 atributos, no qual o erro dos classificadores induzidos é baixo. Esse pequeno número de atributos representa 0,09% do número original de atributos. Considerando o erro da classe majoritária, a precisão desse último experimento é muito boa para esse conjunto de dados e algoritmos de aprendizado utilizados.

Os erros obtidos pelos classificadores induzidos por **See5** e CN2 são ilustrados na Figura 7.1.2. É interessante verificar que o erro praticamente permanece constante para os classificadores obtidos por **See5**, com exceção do experimento E_9 , o qual consideramos uma situação especial. Assim, é possível obter uma boa representação para esse conjunto de textos com apenas 2874 *stems*, de um total de 22214 *stems* iniciais.

A fim de ilustrar o desempenho de um algoritmo de SVM nesse conjunto de dados, usando diferentes números de atributos, as tabelas atributo-valor dos experimentos E_1 , E_{5*} e E_9 , foram selecionadas e submetidas ao algoritmo SVM Torch II usando a função linear

de *Kernel* e parâmetros *default*. Os resultados obtidos são apresentados na Tabela 7.3.

Tabela 7.3: NILC: resultados com SVMTorch II

Exp	Inf	Sup	stems	Med	Err10cv %
E_1	6		6202	tf $tfidf_n$	$0,4 \pm 1,3$ $0,4 \pm 1,3$
E_{5*}	30	808	2412	tf $tfidf_n$	$0,8 \pm 1,7$ $0,4 \pm 1,3$
E_9	6	808	19	tf $tfidf_n$	$0,0 \pm 0,0$ $0,0 \pm 0,0$

Como pode ser observado, SVMTorch II conseguiu um excelente desempenho para esse conjunto de dados. Apesar de, geralmente, SVM obter um bom desempenho utilizando dados de alta dimensionalidade diminuindo a necessidade de seleção de atributos (Joachims, 2002), os resultados obtidos usando o conjunto de dados de dimensão 19 criado pelo PRETEXT, mostram que a seleção de atributos pode melhorar muito a precisão de SVMTorch II.

É importante enfatizar que neste trabalho o algoritmo de SVM foi utilizado apenas para verificar o desempenho do algoritmo para o conjunto de dados específico, visto que a saída de SVM é um classificador caixa preta². Um dos focos deste trabalho é a indução de um modelo (classificador) simbólico, embora muitas vezes com uma precisão menor, mas que permita interpretar, analisar e usar o conhecimento de diferentes formas, tais como organizar uma pequena função de índice para recuperar mais eficientemente documentos classificados. Mais detalhes dos resultados obtidos nesses experimentos encontram-se em (Martins, Monard, & Matsubara, 2003b,c).

7.2 O Conjunto de Textos 7SECTORS

O conjunto de textos denominado 7SECTORS, consiste de um subconjunto de 4582 páginas *web*, em inglês, de empresas classificadas em uma hierarquia de setores industriais, disponibilizada por *Market Guide Inc.*³. O tamanho total do conjunto de documentos é de 37 MB em formato html, os quais estão classificados em dois níveis. No primeiro nível, classificação usada neste trabalho, os documentos estão classificados em 7 classes diferentes.

Como os documentos consistem de páginas *web*, foi necessário converter as páginas html para arquivos textos (extensão `txt`). Com a conversão, vários documentos com

²Black box.

³<http://www.marketguide.com>.

conteúdo textual praticamente nulo foram gerados, devido ao fato desse conjunto possuir documentos que consistem somente de imagens. Assim, foram selecionados apenas os documentos com tamanho igual ou superior a 2 KB. O tamanho desse novo conjunto de textos é de aproximadamente 9 MB, o qual contém 1729 documentos. Cada documento é um arquivo com tamanho médio e desvio padrão de $5,4 \pm 6,1$ KB. A distribuição dos documentos por classe é apresentada na Tabela 7.4.

Tabela 7.4: 7SECTORS: distribuição de documentos por classe

Basic.Materials	Healthcare	Utilities	Energy	Technology	Financial	Transportation	Total
352	164	122	123	433	341	194	1729
20,4%	9,5%	7,1%	7,1%	25,0%	19,7%	11,2%	100%

7.2.1 O Pré-processamento

A conversão dos documentos no formato html para arquivos em formato txt foi realizada durante a geração dos *stems*, por meio da ferramenta PRETEXT. Inicialmente, foram encontrados 28646 *stems* em todo o conjunto de documentos. O limiar inferior adotado foi 3, assim, apenas os *stems* com frequência 1 e 2 foram descartados. Nesse caso, de um total de 28646 *stems* permaneceram apenas 10956, cuja frequência média e desvio padrão é $181,1 \pm 314$. A distribuição da frequência dos *stems* é mostrada na Figura 7.3. Como pode ser observado, apenas um *stem* apareceu com frequência acima de 10000 e outro com frequência acima de 6000.

Figura 7.3: 7SECTORS: frequência dos *stems*

Nesses experimentos foi utilizado o algoritmo C4.5rules⁴ (Quinlan, 1988), que induz regras de decisão. A fim de comparar o desempenho foi utilizado o Rainbow (McCallum, 1996), um sistema para classificação de textos em inglês que usa a técnica de aprendizado estatística Naïve Bayes. Os resultados são mostrados a seguir.

7.2.2 Resultados Obtidos

Os resultados obtidos nos experimentos realizados são apresentados na Tabela 7.5 similar à Tabela 7.2 na página 129. Nessa tabela, a coluna ‘ k -Sd’ representa o valor de k , relacionado à constante definida pelo usuário para o desvio padrão, quando é utilizada a facilidade de corte automático gerado por PRETEXT— Seção 5.3.1 na página 80.

Tabela 7.5: 7SECTORS: resultados com C4.5rules

Resultados obtidos usando a medida tf										
Exp	Inf	Sup	<i>stems</i>	k -Sd	ErrA %	Err10cv %	# R	# AtrR	F_1	% Esp
E_{1a}	3		10956		12,6	$12,2 \pm 0,8$	56	91	0,87	1,6
E_{2a}	5		7897		11,7	$11,7 \pm 1,0$	57	90	0,88	2,1
E_{3a}	13		4377		12,3	$10,6 \pm 1,1$	56	93	0,88	3,6
E_{4a}	22		3227		12,1	$11,6 \pm 0,9$	55	94	0,88	4,7
E_5	27		2880		12,3	$9,8 \pm 1,0$	55	96	0,88	5,1
E_6	30		2719		12,8	$8,2 \pm 1,2$	52	92	0,88	5,4
E_7	70		1593		13,2	$12,5 \pm 0,6$	48	93	0,87	8,3
E_8	30	3000	2709		12,8	$11,6 \pm 0,8$	52	92	0,87	5,2
E_9	135	765	825	1,0	21,1	$22,7 \pm 1,9$	39	116	0,79	8,3
E_{10}	99	957	1124	1,2	15,8	$14,6 \pm 1,0$	46	91	0,84	7,6
E_{11}	81	1061	1343	1,3	17,6	$14,1 \pm 1,3$	53	95	0,82	7,2
Resultados obtidos usando a medida $tfidf_n$										
E_{1b}	3		10956		12,6	$10,8 \pm 1,1$	55	91	0,87	1,5
E_{2b}	5		7897		11,9	$12,7 \pm 1,1$	56	90	0,88	2,0
E_{3b}	13		4377		12,3	$11,0 \pm 0,5$	56	90	0,88	3,5
E_{4b}	22		3227		12,0	$11,4 \pm 0,9$	55	94	0,88	4,6

Os experimentos E_i , $i \in \{1, 2, \dots, 11\}$, foram realizados utilizando a medida tf . Outros experimentos usando a mesma configuração dos quatro primeiros experimentos foram realizados, porém, usando a medida $tfidf_n$, cujos resultados são mostrados na segunda parte na Tabela 7.5, denotados por E_{ib} , $i \in \{1, \dots, 4\}$. Como pode ser observado nesses quatro experimentos, a medida $tfidf_n$ utilizada parece não ter, para este conjunto de documentos, influência direta na precisão do classificador induzido. Pode também ser observado que os valores de % Esp são menores para a medida $tfidf_n$, a qual, como já

⁴Nos experimentos foram utilizados os algoritmos C4.5rules ou o See5. Como já mencionado, o See5 é uma versão comercial com algumas otimizações do C4.5rules. O C4.5rules foi utilizado conjuntamente com o ambiente SNIFFER que faz parte do DISCOVER LEARNING ENVIRONMENT - DLE (Batista & Monard, 2003), para executar *10-fold cross-validation*.

mentionado, zera valores de atributos que participam de todos os documentos do conjunto de textos.

O erro dos classificadores nos diversos experimentos varia entre 8 e 12 %, o que caracteriza uma boa precisão para esse conjunto de documentos, com exceção do experimento E_9 que apresenta um erro de 22,7%. Esse erro mostra que os cortes de Luhn definidos para esse experimento não são apropriados. Nesse estudo de caso, o classificador induzido por C4.5rules teve o melhor desempenho no experimento E_6 utilizando apenas 2719 *stems*, ou seja, somente 9,5% dos 28646 atributos originais. Assim, pode-se concluir que os cortes de Luhn mais apropriados, utilizados nesses experimentos, são os definidos no experimento E_6 . Os resultados dos experimentos são mostrados graficamente na Figura 7.4.

Um ponto importante a ser observado, para este conjunto de documentos, refere-se aos altos valores de ErrA %. Normalmente, esse erro é sempre menor que o erro calculado utilizando *10-fold cross-validation*, já que ErrA % é o erro aparente, no qual o conjunto de treinamento e o conjunto de teste são iguais. O fato desse erro ser alto, mostra que o modelo (classificador) induzido é bastante instável. Também, o alto número de regras induzidas (# R) e de atributos diferentes utilizados no modelo induzido (# AtrR) mostram que não foi possível para C4.5rules realizar uma boa generalização.

Figura 7.4: 7SECTORS: erros obtidos em relação ao número de *stems* e a % dados esparsos

Para os experimentos usando o Rainbow, os resultados experimentais obtidos para a medida *tf* são apresentados na Tabela 7.6, na qual cada experimento é denotado por

$E'_i, i \in \{1 \dots 13\}$ e a coluna denotada por ‘*words*’ representa o número de palavras, e não de *stems*, usadas no experimento. Com relação aos cortes de Luhn, Rainbow permite somente realizar o corte inferior.

Tabela 7.6: 7SECTORS: resultados com Rainbow

Exp	Inf	<i>words</i>	Err10cv %
E'_1	3	15370	$11,7 \pm 2,1$
E'_2	5	11378	$10,9 \pm 1,8$
E'_3	13	6389	$10,6 \pm 3,0$
E'_4	22	4540	$15,2 \pm 2,6$
E'_5	27	3975	$13,3 \pm 3,5$
E'_6	30	3731	$13,4 \pm 2,1$
E'_7	70	1940	$18,6 \pm 3,1$
E'_8	80	1742	$19,1 \pm 3,3$
E'_9	90	1559	$21,6 \pm 2,1$
E'_{10}	100	1416	$22,6 \pm 2,6$
E'_{11}	110	1291	$22,7 \pm 3,3$
E'_{12}	120	1187	$23,7 \pm 2,9$
E'_{13}	130	935	$24,8 \pm 3,0$

Considerando a precisão dos classificadores induzidos nos experimentos realizados, os melhores resultados obtidos pelo Rainbow foram utilizando, aproximadamente, 6000 a 11000 palavras para representar cada documento — experimentos E'_2 e E'_3 na Tabela 7.6. Enquanto que uma melhor precisão foi obtida com a metodologia usando PRETEXT e C4.5rules que utilizam somente 2719 *stems* — experimento E_6 na Tabela 7.5. O corte inferior realizado nesses experimentos usando o Rainbow refere-se à frequência de palavras, enquanto que PRETEXT considera a frequência de *stems*, assim, não é possível realizar uma comparação direta dos resultados obtidos. Entretanto, como também observado pelos autores do Rainbow, o seu comportamento degrada para valores maiores relacionados ao corte inferior, o que não acontece utilizando a metodologia aqui proposta. Os resultados obtidos nesse experimento encontram-se em (Martins, Matsubara, & Monard, 2003).

7.3 O Conjunto de Textos BGMS

O conjunto de textos BGMS foi fornecido pelo grupo de pesquisadores do ISISTAN⁵. Esse conjunto contém 332 documentos escritos em inglês os quais estão classificados em quatro diferentes classes: music, goat, sheep e biomedical. O tamanho total desse conjunto de textos é 641,1 KB. Cada documento é um arquivo texto (extensão `txt`) com tamanho médio e desvio padrão de $1,9 \text{ KB} \pm 3,5 \text{ KB}$. Na Tabela 7.7 é mostrada a distribuição desses documentos por classes.

⁵<http://www.exa.unicen.edu.ar>.

Tabela 7.7: BGMS: distribuição de documentos por classe

Biomedical	Goat	Music	Sheep	Total
136	70	61	65	332
40,9%	21,1%	18,4%	19,6%	100%

7.3.1 O Pré-processamento

O conjunto de documentos foi fornecido à ferramenta PRETEXT, no primeiro passo, especificando como corte inferior o valor 6, o qual representa 10% da classe minoritária Music com 61 exemplos. Nesse caso, de um total de 11419 *stems* permaneceram apenas 1284, cuja frequência média e desvio padrão é $77,2 \pm 85,3$. A distribuição das frequências dos *stems* no conjunto de documentos é ilustrada na Figura 7.5. Como pode ser observado, apenas um *stem* possui frequência maior que 600 e apenas três *stems* com frequência maior do que 300.

Figura 7.5: BGMS: frequência dos *stems*

Apesar da classe dos exemplos ser conhecida, nesse conjunto de textos foram realizados experimentos considerando as duas abordagens utilizadas neste trabalho. A primeira é a abordagem adotada nos experimentos para categorização de textos, utilizada nos experimentos do conjunto de textos do NILC e 7SECTORS descritos neste capítulo. A segunda abordagem é a adotada para agrupamento de textos, utilizada nos experimentos do Capítulo 6, usando a metodologia descrita no Capítulo 4 na página 53. Nessa última abordagem, as classes conhecidas dos exemplos foram omitidas na realização dos experimentos. Os resultados obtidos para cada uma das abordagens são apresentados a

seguir.

7.3.2 Resultados Obtidos para Categorização de Textos

Na Tabela 7.8 são mostrados os resultados obtidos em cinco experimentos realizados, utilizando em cada experimento as medidas tf e $tfidf$. Os algoritmos de aprendizado utilizados foram o C4.5rules (identificado na tabela por C45r) e CN2.

Tabela 7.8: BGMS: resultados com C4.5rules e CN2

Exp	Inf	Sup	stems	Med	ErrA %		Err10cv %		# R		# AtrR		% Esp
					C45r	CN2	C45r	CN2	C45r	CN2	C45r	CN2	
E_1	6		1284	tf	7,2	2,7	$6,7 \pm 1,7$	$7,6 \pm 1,5$	5	33	4	54	3,2
				$tfidf$	8,1	2,7	$13,0 \pm 1,5$	$13,0 \pm 1,7$	8	41	8	69	3,2
E_2	6	163	1276	tf	20,5	3,6	$34,9 \pm 1,8$	$29,8 \pm 1,9$	15	71	18	118	3,1
				$tfidf$	20,5	3,6	$34,1 \pm 2,1$	$27,1 \pm 1,5$	15	71	18	118	3,1
E_3	61		60	tf	7,2	7,5	$12,1 \pm 1,6$	$9,0 \pm 1,5$	9	22	10	29	13,3
				$tfidf$	7,8	10,1	$12,1 \pm 1,2$	$14,1 \pm 1,7$	11	22	17	29	13,3
E_4	30		172	tf	8,1	5,4	$9,3 \pm 1,6$	$9,6 \pm 1,1$	7	24	9	42	9,1
				$tfidf$	8,1	7,5	$16,3 \pm 1,5$	$13,2 \pm 1,5$	10	28	12	48	9,1
E_5	15		446	tf	6,3	4,2	$8,1 \pm 0,9$	$8,2 \pm 1,5$	9	28	10	45	6,0
				$tfidf$	8,7	6,0	$13,9 \pm 1,2$	$11,7 \pm 1,5$	9	32	12	53	6,0

No primeiro experimento foram executados os algoritmos com todos os atributos, excluindo apenas os atributos com frequência menor que 6. Pode ser observado que o erro encontrado por C4.5rules usando a medida tf foi muito bom considerando que o erro da classe majoritária é 59,04%. O segundo experimento ilustra a busca por um valor de corte superior apropriado. O valor escolhido foi 163 que está relacionado com a média 77,2 da frequência dos *stems* mais um desvio padrão. Esse experimento não teve um bom desempenho pois, apesar de ter retirado apenas 8 atributos, pode ser observado que foram retirados atributos relevantes para discriminar as classes, incrementando muito o erro. Por exemplo, para a medida tf o erro incrementou aproximadamente 5 vezes (de 6,7% para 34,0%) para C4.5rules e quatro vezes (7,6% para 29,8%) para CN2.

No experimento E_3 foi utilizado como ponto de corte inferior o número de exemplos pertencentes a classe minoritária. Pode ser observado que esse valor não é apropriado, visto que o erro aumenta, com relação ao experimento E_1 , para ambos algoritmos. Assim, para os experimentos E_4 e E_5 definiu-se corte inferior com o valor 30 e 15, respectivamente, os quais representam, aproximadamente, 50 e 25% do valor anterior. Percebe-se que apesar do erro ter diminuído com relação ao experimento E_3 , o erro obtido no experimento E_1 para a medida tf é ainda menor.

Considerando somente o erro de classificação, pode-se concluir que os resultados do

experimento E_1 são os melhores. Com relação aos modelos induzidos, é possível observar que o mais simples é o induzido por C4.5rules, já que esse classificador consegue resultado semelhante com um conjunto de 5 regras e 4 atributos diferentes, enquanto que CN2 necessita de um conjunto de 33 regras que utilizam 54 atributos diferentes. Na Figura 7.6 são mostrados os erros dos classificadores induzidos nos experimentos E_1 , E_3 , E_4 e E_5 .

Figura 7.6: BGMS: erro dos classificadores

Para verificar o desempenho do algoritmo de SVM, nesse conjunto de dados, os dados utilizados nos experimentos E_1 e E_5 , com os quais foram obtidos os menores erros de classificação, foram submetidos ao algoritmo SVM Torch II. Os resultados obtidos são mostrados na Tabela 7.9, na qual Err10cvL representa os erros obtidos usando *10-fold cross-validation* e o *Kernel* linear e Err10cvG representa os erros obtidos usando *10-fold cross-validation* e o *Kernel* gaussiano com desvio padrão 10.

Tabela 7.9: BGMS: resultados com SVM Torch II

Exp	Min	Max	stems	Med	Err10cvL %	Err10cvG %
E_1	6		1284	<i>tf</i>	13,5 ± 6,5	17,8 ± 5,8
				<i>tfidf</i>	26,8 ± 7,0	16,3 ± 4,1
E_5	15		446	<i>tf</i>	12,9 ± 4,9	15,0 ± 4,4
				<i>tfidf</i>	23,0 ± 9,4	15,4 ± 6,5

Como pode ser observado, os erros obtidos com SVM Torch II foram maiores do que os obtidos com C4.5rules e CN2 nesses experimentos. Isso mostra mais uma vez a importância de realizar experimentos para decidir qual algoritmo de aprendizado é melhor para um determinado conjunto de dados. Como mencionado anteriormente, a técnica de SVM é considerada muito boa para dados com alta dimensionalidade. Ainda

que na maioria dos casos a precisão de SVM é superior à de algoritmos simbólicos, esse resultado não se verifica sempre, como pode ser observado nos experimentos realizados neste trabalho. Ainda, é interessante observar que na maioria dos experimentos realizados, a medida *tf* obteve melhores resultados que a medida *tfidf*. Esse é um resultado que também consideramos particular para este conjunto de dados e indutores utilizados, já que a medida *tfidf* não está diretamente correlacionada com a medida *tf*.

Novamente, é importante salientar que o uso de um indutor simbólico permite ao usuário/especialista verificar na primeira execução da ferramenta, ou posteriormente, os atributos (*stems*) utilizados pelas regras induzidas. Após, utilizando as informações fornecidas nos arquivos gerados pelo módulo **Stem.pl** da ferramenta PRETEXT, é possível verificar se esses *stems* correspondem a palavras consideradas relevantes. Caso contrário, essas palavras podem ser colocadas nas listas de *stopwords* do usuário executando novamente a ferramenta ou mesmo criando taxonomias definidas pelo usuário.

7.3.3 Resultados Obtidos para Agrupamento de Textos

Neste experimento com o conjunto de documentos BGMS, AutoClass foi executado com valores *default* para seus parâmetros, inclusive não especificando o número de clusters a serem retornados, encontrando 6 (0...5) clusters. Definido pelo especialista o corte inferior, foram considerados apenas os *stems* cuja frequência é menor ou igual a 15 e a medida *tfidf*. Na Tabela 7.10 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 distintas⁶: music, goat, sheep e biomedical.

Tabela 7.10: BGMS: resultados com AutoClass

Cluster	# ExCl		biomedical		goat		music		sheep	
0	160	(48,2%)	64	(47,1%)	44	(62,9%)	10	(16,4%)	42	(64,6%)
1	67	(20,2%)	63	(46,3%)	1	(1,4%)	0		3	(4,7%)
2	56	(16,9%)	2	(1,5%)	2	(2,9%)	51	(83,6%)	1	(1,5%)
3	36	(10,8%)	0		18	(25,7%)	0		18	(27,7%)
4	10	(3,0%)	4	(2,9%)	5	(7,1%)	0		1	(1,5%)
5	3	(0,9%)	3	(2,2%)	0		0		0	
	332	(100,0%)	136	(100,0%)	70	(100,0%)	61	(100,0%)	65	(100,0%)

Foi possível observar que, 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

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

0 contém 48,2% dos exemplos, distribuídos 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 *See5*, bem como a matriz de confusão utilizando todos os exemplos, são mostradas a seguir:

Extracted rules:

Rule 1: (cover 271)
 min <= 0
 -> class C6(0) [0.553]

Rule 2: (cover 16)
 artist > 0
 -> class C6(2) [0.778]

Rule 3: (cover 7)
 thread > 0
 -> class C6(3) [0.889]

Default class: C6(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 C6(0)
66		1				(b): class C6(1)
3		53				(c): class C6(2)
29			7			(d): class C6(3)
9		1				(e): class C6(4)
3						(f): class C6(5)

Como pode ser observado, o algoritmo induziu apenas 3 regras que “explicam” os exemplos nos clusters $C6(0)$, $C6(2)$ e $C6(3)$ encontradas por **AutoClass**, com um erro associado ao classificador bastante alto (37,0%). Pode-se verificar também que a maioria dos exemplos foram classificados no cluster $C6(0)$ (271 exemplos). O resultado obtido não é bom e isso pode ser justificado porque enquanto **AutoClass** conseguiu agrupar bem os exemplos considerando a probabilidade dos exemplos pertencerem a um mesmo cluster, o **See5**, 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. No entanto, quando foi fixado em 11 clusters, **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 7.11.

Tabela 7.11: BGMS: resultados com 11 clusters

Cluster	#	ExCl	biomedical		goat		music		sheep	
0	170	(51,2%)	67	(49,2%)	52	(74,3%)	10	(16,4%)	41	(63,1%)
1	56	(16,6%)	54	(39,7%)	0		0		2	(3,1%)
2	51	(15,42%)	2	(1,5%)	1	(1,4%)	47	(77,0%)	1	(1,5%)
3	25	(7,5%)	5	(3,7%)	4	(5,7%)	0		16	(24,6%)
4	14	(4,2%)	2	(1,5%)	9	(12,9%)	0		3	(4,6%)
5	5	(1,5%)	5	(3,7%)	0		0		0	
6	4	(1,2%)	1	(0,7%)	1	(1,4%)	0		2	(3,1%)
7	4	(1,2%)	0		0		4	(6,6%)	0	
8	3	(0,9%)	0		3	(4,3%)	0		0	
	332	(100,0%)	136	(100,0%)	70	(100,0%)	61	(100,0%)	65	(100,0%)

Nesse experimento, novamente se verifica a predominância do cluster 0 contendo 51% dos exemplos, porém distribuídos nas quatro classes distintas. Para o experimento com 11 clusters o **See5** induziu um número alto de regras. Porém, o erro geral foi reduzido de forma bastante significativo, de 37,0% para 5,7%, melhorando também o fator de confiança das regras. Ainda, foi observado que vários exemplos são cobertos por mais de uma regra, isto porque o **See5** é um algoritmo que induz regras não ordenadas ([Baranauskas & Monard, 2000b](#)). Ou seja, as regras podem se sobrepor e um exemplo pode ser coberto por mais de uma regra. As regras obtidas pelo **See5**, bem como a matriz de confusão utilizando todos os exemplos, para o experimento com 11 clusters, são mostrados em ([Martins, Godoy, Monard, Matsubara, & Amandi, 2003](#)).

Foi possível observar, também, que não considerando os clusters 5, 6, 7 e 8, os quais contêm um total de 16 exemplos, os 316 exemplos restantes nos outros clusters conseguem

ser bem “explicados” pelas regras induzidas por **See5**. Entretanto, esses clusters contêm exemplos das quatro classes *music*, *goat*, *sheep* e *biomedical*. Na realidade, o objetivo do trabalho realizado com esse conjunto de documentos, foi analisar em maiores detalhes, a proposta de um novo algoritmo de cluster hierárquico com os resultados obtidos com **AutoClass** e a metodologia aqui proposta para “explicar” esses clusters. Os resultados mostraram que **AutoClass** agrupou os exemplos considerando outras características diferentes que aquelas que permitem reconhecer essas quatro classes, determinadas pelo especialista, enquanto que o algoritmo de cluster hierárquico conseguiu agrupar razoavelmente esses exemplos em clusters que contêm textos de somente uma dessas quatro classes. Os resultados de todos os experimentos realizados, bem como o algoritmo hierárquico, estão publicados em ([Martins, Godoy, Monard, Matsubara, & Amandi, 2003](#)).

7.4 O Conjunto de Textos LNAI

O conjunto de textos denominados LNAI são resumos (*abstracts*) de artigos obtidos dos Anais do Lecture Notes in Artificial Intelligence — LNAI. Foram digitalizados 276 resumos e transformados em arquivos textos (extensão **txt**) ([Secato & Lopes, 2003](#)), com tamanho total de 260,1 KB. Cada documento é um arquivo com tamanho médio e desvio padrão de $0,9 \pm 0,3$ KB. Os documentos (resumos) estão relacionados à área de Raciocínio Baseado em Casos (RBC) e não possuem uma classe associada.

Na realidade, o trabalho aqui apresentado, é um trabalho inicial de um projeto mais abrangente, do qual participam diversos pesquisadores do LABIC ([Lopes, 2003](#)). Nesse projeto é proposta a construção de uma ferramenta denominada Ferramenta Inteligente de apoio à Pesquisa (FIP), cujo objetivo é localizar e recuperar automaticamente artigos e publicações na Web sobre um tema ou tópico de pesquisa selecionado pelo usuário; analisar os artigos; e construir uma mapa que represente uma ontologia relacionada ao tópico proposto pelo usuário. Esse mapa deve indicar ao pesquisador quais são as publicações/autores mais importantes, os temas relacionados, precedência histórica, relevância das publicações, entre outras informações. Inicialmente, foi decidido construir um corpora inicial, sem recuperação na Web, para facilitar o desenvolvimento e teste dos diversos módulos e técnicas propostas nesse projeto para desenvolver a FIP. O conjunto de documentos LNAI utilizado nos experimentos descritos a seguir é o primeiro conjunto digitalizado desse corpora inicial.

7.4.1 O Pré-processamento

Inicialmente, os documentos foram pré-processados usando a ferramenta PRETEXT. Nesse caso, foram encontrados 2471 *stems* em todo conjunto de documentos. O corte inferior de Luhn foi fixado em 3, assim, apenas *stems* com frequência 1 e 2 foram descartados. Na Figura 7.7 é mostrada a distribuição da frequência dos *stems* gerados. De um total de 2471 permaneceram apenas 571, cuja frequência média e desvio padrão é de $88,6 \pm 106,5$.

Figura 7.7: LNAI: frequência dos *stems*

Considerando que esse conjunto de documentos não possui a classe associada, os experimentos foram realizados utilizando a metodologia proposta no Capítulo 4, agrupando os resumos em clusters com o objetivo de descobrir algum significado conceitual relacionado aos clusters juntamente com um especialista da área. Assim, diversas configurações para execução dos experimentos foram estabelecidas para tentar atingir esse objetivo. Os resultados de alguns desses experimentos são mostrados a seguir. Uma descrição detalhada de todos os experimentos realizados encontra-se em (Martins, Matsubara, Monard, & Lopes, 2003).

7.4.2 Resultados Obtidos

Para esses experimentos, foram utilizadas várias listas de *stopwords* definidas pelo especialista, conforme os experimentos iam sendo realizados. A lista original de *stopwords* continha 571 palavras. No decorrer dos experimentos, as palavras consideradas irrelevantes para o domínio em questão, detectadas pelo especialista, foram inseridas em novas

listas de *stopwords*. O total de *stopwords* usadas nesses experimentos foi de 571, 746 ou 788 palavras.

Um outro aspecto considerado refere-se ao peso das palavras utilizadas no título dos artigos que constituem o conjunto desses documentos. O especialista tinha interesse em verificar a importância dessas palavras. Assim, foi decidido realizar alguns experimentos considerando um peso maior para essas palavras. Os resultados são mostrados na Tabela 7.12, semelhante à Tabela 7.2 na página 129, na qual as novas colunas são:

- ‘StpW’ representa a quantidade de palavras na lista de *stopwords*;
- ‘PesoT’ representa o peso atribuído às palavras do título dos resumos;
- ‘Clus’ representa o número de clusters encontrados por AutoClass;
- ‘Maj’ e ‘Min’ representam o número de exemplos nos clusters com a maior e menor quantidade de exemplos, respectivamente;

Tabela 7.12: LNAI: resultados para 1-gram

Resultados obtidos: não fixando clusters													
Exp	Inf	Sup	StpW	PesoT	stems	Med	Clus	Maj	Min	ErroA %	Err10cv %	# R	# AtrR
E_{1a}	3	–	571	1	861	tf	1	276	0	–	–	–	–
E_{1b}						$tfidf$	27	15	6	52,5	$92,4 \pm 1,1$	48	62
E_{2a}	3	–	571	5	1008	tf	1	276	0	–	–	–	–
E_{2b}						$tfidf$	25	16	6	94,2	$94,9 \pm 1,2$	0	0
E_{3a}	3	–	721	1	804	tf	1	276	0	–	–	–	–
E_{3b}						$tfidf$	30	13	5	95,3	$95,6 \pm 0,7$	0	0
E_{4a}	3	–	721	5	951	tf	1	276	0	–	–	–	–
E_{4b}						$tfidf$	23	18	7	93,5	$93,5 \pm 1,0$	0	0
Resultados obtidos: fixando clusters													
Exp	Inf	Sup	StpW	PesoT	stems	Med	Clus	Majo	Mino	ErroA %	Err10cv %	# R	# AtrR
E_{5a}	3	–	571	1	861	$tfidf$	2	139	137	10,9	$44,2 \pm 3,1$	12	30
E_{5b}							4	105	82	22,1	$68,0 \pm 3,6$	29	30
E_{5c}							3	75	64	23,6	$60,5 \pm 2,1$	19	45
E_{6a}	3	–	746	1	804	$tfidf$	2	138	138	10,5	$46,0 \pm 1,9$	11	29
E_{6b}					788					9,8	$44,3 \pm 2,8$	13	38
E_{6c}					763					19,9	$43,1 \pm 2,9$	7	26
E_{6d}					754					16,3	$47,9 \pm 3,3$	14	21
E_{7a}	3	295	788	1	759	$tfidf$	2	147	129	15,2	$50,3 \pm 3,8$	9	31
E_{7b}							5	62	47	26,8	$72,1 \pm 2,7$	33	61
E_8	3	–	746	5	951	$tfidf$	2	144	132	10,9	$48,2 \pm 4,1$	14	34
E_{9a}	5	295	746	1	584	$tfidf$	2	140	136	7,6	$39,6 \pm 3,8$	9	32
E_{9b}							5	64	47	33,0	$73,9 \pm 2,4$	30	59
E_{10}	5	600	746	5	752	$tfidf$	2	142	134	17,0	$50,0 \pm 2,7$	9	24

Inicialmente, foram executados experimentos usando apenas *stems* compostos por 1-gram. A Tabela 7.12 está dividida em duas partes: a primeira parte representa os

resultados obtidos sem fixar o número de clusters e a segunda os resultados fixando o número de clusters em 2, 3, 4 e 5. Quando o número de clusters não é fixado, pode ser observado que em todos os experimentos usando a medida *tf*, **AutoClass** não conseguiu encontrar mais do que um cluster. Isso significa que os dados (textos) não estão bem separados probabilisticamente quando representados com os valores dessa medida. É interessante salientar que em outros experimentos, mesmo fixando o número de clusters e usando a medida *tf*, **AutoClass** não conseguiu separar os dados, informando somente um cluster.

O número de clusters encontrados por **AutoClass** para medida *tfidf* varia entre 23 a 30 clusters, todos com probabilidade 1. Foi observado também que a distribuição dos exemplos nos clusters em praticamente todos os experimentos está balanceada. Por exemplo, no experimento E_{1b} foram encontrados 27 clusters entre os quais os clusters com maior e menor número de exemplos contêm 15 e 6 exemplos, respectivamente. Apesar de **AutoClass** conseguir dividir os dados em vários clusters com probabilidade 1 para a medida *tfidf*, **See5** não conseguiu induzir conhecimento a partir desses agrupamentos, mesmo quando o número de *stopwords* foi incrementado e, também, mesmo quando o peso do título foi aumentado em cinco vezes.

Assim, decidiu-se executar novamente os experimentos fixando o número de clusters em um número menor em relação àqueles encontrados utilizando o parâmetro *default* de **AutoClass**, usando apenas a medida *tfidf*. Primeiramente, fixou-se o número de clusters em 2. O erro encontrado por **See5** foi de 44,2% que é um valor próximo do erro da classe majoritária (49,6%). Os resultados são apresentados na segunda parte na Tabela 7.12. Após fixar o número de clusters em 2, novos experimentos foram realizados fixando vários valores para o número de clusters. Na Tabela 7.12 são mostrados apenas os resultados obtidos para o número de clusters fixado em 2, 3, 4 e 5. Como pode ser verificado, mesmo fixando o número de clusters os resultados não foram bons, apesar do erro ter decrementado.

Considerando que os resultados obtidos não foram bons usando essa metodologia, os dados foram submetidos ao algoritmo SVM Torch II, para verificar o desempenho usando outro algoritmo de aprendizado. Os resultados obtidos, utilizando o algoritmo SVM Torch II com o *Kernel* linear e valores *default*, para a mesma configuração do experimento E_{9a} , que representa o melhor resultado, são apresentados na Tabela 7.13.

Tabela 7.13: LNAI: resultados para 1-gram com SVM Torch II

Exp	Inf	Sup	<i>stems</i>	Med	Err10cvL %
E_{13}	5	295	584	<i>tfidf</i>	$24,0 \pm 10,9$

Como pode ser observado, os resultados obtidos pelo SVM não foram significativamente melhores do que os resultados obtidos pelos experimentos anteriores utilizando algoritmos de aprendizado simbólicos. Isso mais uma vez mostra que os dados não estão bem separados conceitualmente, para os clusters encontrados usando esses algoritmos.

Para verificar o desempenho dos *stems* representados com 2-gram, foram executados novos experimentos. PRETEXT encontrou 12391 *stems* com 2-gram, porém, 10642 *stems* 2-gram aparecem apenas 1 vez, o que corresponde a 85,9% de todos os *stems*. Os resultados dos experimentos são apresentados na Tabela 7.14, usando como corte inferior de Luhn o valor 3.

Tabela 7.14: LNAI: resultados para 2-gram sem conhecimento do especialista

Resultados obtidos: não fixando clusters													
Exp	Inf	Sup	StpW	PesoT	<i>stems</i>	Med	Clus	Majo	Mino	ErroA %	Err10cv %	# R	# AtrR
E_{11a}					666					48,9	$90,5 \pm 1,7$	57	68
E_{11b}					665					51,1	$89,8 \pm 1,9$	55	65
E_{11c}	3	–	746	1	664	<i>tfidf</i>	19	20	8	52,2	$90,2 \pm 0,9$	54	63
E_{11d}					654					49,3	$92,0 \pm 1,0$	54	65
E_{11e}					652					47,4	$89,7 \pm 1,8$	59	69
E_{12}	5				135		22	21	7	34,8	$70,3 \pm 1,5$	56	68
Resultados obtidos: fixando clusters													
Exp	Inf	Sup	StpW	PesoT	<i>stems</i>	Med	Clus	Majo	Mino	ErroA %	Err10cv %	# R	# AtrR
E_{13a}					135					22,1	$53,7 \pm 2,5$	43	45
E_{13b}	5	–	746	1	132	<i>tfidf</i>	7	66	24	23,6	$51,8 \pm 3,3$	42	46

Os resultados obtidos usando 2-gram foram um pouco melhores do que os obtidos por 1-gram se for considerado que o erro da classe majoritária é de 76,1%, ainda assim, não é um bom resultado. Como mencionado, diversos experimentos foram realizados, inclusive usando diversas combinações de *gram*, ou seja, para 1, 2 ou 3-gram individualmente, bem como para 1 e 2-gram, 2 e 3-gram, 1, 2 e 3-gram. Nos experimentos, foram utilizadas também taxonomias definidas pelo especialista. No entanto, nenhum dos experimentos conseguiu bons resultados para esse conjunto de textos.

Assim, é possível concluir que para um conjunto de dados textuais de tamanho muito pequeno e relacionados a um assunto muito específico, nesse caso Raciocínio Baseado em Casos, a tarefa de agrupamento em sub-temas do assunto específico é árdua e difícil, e outros métodos de agrupamento diferentes da metodologia usada neste trabalho devem ser também pesquisados. Entretanto, esses resultados iniciais mostraram a importância de tratar documentos maiores relacionados aos artigos a serem processados pela ferramenta FIP. Atualmente, estão sendo digitalizados os artigos completos (não somente os resumos) de outros LNAI, tanto para RBC quanto para artigos científicos de outras áreas, tais como Programação Lógica Indutiva e Recuperação da Informação. Quando esses documentos

maiores estiverem prontos, experimentos semelhantes aos aqui descritos serão realizados com o objetivo de auxiliar o especialista na rotulação desses documentos e na construção de classificadores precisos.

7.5 Considerações Finais

Na Tabela 7.15 é apresentado um resumo dos conjuntos de documentos utilizados nos experimentos descritos neste capítulo.

Tabela 7.15: Resumo dos conjuntos de dados textuais

	NILC	7SECTORS	BGMS	LNAI
Idioma	português	inglês	inglês	inglês
Tamanho KB	$24,9 \pm 13,8$	$5,4 \pm 6,1$	$1,9 \pm 3,5$	$0,9 \pm 0,3$
Qtde Textos	248	1729	332	276
<i>Stems</i> (total)	22214	28646	11419	2471
Método	categorização	categorização	categorização agrupamento	agrupamento

Os experimentos usando conjuntos de dados textuais realizados neste trabalho foram pré-processados utilizando as facilidades existentes na ferramenta PRETEXT. Pelas características implementadas no PRETEXT, a realização de experimentos visando a obtenção de um bom classificador se torna uma tarefa menos custosa, buscando uma representação com uma dimensão menor para o conjunto de atributos, mantendo ou melhorando a precisão do classificador induzido por meio da eliminação de *stems* com uma certa frequência, do incremento da lista de *stopwords* específicas do domínio, bem como na definição de taxonomias apropriadas para generalizar alguns dos atributos.

Deve ser observado que um dos objetivos deste trabalho é encontrar padrões, usando qualquer algoritmo de aprendizado não supervisionado, que possam ser explicados conceitualmente. Para explicar esses padrões conceitualmente foram utilizados, principalmente, os algoritmos C4.5rules, See5, CN2, que são algoritmos simbólicos. Foram também utilizados os algoritmos usando SVMTorch II e o Rainbow. Geralmente, os SVM apresentam bons resultados no caso de dados com alta dimensionalidade. Porém, os classificadores induzidos usando SVM são do tipo caixa preta e existem situações nas quais é mais importante induzir um classificador simbólico, embora com uma precisão menor, mas que permita interpretar, analisar e usar o conhecimento de diferentes formas, tais como organizar uma pequena função de índice para recuperar mais eficientemente documentos classificados.

No entanto, os experimentos realizados confirmam mais uma vez que a escolha do

algoritmo de aprendizado é dependente do conjunto de dados e da tarefa a ser realizada. A técnica de [SVM](#), que em conjunto de dados com uma alta dimensionalidade geralmente obtém um excelente desempenho comparado a outros algoritmos de aprendizado, mostrou em alguns experimentos realizados, que isso nem sempre ocorre. Trabalhos experimentais utilizando vários algoritmos e parâmetros é essencial nesta área de pesquisa.

Conclusão

“Tudo é loucura ou sonho no início. Nada do que o homem fez no mundo teve início de outra maneira. Mas já tantos sonhos se realizaram, que não temos direito de duvidar de nenhum.” [Monteiro Lobato]

8.1 Principais Contribuições

Neste trabalho foram pesquisados alguns problemas relacionados a dois tipos de conjuntos de dados utilizados por algoritmos de aprendizado: (i) dados não rotulados, ou seja, sem um conceito associado aos dados; (ii) dados não estruturados, tais como, dados textuais ou documentos. Vários desafios estão envolvidos com esses dois tipos de conjunto de dados. Nossos objetivos relacionados a dados estruturados e não estruturados, foram:

1. encontrar agrupamentos de conjunto de dados não rotulados para descobrir o conceito representado pelos clusters desses dados;
2. realizar o pré-processamento de dados textuais não estruturados para induzir um bom classificador.

Para o primeiro objetivo usando dados não rotulados, foi proposta uma metodologia para auxiliar o especialista na descoberta e interpretação de padrões presentes nesses dados usando algoritmos de aprendizado. A metodologia proposta é um processo iterativo e interativo que utiliza algoritmos de aprendizado não supervisionado para descobrir

clusters nos dados, e algoritmos de aprendizado supervisionado para induzir um conceito relacionado aos dados originais rotulados com o cluster ao qual pertencem. A partir dos resultados obtidos pelos algoritmos, **AutoClass** e **See5** utilizados neste trabalho, foi possível verificar os padrões existentes analisando as regras induzidas por **See5**, os principais atributos utilizados por ambos algoritmos, a sobreposição de clusters, entre outros, utilizando diversos conjuntos de dados.

Foram implementadas ferramentas de apoio que visam automatizar algumas etapas do processo, bem como confrontar informações utilizando os algoritmos de aprendizado, supervisionado e não supervisionado, para melhor compreensão dos dados. Também, usando a metodologia e as ferramentas implementadas, no caso das classes serem conhecidas é possível verificar a correspondência dos dados nos clusters $\times$ classes. Dessa forma, se os dados agrupados em clusters não correspondem às classes associadas, significa que um outro critério, diferente do utilizado correspondente às classes, foi adotado na descoberta dos clusters, o que pode levar à descoberta de novos padrões existentes nos dados diferentes dos que correspondem às classes associadas.

A metodologia proposta mostrou ser útil e viável para auxiliar na interpretação de clusters de dados. Os trabalhos resultantes das avaliações experimentais usando essa metodologia estão publicados em dois relatórios técnicos ([Martins, Monard, Haedo, & Matsudo, 2001](#); [Martins, Monard, & Halembeck, 2002b](#)), em um artigo publicado em congresso latino-americano ([Martins & Monard, 2000](#)) e em um artigo publicado em congresso internacional ([Martins, Monard, & Halembeck, 2002a](#)). Encontra-se também publicada em um relatório técnico a descrição dos parâmetros e execução do sistema **AutoClass** ([Matsubara, Martins, & Monard, 2002](#)).

O segundo objetivo deste trabalho está relacionado à descoberta de conhecimento em dados não estruturados no formato textual. O pré-processamento de textos é uma tarefa árdua. Dessa forma, uma das principais contribuições desse trabalho foi a implementação de uma ferramenta computacional denominada **PRETEXT**, que auxilia a etapa de pré-processamento de dados textuais.

PRETEXT não apenas transforma os textos escritos em inglês, português e espanhol em uma tabela atributo-valor usando a abordagem *bag-of-words*, mas também, reduz a dimensionalidade dos atributos, realiza indução construtiva por meio de taxonomias definidas pelo usuário, trabalha com várias listas de *stopwords*, gera informações relacionadas aos textos, tais como, quais as palavras que pertencem a cada *stem*, juntamente com suas frequências relativas, frequência dos *stems*, quantidade de documentos em que cada *stem* aparece, entre outros. O diferencial da ferramenta **PRETEXT** em relação a outras ferramentas existentes é justamente essa diversidade de informações geradas, as

funcionalidades existentes e a facilidade de interação com a mesma.

Por meio dos experimentos realizados em diversos conjuntos de textos usando PRETEXT e diferentes algoritmos de aprendizado foi possível analisar os resultados dos classificadores textuais obtidos. A redução da dimensionalidade dos atributos proposta neste trabalho se mostrou eficiente e viável na avaliação experimental realizada, os quais foram concentrados nos erros dos classificadores e na complexidade do modelo induzido.

Os trabalhos resultantes das avaliações experimentais em mineração de textos usando PRETEXT e algoritmos de aprendizado podem ser encontrados em três relatórios técnicos (Martins, Godoy, Monard, Matsubara, & Amandi, 2003; Martins, Monard, & Matsubara, 2003a; Martins, Matsubara, Monard, & Lopes, 2003), um artigo publicado em congresso nacional (Martins, Monard, & Matsubara, 2003c), um artigo em congresso latino-americano (Martins, Matsubara, & Monard, 2003) e um artigo publicado em congresso internacional (Martins, Monard, & Matsubara, 2003b). Também, a documentação detalhada relacionada a ferramenta computacional PRETEXT está disponível em (Matsubara, Martins, & Monard, 2003). Atualmente, está em preparação um artigo para ser submetido para publicação em periódico internacional (Martins & Monard, 2003).

8.2 Algumas Limitações

Os dois problemas abordados neste trabalho, usando conjunto de dados não rotulados e não estruturados, são problemas cujas possíveis soluções são difíceis de serem avaliadas em problemas do mundo real sem o auxílio do especialista. No caso de dados não rotulados quando o objetivo consiste em encontrar padrões por meio de técnicas de agrupamento, como saber se o agrupamento realizado é o mais adequado aos exemplos utilizados e tarefas especificadas para dados do mundo real?

A metodologia adotada para interpretação de clusters visa auxiliar o especialista a superar esse problema usando algoritmos de aprendizado supervisionado e não supervisionado. Neste trabalho, os algoritmos utilizados na realização dos experimentos foram o **AutoClass** e o **See5**. No entanto, em alguns experimentos pode-se perceber que devido ao *bias* diferente de cada algoritmo, os algoritmos encontram padrões diferentes e, assim, nem sempre os resultados são os esperados. Nesse caso, espera-se que a utilização de algoritmos de aprendizado com *bias* similares tenham um melhor desempenho.

Para dados textuais não estruturados, a maior limitação está nos próprios dados relacionados principalmente à quantidade de documentos disponíveis, o tamanho dos documentos e o domínio em questão. Ou seja, como o vetor que representa cada docu-

mento é um vetor de alta dimensão, é necessário um considerável número de documentos disponíveis ao algoritmo de aprendizado para que se obtenha uma boa representação relacionada com a frequência desses atributos, caso seja utilizada alguma medida relacionada à frequência dos atributos. Como pode ser verificado nos experimentos realizados com o conjunto de textos LNAI — Seção 7.4 na página 142 —, o tamanho dos documentos tem influência no desempenho do classificador.

Ainda, existe a limitação da própria abordagem *bag-of-words* pois um documento textual, por exemplo, não é simplesmente um conjunto de palavras. O contexto e o sentido das frases contêm informações que são perdidas quando os documentos são transformados em uma tabela atributo-valor. Assim, mineração de dados textuais é uma tarefa árdua pois além das próprias especificidades de qualquer processo relacionado a KDD, o pré-processamento dos documentos deve ser realizado de maneira tal que a nova representação desses documentos represente de forma mais adequada o conjunto de textos.

Uma dificuldade encontrada durante a realização deste trabalho está relacionada à disponibilidade de dados textuais que possam ser avaliados comparativamente com outros resultados publicados. Existem alguns conjuntos de textos disponíveis escritos em inglês, porém, é difícil textos escritos em português e espanhol disponíveis serem encontrados para esse fim¹. Entretanto, mesmo para os textos disponíveis escritos em inglês, muitas vezes utilizados nos experimentos publicados na literatura, a dificuldade em comparar os resultados experimentalmente é dependente de fatores tais como: o conjunto de textos que realmente foi utilizado nos experimentos (muitas vezes, apenas um subconjunto foi utilizado ou existem diferentes versões para um mesmo conjunto de dados), as medidas que foram usadas para avaliar o desempenho e, principalmente, o tipo de pré-processamento realizado nos conjuntos de textos. Ainda, deve ser levado em consideração que geralmente a avaliação de classificadores de textos é realizada experimentalmente e não analiticamente, devido principalmente à subjetividade e forma não estruturada dos dados (Sebastiani, 2002).

8.3 Trabalhos Futuros

Alguns possíveis refinamentos e extensões relacionados a este trabalho para trabalhos futuros são apresentados a seguir.

Metodologia de interpretação de Clusters Futuramente, pretende-se investigar o

¹Um dos objetivos futuros está relacionado com a construção de diversos conjuntos de documentos em português para servir de *benchmark* a trabalhos da área de mineração de textos.

uso de outros algoritmos de aprendizado supervisionado e não supervisionado na metodologia proposta. Como mencionado, os algoritmos de aprendizado usados neste trabalho, **AutoClass** e **See5**, utilizam *bias* diferentes. Assim, mesmo que um algoritmo tenha um bom desempenho para um conjunto de dados, o outro algoritmo pode não ter o mesmo desempenho. Nesse caso, acredita-se que algoritmos com *bias* semelhantes quando utilizados em conjunto possam obter um melhor desempenho. Como por exemplo, a utilização de algoritmos de aprendizado não supervisionado que usam medidas de distância para encontrar os clusters combinado com um algoritmo supervisionado que induz árvores de decisão.

Um outro trabalho a ser realizado é a integração da metodologia proposta ao projeto DISCOVER. Como mencionado na Seção 4.3 na página 67, a saída dos diversos algoritmos de agrupamento é fortemente dependente da técnica de agrupamento (otimização, hierárquica, probabilística, *clumping*) utilizada, apresentando distintos formatos de saída. A idéia, portanto, é encontrar uma sintaxe padrão para representar a saída dos diversos tipos de algoritmos que utilizam diferentes técnicas de agrupamento — Figura 4.10 na página 68. Dessa forma, qualquer técnica de agrupamento poderá ser usada na metodologia proposta conjuntamente com todos os algoritmos de aprendizado supervisionado, atualmente integrados no DISCOVER.

Mineração de textos usando a abordagem *bag-of-words* Em trabalhos futuros será melhor verificada a influência de todas as medidas de atribuição de valores aos atributos implementadas no PRETEXT — Seção 5.2 na página 74 — utilizando diferentes conjuntos de textos e verificando o comportamento de cada uma das medidas no desempenho dos classificadores induzidos.

Pretende-se também investigar uma forma de encontrar pontos de corte de Luhn (inferior e superior) de maneira que possam ser calculados de forma automática tendo como referência o tamanho do conjunto de textos e a frequência dos *stems*. Ou seja, calcular os pontos de corte que sejam dependentes e independentes do conjunto de textos que está sendo processado. Dependentes porque são calculados levando em consideração um conjunto específico de textos. Independentes porque podem ser realizados de forma automática para qualquer conjunto de textos.

Pretende-se investigar os atributos utilizados por diferentes classificadores. No caso de algoritmos de aprendizado simbólico, por exemplo, é possível verificar o conjunto de atributos que comparecem nas regras dos classificadores induzidos e verificar o comportamento desses algoritmos quando todos esses atributos, mas somente eles, são considerados para representar os documentos. Essa pode ser uma maneira de selecionar um pequeno conjunto de atributos que, possivelmente, são relevantes

na discriminação dos textos. Essa idéia foi utilizada no experimento E_9 realizado com o conjunto de Textos NILC, Tabela 7.2 na página 129. De um total de 6202 *stems* (atributos) apenas 19 *stems*, os quais apareceram conjuntamente nas regras induzidas por *See5* e CN2, foram suficientes para obter o melhor classificador.

Pretende-se avaliar conjunto de textos com classes desbalanceadas com o objetivo de melhorar a classificação da classe minoritária (Batista, Carvalho, & Monard, 2000). Por fim, uma outra questão a ser considerada no futuro é a construção e uso de taxonomias utilizando dicionários, ou outras informações, específicos da língua, a fim de facilitar o trabalho do usuário na definição dessas taxonomias.

Algoritmos de *Stemming*

Algoritmos de *stemming* consistem em uma normalização lingüística, na qual as formas variantes de uma palavra são reduzidas a uma forma comum tendo como consequência a diminuição do número de atributos. Muitas vezes, o algoritmo remove prefixos ou sufixos de uma palavra, ou mesmo transforma um verbo para sua forma no infinitivo. O algoritmo de *stemming* é, portanto, fortemente dependente da língua na qual os textos estão escritos. Para determinadas línguas, como as orientais, pode ser impossível ou extremamente complexa a aplicação de um algoritmo de *stemming*. Na ferramenta PRETEXT foram implementados os algoritmos de *stemming* para textos escritos na língua inglesa, portuguesa, espanhola e usando o algoritmo do Porter, cujas regras são descritas a seguir.

A.1 Regras do Algoritmo de *Stemming* para o Inglês

O algoritmo de *stemming* implementado neste trabalho para a língua inglesa é basicamente o algoritmo e a implementação original do Porter que se encontra em <http://www.tartarus.org/~martin/PorterStemmer>. Para um melhor entendimento do algoritmo, é descrito seu funcionamento como mostrado em (Porter, 1980). As regras foram definidas usando uma medida que determina quando uma palavra é suficientemente grande para poder eliminar o sufixo. Essa medida, denotada por m , é uma função das seqüências de vogais (a, e, i, o, u ou y se for precedida por uma consoante) seguidas por seqüências de consoantes. Se V é uma seqüência de vogais e C é uma seqüência de consoantes, então m é:

$$C(VC)^mV$$

onde C e V nos extremos são opcionais e m é o número de vezes que uma sequência VC é repetida dentro da palavra. Por exemplo:

- $m = 0$ free, why
- $m = 1$ frees, whose
- $m = 2$ prologue, compute

As regras para eliminar os sufixos são descritos da seguinte forma:

$$[condição] S_1 \rightarrow S_2$$

o que significa que se uma palavra termina com o sufixo S_1 e o *stem* que se encontra antes de S_1 satisfaz a condição estipulada, S_1 será trocada por S_2 . A $[condição]$, que é opcional, usualmente é dada por m , por exemplo:

$$(m > 1)EMENT \rightarrow$$

indica que se for verdade que $m > 1$ o sufixo EMENT é eliminado.

A $[condição]$ pode ser formada por outras condições concatenadas com os operadores lógicos AND, OR ou NOT, e pode incluir algum dos seguintes elementos:

- $* < X >$ o *stem* termina com a letra indicada por X
- $*v*$ o *stem* contém pelo menos uma vogal
- $*d$ o *stem* termina com consoante dupla
- $*o$ o *stem* termina com consoante-vogal-consoante e a última consoante não é w, x ou y .

Na Tabela A.1 são apresentadas as regras de eliminação de sufixos previstas pelo algoritmo na notação descrita.

Nº	Condição	Ação	Exemplo
1a		sses $\rightarrow$ es	caresses $\rightarrow$ caress
		ies $\rightarrow$ i	ponies $\rightarrow$ poni ties $\rightarrow$ ti
		ss $\rightarrow$ ss	caress $\rightarrow$ caress
		s $\rightarrow$	cats $\rightarrow$ cat
1b	$m > 0$	eed $\rightarrow$ ee	feed $\rightarrow$ feed agreed $\rightarrow$ agree
1b(a)	$*v*$	ed $\rightarrow$	plastered $\rightarrow$ plaster bled $\rightarrow$ bled
1b(b)	$*v*$	ing $\rightarrow$	motoring $\rightarrow$ motor sing $\rightarrow$ sing
	se(1b(a) OR 1b(b))	at $\rightarrow$ ate	conflat(ed) $\rightarrow$ conflate

continua na próxima página

continuação da página anterior

Nº	Condição	Ação		Exemplo	
		bl	→ ble	troubl(ed)	→ trouble
		iz	→ ize	siz(ed)	→ size
	se(1b(a) OR 1b(b)) AND (*d AND NOT (*l OR *s OR *Z))	at	→ single → letter	hopp(ing)	→ hop
				tann(ed)	→ tan
				fall(ing)	→ fall
				hiss(ing)	→ hiss
				fizz(ed)	→ fizz
	se(1b(b) AND ($m = 1$ AND *O))		→ e	fail(ing)	→ fail
				fil(ed)	→ file
1c	*v*	y	→ i	happy	→ happi
				sky	→ sky
2	$m > 0$	ational	→ ate	relational	→ relate
	$m > 0$	tional	→ tion	conditional	→ condition
				rational	→ rational
	$m > 0$	enci	→ ence	valenci	→ valence
	$m > 0$	anci	→ ance	hesitanci	→ hesitance
	$m > 0$	izer	→ ize	digitizer	→ digitize
	$m > 0$	abli	→ able	conformabli	→ conformable
	$m > 0$	alli	→ al	radicalli	→ radical
	$m > 0$	entli	→ ent	differentli	→ diferent
	$m > 0$	eli	→ e	vileli	→ vile
	$m > 0$	ousli	→ ous	analogousli	→ analogous
	$m > 0$	ization	→ ize	vietnamization	→ vietnamize
	$m > 0$	ation	→ ate	predication	→ predicate
	$m > 0$	ator	→ ate	operator	→ operate
	$m > 0$	alism	→ al	feudalism	→ feudal
	$m > 0$	iveness	→ ive	decisiveness	→ decisive
	$m > 0$	fulness	→ ful	hopefulness	→ hopeful
	$m > 0$	ousness	→ ous	callousness	→ callous
	$m > 0$	aliti	→ al	formaliti	→ formal
	$m > 0$	iviti	→ ive	sensitiviti	→ sensitive
	$m > 0$	biliti	→ ble	sensibiliti	→ sensible
3	$m > 0$	icate	→ ic	triplicate	→ triplic
	$m > 0$	ative	→	formative	→ form
	$m > 0$	alize	→ al	formalize	→ formal
	$m > 0$	iciti	→ ic	electriciti	→ electric
	$m > 0$	ical	→ ic	electrical	→ electric
	$m > 0$	ful	→	hopeful	→ hope
	$m > 0$	ness	→	goodness	→ good
4	$m > 1$	al	→	revival	→ reviv
	$m > 1$	ance	→	allowance	→ allow

continua na próxima página

continuação da página anterior

Nº	Condição	Ação	Exemplo
	$m > 1$	ence →	inference → infer
	$m > 1$	er →	airliner → airlin
	$m > 1$	ic →	gyroscopic → gyroscop
	$m > 1$	able →	adjustable → adjust
	$m > 1$	ible →	defensible → defens
	$m > 1$	ant →	irritant → irrit
	$m > 1$	ement →	replacement → replac
	$m > 1$	ment →	adjustament → adjust
	$m > 1$	ent →	dependent → depend
	$m > 1$ AND ($*s$ OR $*T$)	ion →	adoption → adopt
	$m > 1$	ou →	homologou → homolog
	$m > 1$	ism →	comunism → comun
	$m > 1$	ate →	activate → activ
	$m > 1$	iti →	angulariti → angular
	$m > 1$	ous →	homologous → homolog
	$m > 1$	ive →	effective → effect
	$m > 1$	ize →	bowdlerize → bowdler
5a	$m > 1$	e →	probate → probat
		→	rate → rate
	$m = 1$ AND NOT $*o$	e →	cease → ceas
5b	$m = 1$ AND $*d$ AND $*l$	→ single letter	controll → control roll → rol

Tabela A.1: Regras de eliminação de sufixos para o inglês

Como mencionado, no PRETEXT foram implementadas essas regras no algoritmo de *stemming* para o inglês. O algoritmo consiste basicamente em eliminar as palavras que são *stopwords* e aplicar as regras descritas em cada palavra que aparece nos documentos.

A.2 Regras do Algoritmo de *Stemming* para o Português

O algoritmo de *stemming* implementado para a língua portuguesa está baseado no algoritmo do Porter. Também, nesse algoritmo, os sufixos que estão ao final de uma palavra são eliminados, de acordo com um comprimento mínimo estabelecido e considerando algumas regras pré-estabelecidas.

Para estabelecer o comprimento mínimo são utilizadas duas variáveis, **posV** e **pos2**, que indicam duas posições críticas dentro da palavra. Ambas variáveis são complementares e possuem a mesma função de m , no algoritmo do Porter para inglês, isto é, determinar se

a palavra é suficientemente grande para eliminar um sufixo. **posV** é utilizado para eliminar terminações verbais, enquanto que **pos2** para eliminar outras formas de sufixos. Exemplos do uso dessas variáveis são mostrados na Figura A.1, onde **V** e **2**, são respectivamente, **posV** e **pos2** e são determinadas da seguinte forma:

- **pos2** está no fim de uma sequência de consoantes que segue a segunda sequência de vogais;
- se a palavra começa com duas ou mais vogais, **posV** está na primeira consoante;
- se começa com vogal-consoante, **posV** está na segunda vogal;
- se começa com consoante, **posV** está na primeira vogal ou na terceira letra que está mais a direita.

abandoná-lo	augusto
↑ ↑	↑ ↑
V 2	V 2
baseando-se	proposição
↑ ↑	↑ ↑
V 2	V 2

Figura A.1: Medidas de comprimento mínimo da palavra

Após determinar essas duas posições, **posV** e **pos2**, aplicam-se as regras descritas na Tabela A.2. As funções **depois(variável)** presente em algumas das regras denota que a regra só pode ser aplicada se o sufixo encontrar-se depois da posição indicada em **variável**.

Uma das grandes diferenças entre o algoritmo de *stemming* para palavras em inglês e palavras em português ou espanhol está no fato de que caso não seja possível eliminar nenhum sufixo de acordo com estas regras, são analisadas as terminações verbais da palavra. O fato das linguagens provenientes do latim terem formas verbais altamente conjugadas com sete tempos, cada uma com seis finais diferentes, é necessário um tratamento diferenciado para essas terminações. Para a língua portuguesa as terminações verbais são mostradas na Tabela A.3.

O algoritmo implementado no PRETEXT para o português manipula essas características e cada palavra, após ser submetida ao algoritmo de *stemming*, será denominada *stem*. Os passos utilizados no algoritmo podem ser descritos da seguinte forma:

Passo 1: Eliminam-se as palavras que são *stopwords* e alguns verbos irregulares, tais como **dar, diz, est, faz, hav, ir, pod, sab, ser, ter, ver, vir**. Para cada palavra

No	Condição	Ação	Exemplo
1		[ae]is → [ae]l ns → m res → r s →	casais → casal álbuns → álbum fatores → fator casas → casa
2	depois(pos2)	idade →	adversidade → advers
3	se(2) e depois(pos2)	abil → iv → ic →	amigabil(idade) → amig exclusiv(idade) → exclus infelic(idade) → infel
4	depois(pos2)	ic[ao] →	irônic[ao] → irôn
5	depois(pos2)	ável → ível →	lamentável → lament acessível → acess
6	depois(pos2)	ismo →	alcoolismo → alcool
7	se(1) ou depois(pos2)	ção → ção →	acomodaçõe(s) → acomoda resolução → resolu
8	se(1) ou depois(pos2)	ador → adora →	colonizador(es) → coloniz colonizador(as) → coloniz
9	depois(pos2)	os[ao] →	corajos[ao] → coraj
10	depois(pos2)	ista →	dermatologista → dermatolog
11	depois(pos2)	amento → imento → amente →	pensamento → pens discernimento → discern discretamente → discret
12	se(11) ou depois(pos2)	os → ativ → ad → iv → ic →	generos(amente) → gener comparativ(amente) → compar deliberad(amente) → deliber compulsiv(amente) → compuls demagogic(amente) → demagog
13	depois(pos2)	mente →	difícilmente → difícil
14	se(13) ou depois(pos2)	avel → ivel →	favoravel(mente) → favor possivel(mente) → poss
15	depois(pos2)	iv[ao] →	primitiv[ao] → primit
16	se(15) ou depois(pos2)	at →	recreat(ivo) → recre
17	depois(pos2)	eza →	sutileza → sutil

Tabela A.2: Regras de eliminação de sufixos para o português

verifica se a mesma não pertence a uma lista de *stopwords* da língua portuguesa. Caso a palavra pertencer à lista de *stopwords*, a mesma é ignorada e não será considerada como *stem*, ou seja, não será um atributo relevante. Além das *stopwords*, os verbos irregulares mais comuns também são eliminados os quais de certa forma, podem ser considerados como *stopwords*.

Passo 2: Estabelecem-se os valores de das variáveis **posV** e **pos2** indicando duas posições críticas dentro da palavra. Como mencionado, é importante manter um tamanho mínimo que um *stem* deve ter. Caso contrário, palavras muito pequenas podem ser reduzidas a um *stem* que não seja tão interessante quanto a palavra tal como aparece no texto. Por exemplo, a palavra **ação** poderia ser reduzida ao *stem* **a**;

Passo 3: Eliminam-se as terminações: -los, -las, -se e depois cada a palavra é submetida

1ª Conjugação						
Infinitivo	-ar					
	1ª PS	2ª PS	3ª PS	1ª PP	2ª PP	3ª PP
Presente	-o	-as	-a	-amos	-ais	-am
Subjuntivo	-e	-es	-e	-emos	-eis	-em
Futuro	-arei	-arás	-ará	-aremos	-areis	-arão
Condicional	-aria	-arias	-aria	-aríamos	-aríeis	-ariam
Imperfeito	-ava	-avas	-ava	-ávamos	-áveis	-avam
Passado	-ei	-aste	-ou	-ávamos	-astes	-aram
Imp. Subjuntivo	-asse	-asses	-asse	-ássemos	-ásseis	-assem
Mais Perfeito	-ara	-aras	-ara	-áramos	-áreis	-aram
Presente Partic.	-ando					
Passado Partic.	-ada -ado -adas -ados					
2ª Conjugação						
Infinitivo	-er					
	1ª PS	2ª PS	3ª PS	1ª PP	2ª PP	3ª PP
Presente	-o	-es	-e	-emos	-eis	-em
Subjuntivo	-a	-as	-a	-amos	-ais	-am
Futuro	-erei	-erás	-erá	-eremos	-ereis	-erão
Condicional	-eria	-erias	-eria	-eríamos	-eríeis	-eriam
Imperfeito	-ia	-ias	-ia	-íamos	-íeis	-iam
Passado	-i	-este	-eu	-emos	-estes	-eram
Imp. Subjuntivo	-esse	-esses	-esse	-éssemos	-ésseis	-essem
Mais Perfeito	-era	-eras	-era	-éramos	-éreis	-eram
Presente Partic.	-endo					
Passado Partic.	-ida -ido -idas -idos					
3ª Conjugação						
Infinitivo	-ir					
	1ª PS	2ª PS	3ª PS	1ª PP	2ª PP	3ª PP
Presente	-o	-es	-e	-imos	-is	-em
Subjuntivo	-a	-as	-a	-amos	-ais	-am
Futuro	-irei	-irás	-irá	-iremos	-ireis	-irão
Condicional	-iria	-irias	-iria	-iríamos	-iríeis	-iriam
Imperfeito	-ia	-ias	-ia	-íamos	-íeis	-iam
Passado	-i	-iste	-iu	-imos	-istes	-eram
Imp. Subjuntivo	-isse	-isses	-isse	-íssemos	-ísseis	-issem
Mais Perfeito	-ira	-iras	-ira	-íramos	-íreis	-iram
Presente Partic.	-indo					
Passado Partic.	-ida -ido -idas -idos					

Tabela A.3: Terminações verbais do português

às regras estabelecidas;

Passo 4: Aplicam-se as regras de eliminação de sufixos da Tabela A.2, desde que se cumpra o **Passo 2**;

Passo 5: Se não foi eliminado nenhum sufixo no passo anterior, são consideradas as terminações verbais dos verbos regulares que são apresentadas na Tabela A.3. Para eliminar quaisquer desses sufixos, deve ser verdade **depois(posV)**.

Com relação à implementação do algoritmo do Porter em particular e os seus resultados, vale ressaltar algumas observações, tais como o *stem* retornado para a palavra **pensamento**, por exemplo, será **pensam** e não **pens**, por causa do **pos2**. Assim como para a palavra **beleza** o *stem* será **beleza** e não **bel**. Alguns verbos de tamanho muito pequeno não são reduzidos, por exemplo, **abrir** fica **abrir**, **abrem** fica **abrem**, **abriu** fica **abriu**, devido à restrição de **posV**. Também, o algoritmo trabalha com palavras acentuadas normalmente, porém, quando uma palavra é reduzida a seu *stem*, todas as letras acentuadas foram transformadas em letras sem acentuação. Isto porque se pode gerar *stems* diferentes para palavras que deveriam ter o mesmo *stem*, como por exemplo, o *stem* da palavra **academia** ficaria **acad** e da palavra **acadêmicos** ficaria **acadêm**. As letras maiúsculas são transformadas em minúsculas. Os números foram eliminados bem como palavras de apenas um dígito.

A.3 Regras do Algoritmo de *Stemming* para o Espanhol

O algoritmo de *stemming* para o espanhol é semelhante ao algoritmo empregado para o idioma em português, ou seja, os sufixos que estão no final de um *stem* com um comprimento mínimo estabelecido são eliminados considerando algumas regras pré-estabelecidas (Godoy, 2001). As regras utilizadas para a língua espanhola são mostradas nas Tabelas A.4 na página oposta e A.5 na página 164 e os passos utilizados pelo algoritmo para textos em espanhol implementado no PRETEXT são os mesmos utilizados para os textos em português, Seção A.2 na página 158.

Nº	Condição	Ação	Exemplo
1		s →	absurdas → absurda
2	depois(pos2)	idad →	actividad → activ
3	depois(pos2)	idade →	actividade(s) → activ
4	Se (2 ou 3) e depois(pos2)	abil → iv → ic →	irresponsabil(idad) → irrespons radioactiv(idad) → radioact complic(idad) → compl
5	depois(pos2)	ic[ao] →	ecológic[ao] → ecológ
6	depois(pos2)	logía → log	analogía → analog
7	depois(pos2)	able →	aceptable → acept
8	depois(pos2)	ible →	apacible → apac
9	depois(pos2)	ismo →	atletismo → atlet
10	depois(pos2)	ación →	coronación → coron
11	depois(pos2)	ución → u	atribución → atribuc
12	depois(pos2)	acione →	alteracione(s) → alter
13	depois(pos2)	ucione → u	atribucione(s) → atrib
14	depois(pos2)	ador →	acusador → acus
15	depois(pos2)	ador[ae] →	innovador → innov
16	depois(pos2)	os[ao] →	ingenioso → ingeni
17	depois(pos2)	ista →	maquinista → maquin
18	depois(pos2)	amiento → imiento →	mejoramiento → mejor ofrecimiento → ofrec
19	depois(pos2)	amente →	oportunamente → oportun
20	se(19) e depois(posV)	os → ativ → ad → ic → iv →	orgullos(amente) → orgull significativ(amente) → signific acertad(amente) → acert demagogic(amente) → demagog figurativ(amente) → figurat
21	depois(pos2)	mente →	finalmente → final
22	se(21) e depois(pos2)	able → ible →	inevitable(mente) → inevit irremisible(mente) → irremis
23	depois(pos2)	iv[ao] →	obsesiva → obses
24	se(23) e depois(pos2)	at →	legislat(ivo) → legisl
25	depois(pos2)	anza →	confianza → confi
26	se(10 ou 12 ou 14) e depois(pos2)	ic →	calific(adora) → calif

Tabela A.4: Regras de eliminação de sufixos para o espanhol

1ª Conjugação						
Infinitivo	-ar					
	1ª PS	2ª PS	3ª PS	1ª PP	2ª PP	3ª PP
Presente	-o	-as	-a	-amos	-áis	-an
Subjuntivo	-e	-es	-e	-emos	-éis	-en
Futuro	-aré	-arás	-ará	-aremos	-aréis	-arán
Condicional	-aría	-arías	-aría	-aríamos	-aríais	-arían
Imperfeito	-aba	-abas	-aba	-ábamos	-abais	-aban
Passado	-é	-aste	-ó	-amos	-asteis	-aron
Imp. Subjuntivo	-ase	-ases	-ase	-ásemos	-aseis	-asen
	-ara	-aras	-ara	-áramos	-árais	-aran
Presente Partic.	-ando					
Passado Partic.	-ada -ado -adas -ados					
Imperativo	-ad					
2ª Conjugação						
Infinitivo	-er					
	1ª PS	2ª PS	3ª PS	1ª PP	2ª PP	3ª PP
Presente	-o	-es	-e	-emos	-éis	-enm
Subjuntivo	-a	-as	-a	-amos	-áis	-an
Futuro	-eré	-erás	-erá	-eremos	-eréis	-erán
Condicional	-ería	-erías	-ería	-eríamos	-eríais	-erían
Imperfeito	-ía	-ías	-ía	-íamos	-íais	-ían
Passado	-í	-iste	-ió	-imos	-isteis	-ieron
Imp. Subjuntivo	-iese	-ieses	-iese	-iésemos	-ieseis	-iesen
	-iera	-ieras	-iera	-iéramos	-ierais	-ieran
Presente Partic.	-iendo					
Passado Partic.	-ida -ido -idas -idos					
Imperativo	-ed					
3ª Conjugação						
Infinitivo	-ir					
	1ª PS	2ª PS	3ª PS	1ª PP	2ª PP	3ª PP
Presente	-o	-es	-e	-imo	-ís	-en
Subjuntivo	-a	-as	-a	-amos	-áis	-an
Futuro	-iré	-irás	-irá	-iremos	-iréis	-irán
Condicional	-iría	-irías	-iría	-iríamos	-iríais	-irían
Imperfeito	-ía	-ías	-ía	-íamos	-íais	-ían
Passado	-í	-iste	-ió	-imos	-isteis	-ieron
Imp. Subjuntivo	-iese	-ieses	-iese	-iésemos	-ieseis	-iesen
	-iera	-ieras	-iera	-iéramos	-ierais	-ieran
Presente Partic.	-iendo					
Passado Partic.	-ida -ido -idas -idos					
Imperativo	-id					

Tabela A.5: Terminações verbais do espanhol

Sintaxe de Dados do Discover

A sintaxe padrão de dados do DISCOVER— DSX¹ — proposta para os exemplos (Batista, 2001; Batista & Monard, 2003) utiliza arquivos do tipo texto para declarar os nomes dos atributos (e seus respectivos domínios) e, quando for o caso, os valores que esses atributos assumem no conjunto de exemplos. Os nomes dos atributos são declarados em um arquivo de nomes, designados com a extensão **.names**. Os valores que esses atributos assumem em um conjunto de exemplos são declarados no arquivo de dados, designados com a extensão **.data**. Um conjunto de exemplos somente está na sintaxe padrão do DISCOVER se esses dois arquivos estiverem presentes. Os dois arquivos devem ter o mesmo nome, se diferenciando apenas pela extensão.

A sintaxe dos arquivos de nomes (**.names**) e dados (**.data**) é uma extensão do formato utilizado pelo C4.5. Essa extensão tem como principal objetivo adequar o formato dos arquivos com as necessidades do projeto DISCOVER.

A primeira declaração em um arquivo de nomes define qual deve ser o nome do atributo classe, se ele existir. Se o atributo classe não existir, então o nome **null** deve ser utilizado. Em outras palavras, **null** indica que o conjunto de exemplos correspondente pode ser utilizado para AM não supervisionado.

No caso do atributo classe existir, ele pode ser qualquer atributo presente no conjunto de exemplos. Após a declaração do atributo classe são declarados os demais atributos. Cada atributo possui um identificador e um domínio a ele associado. São válidos os

¹Discover Dataset Syntax.

identificadores que são combinações de números, letras e “_” (sublinhado), em qualquer seqüência. Para identificadores mais complexos que envolvem outros caracteres que não sejam os especificados anteriormente (como espaços, letras acentuadas, etc) é necessário colocar o identificador entre aspas. Desta forma, são identificadores válidos: abc, 1, 1a, _1a, “_12a”, “válido”. Na Figura B.1 é mostrada a gramática para o arquivo de nomes.

```

S ::= <class-defs>
    | <feature-defs>

<class-defs> ::= <feature-name>.
    | null.

<feature-name> ::= <identifier>

<feature-defs> ::= <feature-name> : <feature-type> .
    | <feature-name> : <feature-type> : <extended-defs> .
    | <feature-name> : <feature-type> := <expression> : <extended-defs> .

<feature-type> ::= real
    | integer
    | boolean
    | nominal
    | nominal (<list>)
    | enumerated (<list>)
    | date
    | time
    | string

<extended-defs> ::= <extended-def>
    | <extended-def> : <extended-defs>

<extended-def> ::= <identifier>
    | <identifier> (<list>)

<list> ::= <identifier>
    | <identifier> , <list>

```

Figura B.1: Gramática do arquivo de nomes

Para ilustrar, considere o freqüentemente utilizado conjunto de dados *voyage* (Quinlan, 1988) que contém informações sobre medições diárias das condições metereológicas e uma classificação se o dia é apropriado a uma visita à fazenda ou não, no qual cada exemplo é composto pelos atributos:

- outlook: assume os valores “sunny”, “overcast” ou “rain”;

- temperature: um valor numérico que indica a temperatura em graus *Celsius*;
- humidity: outro valor numérico que indica a umidade relativa do ar;
- windy: assume os valores “yes” ou “no”.

Nesse conjunto de dados o nome do atributo classe é **voyage** e pode assumir os valores “go” e “dont_go”. O arquivo **voyage.names** correspondente é mostrado na Figura B.2.

```
voyage.

outlook:    sunny, overcast, rain.
temperature: continuous.
humidity:   continuous.
windy:      true, false.
voyage:     go, dont_go.
```

Figura B.2: O arquivo **voyage.names**

Como já mencionado, se o primeiro nome desse arquivo de nomes for **null**, então, esse conjunto de exemplos não possui um atributo de classe, *i.e.*, seria um conjunto de exemplos não rotulados.

O arquivo de dados correspondente ao arquivo de nomes contém n exemplos, um exemplo em cada linha. Cada exemplo i consiste nos valores dos atributos desse exemplo, separados por vírgula. Os valores de cada atributo devem pertencer ao domínio especificado para esse atributo no arquivo de nomes. Desta forma, o “separador de registros” (*record separator*) é o caracter de nova linha (representado em muitas linguagens de programação por **\n**) e o separador de campos (*field separator*) é a vírgula (,). Cada valor presente em uma linha está associado a um atributo do arquivo de nomes. Sendo assim, a ordem em que os valores são declarados em uma determinada linha deve ser a mesma na qual os atributos foram declarados no arquivo de nomes. Valores “*desconhecidos*” são representados por um ponto de interrogação (?) e valores “*não-se-aplica*” são representados com um ponto de exclamação (!). Na Figura B.3 é mostrado o arquivo de exemplos para alguns exemplos do conjunto de dados **voyage**.

Como pôde ser observado, o formato padrão de exemplos definido no projeto DISCOVER pode ser utilizado tanto para AM supervisionado como não supervisionado.

```
sunny,    25, 72, yes, go
sunny,    28, 91, yes, dont_go
overcast, 23, 90, yes, go
overcast, 19, 65, yes, dont_go
rain,     22, 95, no,  go
rain,     19, 70, yes, dont_go
...
```

Figura B.3: O arquivo `voyage.data`

Lista de Abreviaturas

AM Aprendizado de Máquina

BEP *Break-even point*

DAD Discover × AutoClass × Discover

DF *Document Frequency thresholding*

DLE *Discover Learning Environment*

DM *Data Mining*

DSX *Discover Dataset Syntax Environment*

EM *Expectation-Maximization*

EPH *Encuesta Permanente de Hogares*

FIP Ferramenta Inteligente de apoio à Pesquisa

IA Inteligência Artificial

LABIC Laboratório de Inteligência Computacional

KDD *Knowledge Discovery in Databases*

KDT *Knowledge Discovery in Texts*

MD Mineração de Dados

MT Mineração de Textos

Perl *Practical Extraction and Report Language*

PLN Processamento de Língua Natural

RBC Raciocínio Baseado em Casos

RI Recuperação de Informação

SVM *Support Vector Machines*

UCI University of California at Irvine

Weka *Waikato Environment for Knowledge Analysis*

Yale *Yet Another Learning Environment*

Referências Bibliográficas

- Aas, K. & L. Eikvil (1999). Text categorisation: A survey. <http://citeseer.nj.nec.com/aas99text.html>. 35, 44
- Anderberg, M. R. (1973). *Cluster Analysis for Applications*. New York: Academic Press. 53
- Apté, C., F. Damerau, & S. M. Weiss (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems* 12(3), 233–251. <http://citeseer.nj.nec.com/apte94automated.html>. 35, 36, 40
- Baeza-Yates, R. (1998). Searching the WWW: Challenges and possible solutions. In *Proceedings of IBERAMIA'98, Lecture Notes in Artificial Intelligence*, Portugal, Lisboa, pp. 39–51. Springer Verlag. 34
- Baker, L. D. & A. K. McCallum (1998). Distributional clustering of words for text classification. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, & J. Zobel (Eds.), *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, Melbourne, AU, pp. 96–103. ACM Press, New York, US. <http://citeseer.nj.nec.com/baker98distributional.html>. 35
- Banerjee, S. & T. Pedersen (2003). The design, implementation and use of the ngram statistics package. In A. Gelbukh (Ed.), *CICLing-2003, Lecture Notes in Computer Science* 2588, pp. 370–381. Springer-Verlag. <http://citeseer.nj.nec.com/567202.html>. 72
- Baranauskas, J. A. (2001). Extração automática de conhecimento por múltiplos indutores. Tese de Doutorado, ICMC-USP, <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-08102001-112806>. 6

- Baranauskas, J. A. & G. E. A. P. A. Batista (2000). O projeto DISCOVER: Idéias iniciais. Comunicação pessoal. 5
- Baranauskas, J. A. & M. C. Monard (2000a). An environment for rule extraction and evaluation from databases. In *Proceedings of the International Joint Conference IBERAMIA/SBIA (2000)*, Atibaia, SP, Brazil, pp. 187–196. 122
- Baranauskas, J. A. & M. C. Monard (2000b). Reviewing some machine learning concepts and methods. Relatório Técnico 102, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_102.ps.zip. 101, 141
- Baranauskas, J. A. & M. C. Monard (2000c). An unified overview of six supervised symbolic machine learning inducers. Relatório Técnico 103. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_103.ps.zip. 60, 127
- Baranauskas, J. A. & M. C. Monard (2003). Combining symbolic classifiers from multiple inducers. *Knowledge Based Systems* 16(3), 129–136. Elsevier Science. 6
- Basili, R., A. Moschitti, & M. T. Pazienza (2000). Language-sensitive text classification. In *Proceedings of RIAO-00, 6th International Conference Recherche d'Information Assistee par Ordinateur*, Paris, France, pp. 331–343. 15, 35
- Batista, G. E. A. P. A. (2001). Sintaxe padrão do arquivo de exemplos do projeto DISCOVER. <http://www.icmc.sc.usp.br/~gbatista/Discover/SintaxePadraoFinal.htm>. 165
- Batista, G. E. A. P. A. (2003). Pré-processamento de dados em aprendizado de máquina supervisionado. Tese de Doutorado, ICMC-USP. <http://www.icmc.usp.br/~gbatista/pdfs/TeseDoutorado.pdf>. 4, 6
- Batista, G. E. A. P. A., A. Carvalho, & M. C. Monard (2000). Applying one-sided selection to unbalanced datasets. In O. Cairo, L. E. Sucar, & F. J. Cantu (Eds.), *Mexican International Conference on Artificial Intelligence 2000*, pp. 315–325. Springer-Verlag. Lecture Notes in Artificial Intelligence, Best Paper Award Winner, <http://www.icmc.usp.br/~gbatista>. 154
- Batista, G. E. A. P. A. & M. C. Monard. The Discover Object Library (DOL) user's manual. (a ser publicado). 6
- Batista, G. E. A. P. A. & M. C. Monard (2003). Descrição da arquitetura e do projeto do ambiente computacional Discover Learning Environmet - DLE. Relatório Técnico 187,

- ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_187.zip. 6, 96, 133, 165
- Bekkerman, R., R. El-Yaniv, N. Tishby, & Y. Winter (2003). Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research* 3, 1183–1208. 36
- Bernardini, F. C. (2002). Combinação de classificadores simbólicos para melhorar o poder preditivo e descritivo de *ensembles*. Dissertação de Mestrado, ICMC-USP. 6
- Bradley, P. S. & U. M. Fayyad (1998). Refining initial points for *k*-means clustering. In J. W. Shavlik (Ed.), *Proceedings of Fifteenth International Conference on Machine Learning — ICML-1998*, Madison, Wisconsin, USA, pp. 91–99. Morgan Kaufmann. 23
- Caropreso, M. F., S. Matwin, & F. Sebastiani (2001). A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In A. G. Chin (Ed.), *Text Databases and Document Management: Theory and Practice*, pp. 78–102. Hershey, US: Idea Group Publishing. <http://citeseer.nj.nec.com/379242.html>. 35
- 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>. 58
- 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. 44, 58
- Chiara, R. (2003). Aplicação de data mining em logs de servidores Web. Dissertação de Mestrado, ICMC-USP. 22
- Clark, P. & R. Boswell (1989). The CN2 induction algorithm. *Machine Learning* 3(4), 261–283. 127
- Cohen, W. W. & H. Hirsh (1998). Joins that generalize: text classification using WHIRL. In R. Agrawal, P. E. Stolorz, & G. Piatetsky-Shapiro (Eds.), *Proceedings of KDD-98, 4th International Conference on Knowledge Discovery and Data Mining*, New York, US, pp. 169–173. AAAI Press, Menlo Park, US. <http://citeseer.nj.nec.com/cohen98joins.html>. 35
- Cohen, W. W. & Y. Singer (1996). Context-sensitive learning methods for text categorization. In H.-P. Frei, D. Harman, P. Schäuble, & R. Wilkinson (Eds.), *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Devel-*

- opment in *Information Retrieval*, Zürich, CH, pp. 307–315. ACM Press, New York, US. <http://citeseer.nj.nec.com/cohen96contextensitive.html>. 35
- Collobert, R. & S. Bengio (2001). SVM Torch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research* 1, 143–160. 127
- Cover, T. & J. Thomas (1991). *Elements of Information Theory*. Wiley. 43
- Crawford, D. (1998). Gnuplot: An interactive plotting program. <http://www.ucc.ie/gnuplot/gnuplot.html>. 91, 126
- Deerwester, S. C., S. T. Dumais, T. K. Landauer, G. W. Furnas, & R. A. Harshman (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science* 41(6), 391–407. <http://citeseer.nj.nec.com/deerwester90indexing.html>. 43
- Dietterich, T. G. (1997). Limitations on inductive learning (extended abstract). <ftp://ftp.cs.orst.edu/pub/tgd/papers>. 18
- Dörre, J., P. Gerstl, & R. Seiffert (1999). Text mining: Finding nuggets in mountains of textual data. San Diego, CA, pp. 398–401. ACM. 4
- Dosualdo, D. G. (2003). Investigação de regressão no processo de mineração de dados. Dissertação de Mestrado, ICMC-USP. 6
- 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>. 36
- Everitt, B. (1980). *Cluster Analysis*. New York: Academic Press. 27
- Fayyad, U. (1998). Taming the giants and the monsters: Mining large databases for nuggets of knowledge. <http://citeseer.nj.nec.com/fayyad98taming.html>. 4
- Fayyad, U. M., G. G. Grinstein, & A. Wierse (2002). *Information Visualization in Data Mining and Knowledge Discovery*. Morgan Kaufmann Publishers. 4, 35
- Fayyad, U. M., G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (1996). *Advances in Knowledge Discovery and Data Mining*. California: AAAI Press / The MIT Press. 2, 3
- Fischer, S., R. Klinkenberg, I. Mierswa, & O. Ritthoff (2002). Yale: Yet another learning environment — tutorial. Technical Report CI-136/02, University of Dortmund, Collaborative Research Center 531. 5

- Fisher, L. & J. W. Van Ness (1971). Admissible clustering procedures. *Biometrika* 58, 91–104. 21
- Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research* 3, 1289–1305. 36, 50
- Freitas, A. (1999). On rule interestingness measures. *Knowledge-Based Systems* 12(5-6), 309–315. 18
- Gelbukh, A. & G. Sidorov (2001). Zipf and heaps laws coefficients depend on language. In *Proceedings CICLing2001, Conference on Intelligent Text Processing and Computational Linguistics, Lecture Notes in Computer Science*, Number 2004. <http://www.gelbukh.com/CV/Publications/2001/CICLing-2001-Zipf.htm>. 42
- Geromini, M. R. (2002). Projeto e desenvolvimento da interface gráfica do sistema DISCOVER. Qualificação de Mestrado, ICMC-USP. 6
- Godoy, D. L. (2001). Personalsearcher: Un agente inteligente para búsqueda de información en la WWW. Tesis de Maestría, ISISTAN, Argentina. 162
- Gomes, A. K. (2002). Análise do conhecimento extraído de classificadores simbólicos utilizando medidas de avaliação e interessabilidade. Dissertação de Mestrado, ICMC-USP, <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-04072002-144610>. 18
- Guha, S., R. Rastogi, & K. Shim (1998). Cure: An efficient clustering algorithm for large databases. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, USA, pp. 73–84. ACM Press. 26
- Halembeck, G. C. (2001). Sistemas para *clustering*. Comunicação pessoal. 28
- Halkidi, M., M. Vazirgiannis, & Y. Batistakis (2000). Quality scheme assessment in the clustering process. In *Proceedings of the Lecture Notes in Artificial Intelligence – Principles of Data Mining and Knowledge Discovery 1910*, Lyon, France, pp. 265–276. 54
- Hamerly, G. & C. Elkan (2002). Alternatives to the k -means algorithm that find better clusterings. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management — CIKM-2002*, McLean, Virginia, USA, pp. 600–607. ACM Press. <http://doi.acm.org/10.1145/584792.584890>. 23

- Hanson, R., J. Stutz, & P. Cheeseman (1991). 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>. 58
- Harman, D. (1991). How effective is suffixing? *Journal of the American Society for Information Science* 42(1), 7–15. 41
- Hatzivassiloglou, V., L. Gravano, & A. Maganti (2000). An investigation of linguistic features and clustering algorithms for topical document clustering. In *Proceedings of the 23rd Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-00)*, Athens, Greece, pp. 224–231. 32
- Huang, Z. (1997). A fast clustering algorithm to cluster very large categorical data sets in data mining. In *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (SIGMOD-DMKD'97)*, Tucson, Arizona. 26
- IBM (2000). *DB2 Intelligent Miner Brochure*. <http://www-4.ibm.com/software/data/iminer/fordata/>. 4
- Imamura, C. Y. (2001). Pré-processamento para extração de conhecimento de bases textuais. Dissertação de Mestrado, ICMC-USP. 41
- Jacobs, P. (1992). Joining statistics with NLP for text categorization. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pp. 178–185. Morristown: Assoc. Comput. Linguistics. 35
- Jain, A. K., M. N. Murty, & P. J. Flynn (1999). Data clustering: A review. *ACM Computing Surveys* 31(3), 264–323. 25, 26, 69
- Joachims, T. (1997). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In D. H. Fisher (Ed.), *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Nashville, US, pp. 143–151. Morgan Kaufmann Publishers, San Francisco, US. <http://citeseer.nj.nec.com/article/joachims97probabilistic.html>. 35
- Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines*. Kluwer Academic Publishers. 32, 36, 39, 44, 131
- Joachims, T., D. Freitag, & T. M. Mitchell (1997). Web watcher: A tour guide for the world wide web. In *IJCAI (1)*, pp. 770–777. <http://citeseer.nj.nec.com/joachims96webwatcher.html>. 34

- Knuth, D. E. (1973). *The Art of Computer Programming*, Volume 3. Addison-Wesley. 42
- Kohavi, R., D. Sommerfield, & J. Dougherty (1994). *MLC++*: A machine learning library in C++. In *Tools with Artificial Intelligence*, pp. 740–743. IEEE Computer Society Press. 5
- Kohavi, R., D. Sommerfield, & J. Dougherty (1996). Data mining using *MLC++*: A machine learning library in C++. In *Tools with Artificial Intelligence*, pp. 234–245. IEEE Computer Society Press. 5
- Kopla, H. & P. Daly (1999). *A Guide to L^AT_EX: Document Preparation for Beginners and Advanced Users* (3 ed.). Addison-Wesley. 91
- Krovetz, R. (1995). *Word Sense Disambiguation for Large Text Databases*. Tese de Doutorado, University of Massachusetts at Amherst. 41
- Kubat, M., I. Bratko, & R. S. Michalski (1998). A Review of Machine Learning Methods. *Machine Learning and Data Mining*. John Wiley & Sons. 54
- Lagus, K. (2000). *Text Mining with the WEBSOM*. Tese de Doutorado, Department of Computer Science and Engineering, Helsinki University of Technology, Finland. 35
- Lee, H. D. (2000). Seleção e construção de features relevantes para o aprendizado de máquina. Dissertação de Mestrado, ICMC-USP, <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-15032002-113112>. 44
- Lennon, M., D. Pierce, B. Tarry, & P. Willett (1981). An evaluation of some conflation algorithms for information retrieval. *Journal of Information Science* 3, 177–183. 41
- Lewis, D. D. (1992). 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. 36
- Li, H. & A. K. Jain (1998). Classification of text documents. *The Computer Journal* 41(8), 537–546. 35
- Li, H. & K. Yamanishi (1999). Text classification using ESC-based stochastic decision lists. In *Proceedings of CIKM-99, 8th ACM International Conference on Information and Knowledge Management*, Kansas City, US, pp. 122–130. ACM Press, New York, US. <http://citeseer.nj.nec.com/li02text.html>. 35

- Lodhi, H., J. Shawe-Taylor, N. Cristianini, & C. J. C. H. Watkins (2002). Text classification using string kernels. *2*, 419–444. <http://citeseer.nj.nec.com/lodhi02text.html>. 35
- Lopes, A. A. (2003). Ferramenta inteligente de apoio à pesquisa para localizar e recuperar automaticamente artigos e publicações científicas na Web - proposta de trabalho conjunto. Comunicação pessoal. 142
- Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics* 11(1-2), 22–31. 40
- Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development* 2(2), 159–165. 42
- MacQueen, J. (1967). Some methods for classification analysis of multivariate observations. In *Proceedings 5th Berkeley Symp. Math. Stat. Prob.*, pp. 281. 22
- Mani, I. (2001). *Automatic Summarization*. Amsterdam: John Benjamins Publishing Co. 32
- Martins, C. A., D. Godoy, M. C. Monard, E. T. Matsubara, & A. Amandi (2003). Uma experiência em mineração de textos utilizando clustering probabilístico e clustering conceitual. Relatório Técnico 205, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_205.zip. 56, 141, 142, 151
- Martins, C. A., E. T. Matsubara, & M. C. Monard (2003). Um estudo de caso utilizando uma ferramenta computacional que auxilia na redução da dimensão da representação de documentos em tarefas preditivas de mineração de textos. In *Proceedings of IV Workshop on Advances and Trends in AI for Problem Solving - ATAI*, Chillán, Chile. (submetido). 135, 151
- Martins, C. A., E. T. Matsubara, M. C. Monard, & A. A. Lopes (2003). Interpretação de clusters em mineração de textos: um estudo de caso real. Relatório Técnico, ICMC-USP (a ser publicado). ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/. 56, 143, 151
- Martins, C. A. & M. C. Monard (2000). Interpretação de clusters utilizando aprendizado de máquina simbólico. In L. E. Vaz (Ed.), *Proceedings of the 21st Iberian Latin American Congress on Computational Methods in Engineering (CILAMCE)*, Rio de Janeiro. LMC/EPUSP. 13 pgs (CD-ROM). 56, 150

- Martins, C. A. & M. C. Monard (2003). Analyzing the influence of the dimensionality and measures used to represent texts using the bag-of-word approach on the induced classifiers performance. *Applied Artificial Intelligence*. (a ser submetido). 151
- Martins, C. A., M. C. Monard, A. S. Haedo, & N. L. Matsudo (2001). Uma metodologia para auxiliar o processo de interpretação semi-automática de clusters. Relatório Técnico 133, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_133.ps.zip. 56, 116, 121, 150
- Martins, C. A., M. C. Monard, & G. C. Halembeck (2002a). Combining clustering and inductive learning to find and interpret patterns from datasets. In N. C. Debnath, D. Riesco, & G. Montejano (Eds.), *Proceedings of the ACIS, International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA'02)*, pp. 90–95. (Também publicado nos Cadernos de Computação ICMC-USP, N. 02, pp. 309-322). 56, 150
- Martins, C. A., M. C. Monard, & G. C. Halembeck (2002b). A computational framework for interpreting clusters through inductive learning. Relatório Técnico 173, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_173.zip. 56, 150
- Martins, C. A., M. C. Monard, & E. T. Matsubara (2003a). Extração de conhecimento em dados textuais escritos em português: um estudo de caso. Relatório Técnico, ICMC-USP (a ser publicado). ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/. 151
- Martins, C. A., M. C. Monard, & E. T. Matsubara (2003b). Reducing the dimensionality of bag-of-words text representation used by learning algorithms. In *Proceedings of The Third IASTED International Conference on Artificial Intelligence and Applications (AIA 2003)*, Benalmádena, Espanha. (No prelo). 131, 151
- Martins, C. A., M. C. Monard, & E. T. Matsubara (2003c). Uma ferramenta computacional para auxiliar no pré-processamento de textos. In R. de Oliveira Anido & P. C. Masiero (Eds.), *Anais do XXIII Congresso da Sociedade Brasileira de Computação - IV Encontro Nacional de Inteligência Artificial (ENIA)*, Campinas, SP. 6 pgs. (CD-ROM). 131, 151
- Matsubara, E. T., C. A. Martins, & M. C. Monard (2002). Uma descrição dos arquivos e parâmetros utilizados na execução do sistema autotclass. Relatório Técnico 170, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_170.zip. 60, 150
- Matsubara, E. T., C. A. Martins, & M. C. Monard (2003). PreTexT — uma ferramenta para pré-processamento de textos utilizando a abordagem *bag-of-words*. Re-

- latório Técnico 209, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_209.zip. 71, 151
- Matsudo, N. (2000). Data mining - desde una perspectiva informática y estadística. Tesis de Licenciatura en Ciencias de la Computación, Universidad Nacional de Buenos Aires, Argentina. 113
- McCallum, A., K. Nigam, & L. Ungar (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining*, pp. 169–178. <http://citeseer.nj.nec.com/mccallum00efficient.html>. 26
- McCallum, A. K. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow>. 72, 133
- McLachlan, G. J. & T. Krishnan (1997). *The EM Algorithm and Extensions*. John Wiley & Sons. 24
- Melanda, E. A. (2002). Pós-processamento de conhecimento de regras de associação. Qualificação de Doutorado, ICMC-USP. 6
- Merz, C. J. & P. M. Murphy (1998). UCI repository of machine learning datasets. <http://www.ics.uci.edu/~mlearn/MLRepository.html>. 97
- Michalski, R. S., J. G. Carbonell, & T. Mitchell (1983). *Machine Learning: An Artificial Intelligence Approach*. Tioga Publishing Company. 14
- Michalski, R. S. & R. E. Stepp (1987). Clustering. *Encyclopedia of Artificial Intelligence*. John Wiley & Sons. 25, 26
- Milaré, C. R., A. C. P. L. F. Carvalho, & M. C. Monard (2002). An approach to explain neural networks using symbolic algorithms. *International Journal of Computational Intelligence and Applications – IJCIA* 2(4), 365–376. 20
- Milaré, C. R. (2003). Extração de conhecimento de redes neurais artificiais utilizando sistemas de aprendizado simbólico e algoritmos genéticos. Tese de Doutorado, ICMC-USP. 20
- Miller, G. A., R. Beckwith, C. Fellbaum, D. Gross, & K. J. Miller (1990). Introduction to wordnet: An on-line lexical database. *Journal of Lexicography* 3(4), 235–244. 43
- Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence* 18(2), 203–226. Reprinted in Shavlik and Dietterich (eds.), 1990. Readings in Machine Learning, Morgan Kaufmann Publishers, Inc. 19

- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Series in Computer Science. 13, 17
- Mladenic, D. (1998). Feature subset selection in text-learning. In *European Conference on Machine Learning*, pp. 95–100. <http://citeseer.nj.nec.com/article/mladenic98feature.html>. 44
- Monard, M. C. & J. A. Baranauskas (2003a). *Conceitos sobre Aprendizado de Máquina* (1 ed.), Chapter 4, pp. 89–114. Volume 1 of 1 Rezende (2003). 20
- Monard, M. C. & J. A. Baranauskas (2003b). *Indução de Regras e Árvores de Decisão* (1 ed.), Chapter 5, pp. 115–140. Volume 1 of 1 Rezende (2003). 21
- Montes-y-Gómez, M., A. Gelbukh, & A. López-López (2002). Detecting deviations in text collections: An approach using conceptual graphs. In *Proceedings of MICAI-2002: Mexican International Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence*, Number 2313. Springer-Verlag. <http://www.gelbukh.com/CV/Publications/2002/MICAI-2002-Graphs.htm>. 15, 35
- Mood, A., F. Graybill, & D. Boes (1974). *Introduction to the Theory of Statistics* (3 ed.). McGraw-Hill. 43
- Moulinier, I. & J.-G. Ganascia (1996). Applying an existing machine learning algorithm to text categorization. In S. Wermter, E. Riloff, & G. Scheler (Eds.), *Connectionist, statistical, and symbolic approaches to learning for natural language processing*, pp. 343–354. Springer Verlag, Heidelberg, DE. <http://citeseer.nj.nec.com/moulinier96applying.html>. 35
- Paice, C. (1996). Method for evaluation of stemming algorithms based on error counting. *Journal of the American Society for Information Science (JASIS)* 47(8), 632–649. 41
- Pardo, T., L. Rino, & M. Nunes (2003). Gistsumm: A summarization tool based on a new extractive method. In *Proceedings of the 6th Workshop on Computational Processing of the Portuguese Language - Written and Spoken*, Faro, Portugal. 32
- Peterson, R. E. (1997). Eight internet search engines compared. First Monday. http://www.firstmonday.dk/issues/issue2_2/peterson/. 34
- Pila, A. D. (2000). Seleção de atributos relevantes para aprendizado de máquina utilizando a abordagem de rough sets. Dissertação de Mestrado, ICMC-USP, <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-13022002-153921/publico/dissertacaoADP.pdf>. 44

- Porter, M. (1980). An algorithm for suffix stripping. *Program* 14(3), 130–137. 40, 41, 71, 155
- Prati, R. C. (2003). O *framework* de integração do sistema DISCOVER. Dissertação de Mestrado, ICMC-USP, <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-20082003-152116/>. 6
- Prati, R. C., J. A. Baranauskas, & M. C. Monard (2001a). Extração de informações padronizadas para a avaliação de regras induzidas por algoritmos de aprendizado de máquina simbólico. Relatório Técnico 145, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_145.ps.zip. 6
- Prati, R. C., J. A. Baranauskas, & M. C. Monard (2001b). Uma proposta de unificação da linguagem de representação de conceitos de algoritmos de aprendizado de máquina simbólicos. Relatório Técnico 137, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_137.ps.zip. 6
- Quinlan, J. R. (1986). Induction of Decision Trees. *Readings in Machine Learning*, Volume 1, pp. 81–106. Shavlik and Dietterich. 43
- Quinlan, J. R. (1988). *C4.5 Programs for Machine Learning*. CA: Morgan Kaufmann. 60, 133, 166
- Rezende, S. O. (2003). *Sistemas Inteligentes: fundamentos e aplicações* (1 ed.), Volume 1 of 1. Barueri, SP, Brasil: Editora Manole. 181, 182
- Rezende, S. O., R. B. T. Oliveira, L. C. M. Félix, & C. A. J. Rocha (1998). Visualization for knowledge discovery in database. In N. F. F. Ebecken (Ed.), *Data Mining*, England, pp. 81–95. WIT Press - Computational Mechanics Publications. 35
- Rezende, S. O., J. B. Pugliesi, E. A. Melanda, & M. F. Paula (2003). *Mineração de Dados* (1 ed.), Chapter 12, pp. 307–336. Volume 1 of 1 Rezende (2003). 2
- Rijsbergen, C. J. V., D. Harper, & M. Porter (1981). The selection of good search terms. *Information Processing and Management* 17, 77–91. 43
- Rozante, T. A. A. (2003). Implantação do reuso de componentes no processo de desenvolvimento de software. Dissertação de Mestrado, ICMC-USP. 6
- Rulequest-Research (1999). Data Mining Tools See5 and C5.0. <http://www.rulequest.com/see5-info.html>. 127

- Salton, G. & C. Buckley (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management* 24(5), 513–523. 37
- Salzberg, S. L. (1995). On comparing classifiers: a critique of current research and methods. Technical Report JHU-95/06, Department of Computer Science, Johns Hopkins University. <http://www.cs.jhu.edu/~salzberg/critique.ps>. 18
- Scott, S. & S. Matwin (1999). Feature engineering for text classification. In I. Bratko & S. Dzeroski (Eds.), *Proceedings of ICML-99, 16th International Conference on Machine Learning*, Bled, SL, pp. 379–388. Morgan Kaufmann Publishers, San Francisco, US. <http://citeseer.nj.nec.com/article/scott99feature.html>. 35
- Sebastiani, F. (2002). Machine learning in automated text categorisation. *ACM Computing Surveys* 34(1), 1–47. <http://faure.iei.pi.cnr.it/~fabrizio/Publications/ACMCS02.pdf>. 32, 48, 152
- Secato, M. & A. A. Lopes (2003). Corpus de resumos de artigos sobre raciocínio baseado em casos extraídos do *Lecture Notes in Artificial Intelligence - LNAI*. Comunicação pessoal. 142
- Silicon Graphics (2000). *MineSet™ 3.0 Enterprise Edition*. http://www.sgi.com/software/mineset/mineset_data.html. 4
- Sneath, P. H. A. & R. R. Sokal (1973). *Numerical Taxonomy*. San Francisco: Freeman Press. 27
- Stepp III, R. E. & S. Michalski (1986). Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects, *Machine Learning, An Artificial Intelligence Approach*, Volume II. Palo Alto, CA: Morgan Kaufmann. 53
- Tan, A.-H. (1999). Text mining: The state of the art and the challenges. In *Proceedings, PAKDD'99 workshop on Knowledge Discovery from Advanced Databases*, Beijing, pp. 65–70. 31
- Thomas, J. & K. Sycara (1999). Integrating genetic algorithms and text learning for financial prediction. In *Proceedings of the GECCO-2000 Workshop on Data Mining with Evolutionary Algorithms*. 35
- Van Rijsbergen, C. J. (1979). *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow. <http://citeseer.nj.nec.com/vanrijsbergen79information.html>. 34, 43, 48

- Vapnik, V. N. & A. Chervonenkis (1971). On the uniform convergence of relative frequencies of events to their probabilities. *theory of probability and its applications*. (16), 262–280. 127
- Wall, L., T. Christiansen, & R. L. Schwartz (1996). *Programming in PERL*. O’Reilly, Inc. 57, 71
- Weiss, S. M. & N. Indurkha (1998). *Predictive Data Mining: a Practical Guide*. California, USA: Morgan Kaufmann. <http://www.data-miner.com.or>. 4
- Weiss, S. M. & C. A. Kulikowski (1991). *Computer Systems that Learn*. California, USA: Morgan Kaufmann. 14
- Witten, I. H. & E. Frank (2000). *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*, Volume 1. Morgan Kaufmann. <http://www.cs.waikato.ac.nz/ml/weka/index.html>. 5, 22
- Yang, J. & W. Wang (2003). Cluseq: efficient and effective sequence clustering. In *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE)*. (to appear). 32
- Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval* 1(1/2), 69–90. <http://citeseer.nj.nec.com/yang97evaluation.html>. 44, 49
- Yang, Y. & X. Liu (1999). A re-examination of text categorization methods. In M. A. Hearst, F. Gey, & R. Tong (Eds.), *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, Berkeley, US, pp. 42–49. ACM Press, New York, US. <http://citeseer.nj.nec.com/yang99reexamination.html>. 35
- Yang, Y. & J. O. Pedersen (1997). A comparative study on feature selection in text categorization. In D. H. Fisher (Ed.), *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Nashville, US, pp. 412–420. Morgan Kaufmann Publishers, San Francisco, US. <http://citeseer.nj.nec.com/yang97comparative.html>. 40, 44
- Zipf, G. (1949). *Human Behaviour and the Principle of Least Effort*. Addison-Wesley. 42

¹As *url* contidas nas referências bibliográficas foram acessadas durante o desenvolvimento dessa tese.