Clustering de artigos científicos em uma Ferramenta Inteligente de apoio à Pesquisa

Vinícius Veloso de Melo

Orientador: ***Prof. Dr. Alneu de Andrade Lopes***

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

USP – São Carlos
Maio/2005

Aluno: Vinícius Veloso de Melo

A Comissão Julgadora:

Prof. Dr. Alneu de Andrade Lopes _____

Profa. Dra. Maria Carolina Monard _____

Prof. Dr. Nivio Ziviani _____

Este documento foi elaborado com o formatador de textos LaTeX, e as referências bibliográficas foram geradas automaticamente pelo BibTeX utilizando o formato padrão Chicago.

A versão colorida deste documento estará disponível online no site www.saber.usp.br

Agradecimentos

À Deus, por permitir que eu chegasse até aqui.

Aos meus pais, por terem me concebido, me educado, me dado amor e carinho.

Aos meus amigos e colegas de Mestrado, com quem passei momentos de descontração para aliviar um pouco a tensão.

Ao pessoal do LABIC, que me ajudou sempre que precisei.

Aos docentes do ICMC e aos integrantes da minha banca de Qualificação, que deram sugestões importantes para a realização deste trabalho de Mestrado.

Ao meu orientador, Prof. Dr. Alneu de Andrade Lopes, cuja orientação foi de substancial importância para a conclusão deste trabalho.

Aos funcionários do ICMC, pela atenção.

À Prof^{a}. Dr^{a}. Agma, por me fornecer o código da *Slim-Tree* e ao Marcos Rodrigues Vieira, por me ajudar com algumas modificações no código.

Ao Stephen Bickel, por me fornecer o código do *Multi-View Spherical K-Means* e por me ajudar nos experimentos.

Ao CNPQ, pelo suporte financeiro essencial a este trabalho.

Resumo

Com a popularização da Internet, a disponibilização online de documentos de qualquer espécie tornou-se extremamente rápida. Utilizando-se de ferramentas de busca, pode-se ter acesso a quase todos os tipos de informação em questão de segundos. Porém, a quantidade de sites que proporcionam alguma informação importante é, em geral, muito pequena, se comparada ao número total de páginas que é fornecido pela ferramenta de busca. Isso ocorre, basicamente, pelo fato de que as páginas retornadas são ordenadas, por exemplo, de acordo com a quantidade de acessos à página ou à quantidade de links que levam a ela. Isso significa que uma página contendo a informação que o usuário deseja, mas que esteja no final da lista, dificilmente será lida se existir uma grande quantidade de páginas antes dela. Assim, seria de grande ajuda uma ferramenta capaz de: a) recuperar um conjunto apropriado de documentos de acordo com palavras-chave fornecidas pelo usuário; b) analisar o conteúdo dos links encontrados extraindo informações relevantes dos textos e decidir se o documento pode ser importante para o usuário; c) fazer um *clustering* (agrupamento por similaridade) desses documentos relevantes e d) exibir um mapa no qual documentos similares estejam próximos entre si e distantes daqueles relacionados com outra área. Essa ferramenta está sendo desenvolvida no LABIC/ICMC-USP e recebeu o nome de FIP (Ferramenta Inteligente de Apoio à Pesquisa). Este trabalho visa investigar técnicas de *clustering*, principalmente, as aplicadas a documentos e decidir por aquela que melhor atenda os requisitos da FIP em termos de qualidade dos clusters, tempo de processamento e consumo de memória, visto que é tratada uma grande quantidade de documentos na ferramenta. Neste trabalho são testadas técnicas de *clustering* aglomerativo hierárquico, de particionamento e de mapa auto-organizável em córpus de artigos científicos, jornalísticos e de fóruns de discussão; são discutidas as vantagens e desvantagens de cada uma; e indicadas, no caso particular da ferramenta FIP, as abordagens apropriadas.

Abstract

With the Internet popularization, the online deployment of any kind of document has become extremely fast. By using of searching tools, access to almost any kinds of information can be done in a matter of seconds. However, the amount of sites that provide some useful information is, in general, very small, if compared to the total number of pages supplied by the search tool. That happens, basically, by the fact that the retrieved pages are ranked, in general, in accordance with the amount of accesses to the page and/or the amount of links that point to them. That means, a page containing the information that the user desires, but at the end of a huge list, hardly will be seen. Thus, it would be very useful a tool capable of: a) to retrieve an appropriate set of document in accordance with keywords supplied by the user; b) to analyze the content of those documents, extracting relevant information from the texts and to decide if the document is relevant for the user; c) to group (by similarity) those documents; and d) to exhibit a map, in which similar documents are close amongst themselves and distant of those related with other fields. Such a tool is being developed at LABIC/ICMC-USP and has received the name FIP (Ferramenta Inteligente de Apoio à Pesquisa). This work, part of the FIP project, seeks to investigate clustering techniques, mainly, those applied to text, and to decide for the one that best fit to the requirements of FIP in terms of clusters quality, processing time and memory consumption. Those issues are relevant because the tool will deal with a great amount of documents. In this work we test agglomerative hierarchical clustering techniques, partitioning techniques, and self-organizing maps techniques in corpora of scientific articles, journalistic, and discussion forums. We also discuss the advantages and disadvantages of each technique; and indicate the appropriate approaches in the particular case of the FIP tool.

Sumário

Lista de Figuras xvii

Lista de Tabelas xxi

Lista de Abreviaturas xxiv

1 Introdução **1**

1.1 Motivação e Escopo do Trabalho 3

1.2 Organização do Trabalho 10

2 *Clustering* **11**

2.1 Estrutura do Processo de Clustering 11

2.2 Síntese do Processo de Clustering 13

2.2.1 Tratamento dos Dados 14

2.2.2 Métrica de Similaridade 15

2.2.3 Algoritmos de Clustering 17

2.2.4 Avaliação dos Resultados 20

2.3 Considerações Finais 25

3 Clustering de Documentos **27**

3.1 Representação de Documentos 28

3.1.1 Bag-of-Words 28

3.1.2 Bag-of-References 31
3.2 Redução de dimensionalidade 35
3.2.1 Cortes de Luhn 36
3.2.2 Mapa Semântico de Palavras 37
3.2.3 Thesaurus 37
3.2.4 Indexação por Semântica Latente 38
3.2.5 Fastmap 38
3.3 Métricas de Similaridade entre Documentos 39
3.3.1 A distância de Minkowski 39
3.3.2 Medida do Co-seno 41
3.3.3 Similaridade Estendida de Jaccard 41
3.3.4 Atributos Qualitativos 43
3.4 Considerações Finais 44

4 Algoritmos de *Clustering* 47

4.1 *Clustering* Hierárquico 47
4.2 Algoritmos de Particionamento 52
4.2.1 K-Means 53
4.2.2 Bi-Secting K-Means 56
4.2.3 Spherical K-Means 57
4.3 Algoritmos Multi-Visão 57
4.3.1 Multi-View Agglomerative Clustering 58
4.3.2 Multi-View Spherical K-Means 58
4.4 Redes Neurais Artificiais 59
4.4.1 Mapas Auto-Organizáveis (SOM) 60
4.4.2 *Growing Hierarchical Self-Organizing Map* (GHSOM) 66

4.5 Considerações Finais 68

5 Experimentos e Resultados **69**

5.1 Descrição Geral dos Experimentos 69

5.1.1 Ferramentas 69

5.1.2 Pré-Processamento dos Conjuntos de Dados Textuais 78

5.1.3 Córpus e *Datasets* 79

5.2 Experimentos Realizados 82

5.2.1 Experimento 1: Córpus 20 *Newsgroups* 84

5.2.2 Experimento 2: Córpus WebKB 92

5.2.3 Experimento 3: Córpus LNAI/*WebMiner* 100

5.2.4 Experimento 4: *Datasets* do CLUTO 113

5.3 Discussão sobre os Experimentos 119

6 Conclusões e Trabalhos Futuros **123**

6.1 Contribuições do Trabalho 125

6.2 Sugestões de Trabalhos Futuros 127

Lista de Figuras

1.1 Um possível mapa da área de RBC 5

1.2 Técnicas fundamentais dos módulos da FIP 6

1.3 Representação da arquitetura básica da Ferramenta Inteligente de Apoio à Pesquisa. 8

2.1 Representação do processo de *clustering*. 12

2.2 Taxonomia dos Métodos de *Clustering* 17

3.1 Diferentes formatos, em diferentes artigos, para uma mesma referência bibliográfica. 32

3.2 Ilustração representando a curva de Zipf (I) e os cortes de Luhn (II) (Matsubara et al., 2003). 36

3.3 Exemplo de Mapa de Categorias de Palavras, de tamanho 15x21 nodos. 37

3.4 Gráfico representando a similaridade euclideana. 40

3.5 Gráfico representando a similaridade com a medida do co-seno. 41

3.6 Gráfico representando a similaridade com coeficiente estendido de Jaccard. 43

4.1 Dendrograma (Método das Médias das Distâncias). 50

4.2 *Clustering* com *Single-link*. 51

4.3 *Clustering* com *Complete-link*. 51

4.4 Funcionamento do algoritmo *K-Means*. 55

4.5 Estrutura de uma rede do tipo SOM. 61

4.6 Tipos de vizinhança em uma rede do tipo SOM. 62

4.7 Treinamento de uma rede SOM. 63

4.8 Arquitetura básica do WEBSOM. 65

4.9 Exemplo do WEBSOM. 66

4.10 Arquitetura de uma GHSOM treinada. 67

5.1 Gráfico dos resultados da execução de algoritmos de *clustering* no córpus 20 *Newsgroups* (funções *e1* até *i2*). 85

5.2 Gráfico dos resultados da execução de algoritmos de *clustering* no córpus 20 *Newsgroups* (funções *clink* até *wupgma*). 85

5.3 Gráfico de 100 execuções do *Spherical K-Means* no córpus 20 *Newsgroups*. 88

5.4 Mapa 3D do córpus 20 *Newsgroups*. 89

5.5 Mapa do córpus 20 *Newsgroups* gerado usando SOM. 90

5.6 mapa do córpus 20 *Newsgroups* gerado usando GHSOM. 91

5.7 Gráfico dos resultados da execução de algoritmos de *clustering* no córpus WebKB (funções *e1* até *i2*). 93

5.8 Gráfico dos resultados da execução de algoritmos de *clustering* no córpus WebKB (funções *clink* até *wupgma*). 93

5.9 Gráfico de 100 execuções do *Spherical K-Means* no córpus WebKB. . . 95

5.10 mapa 3D do córpus WebKB. 96

5.11 Mapa 3D do córpus WebKB - 10 clusters. 97

5.12 Mapa do córpus WebKB gerado pela ferramenta Som_Pak. 98

5.13 Mapa do córpus WebKB de dimensão 8x6. 98

5.14 Mapa do córpus WebKb gerado pelo GHSOM. 99

5.15 Gráfico dos resultados da execução de algoritmos de *clustering* no córpus LNAI/ *WebMiner* (funções *e1* até *i2*). 102

5.16 Gráfico dos resultados da execução de algoritmos de *clustering* no córpus LNAI/ *WebMiner* (funções *clink* até *wupgma*). 103

5.17 Gráfico de 100 execuções do *Spherical K-Means* no córpus LNAI/ *WebMiner*. 105

5.18 Mapa 3D do córpus LNAI/ *WebMiner* (*bag-of-words*). 108

5.19 Mapa 3D do córpus LNAI/ *WebMiner* (*bag-of-words*) - 7 clusters. 109

5.20 Mapa 3D do córpus LNAI/ *WebMiner* (*bag-of-references*) - 7 clusters. . 111

5.21 Mapa do córpus LNAI/ *WebMiner* gerado pela ferramenta Som_Pak. . 112

5.22 Mapa do córpus LNAI/ *WebMiner* gerado pelo GHSOM. 113

Lista de Tabelas

2.1 Dados Iniciais 15

2.2 Dados Padronizados 15

2.3 Matriz de similaridade (distância) 16

2.4 Matriz de Confusão de Classificação Binária. 21

2.5 Taxas de erro calculadas a partir da Matriz de Confusão. 21

2.6 Matriz de Confusão Multi-Valorada. 22

2.7 Matriz de Confusão para número diferente de classes e clusters. 22

3.1 Representação de documentos em tabela de peso binário. 29

3.2 Representação de documentos em tabela de peso por freqüência. 29

3.3 Representação de documentos em tabela de peso por *tf-idf*. 30

3.4 *Bag-of-words* e *Bag-of-references* 34

3.5 Tabela de valores binários. 43

3.6 Tabela computando número de pares. 44

5.1 Funções de otimização oferecidas pelo CLUTO. 71

5.2 Configuração do SOM 76

5.3 Configuração do GHSOM. 78

5.4 *Datasets* utilizados nos experimentos. 81

5.5 Pureza e Entropia do córpus 20 *Newsgroups* usando CLUTO. 84

5.6 Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus 20 *Newsgroups*. 86

5.7 Matriz de confusão (Agglo - slink) 86

5.8 Matriz de confusão (RBR - $i2$) 87

5.9 Melhores Entropia e Pureza do córpus 20 *Newsgroups* usando CLUTO e Entropia e Pureza médias *Spherical K-Means*. 88

5.10 Pureza e Entropia do córpus WebKB usando CLUTO. 92

5.11 Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus WebKB. 94

5.12 Matriz de confusão (Agglo - *slink*). 94

5.13 Matriz de confusão (Agglo - $i1$). 94

5.14 Melhores Entropia e Pureza do córpus WebKB usando CLUTO e Entropia e Pureza médias *Spherical K-Means*. 95

5.15 Medida de Pureza média dos Clusters. 101

5.16 Pureza e Entropia do córpus LNAI/ *WebMiner* usando CLUTO. 102

5.17 Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus LNAI/ *WebMiner*. 103

5.18 Medida de Pureza média dos Clusters. 104

5.19 Matrizes de Confusão. 105

5.20 Medida de Pureza média e Desvio Padrão dos Clusters usando *o Multi-view Spherical K-Means*. 106

5.21 Melhores Entropia e Pureza do córpus LNAI/ *WebMiner* usando CLUTO e Entropia e Pureza médias do *Spherical K-Means*. 107

5.22 Termos mais discriminativos da *bag-of-references* com 3 clusters. 108

5.23 Referências dos termos mais discriminativos da *bag-of-references* com 3 clusters. 110

5.24 Pureza dos *datasets CLASSIC* e FBIS. 115

5.25 Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus *Classic*. 116

5.26 Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus FBIS. 117

5.27 Pureza dos *datasets* Reuters e WAP. 118

5.28 Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus *Reuters*. 118

5.29 Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus WAP. 119

5.30 Melhores resultados de pureza e entropia médias nos *datasets* do CLUTO. 119

5.31 Melhores resultados de pureza e entropia médias nos *datasets* do CLUTO, com valores de k (clusters) diferentes do número de classes. 120

5.32 Melhores resultados de pureza e entropia médias em todos os experimentos. 120

Lista de Abreviaturas

AM Aprendizagem de Máquina

CHA *Clustering* Hierárquico Aglomerativo

FIP Ferramenta Inteligente de Apoio à Pesquisa

GHSOM *Growing Hierarchical Self-Organizing Map*

IA Inteligência Artificial

KDD Descoberta de Conhecimento em Banco de Dados (*Knowledge Discovery in Data Base*)

LSI Indexação por Semântica Latente*Latent Semantic Indexing*

MD Mineração de Dados

MMD Método das Médias das Distâncias

MT Mineração de Textos

PLI Programação Lógica Indutiva

PLN Processamento de Língua Natural

RBC Raciocínio Baseado em Casos

RI Recuperação de Informação

RNA Redes Neurais Artificiais

SOM *Self-Organizing Map*

Capítulo 1

Introdução

Um dos problemas com que iniciantes, e mesmo pesquisadores mais experientes, se deparam atualmente é com a quantidade de informações disponível, sobre qualquer área de conhecimento, principalmente na web. Cada vez mais, os principais artigos científicos são disponibilizados em sites pessoais ou institucionais. Entretanto, continua sendo difícil encontrar informações específicas sobre determinado assunto, devido às limitações da grande maioria das páginas (uso de código html sem informações semânticas sobre o conteúdo da página) e da própria diversidade e quantidade de material disponível. Além desse fator, na web, juntamente com informações relevantes, existe uma quantidade enorme de material de menor importância. Portanto, a "mineração" (recuperação e seleção) desse material relevante passa pelo conhecimento que o pesquisador tem daquele campo de conhecimento e é, conseqüentemente, difícil para novatos em uma área de pesquisa.

Atualmente é dedicado um esforço das comunidades de Inteligência Artificial (IA) e de Processamento de Língua Natural (PLN) em processos de automação de Mineração de Textos. Mineração de Textos é o processo de encontrar padrões interessantes e úteis em um córpus não estruturado ou semi estruturado de informações textuais. Mineração de Textos combina várias técnicas de outras áres de pesquisa como Extração de Informação, Recuperação de Informação, Processamento de Língua Natural, Sumarização, Categorização, *Clustering*, Visualização, entre outras. Análises estatísticas

dos textos também permitem encontrar trechos importantes, determinar similaridade entre documentos, entre outras informações. A intenção é encontrar conhecimento nos textos de maneira automática. Conhecimento este que pode ser difícil para o usuário, tratando-se de uma grande quantidade de documentos.

Portanto, diversas técnicas de IA e PLN têm sido desenvolvidas e aplicadas no tratamento de textos. Neste trabalho, discute-se detalhadamente o uso de técnicas de *clustering* para organizar os textos em diversas áreas e sub-áreas. O objetivo desta técnica é agrupar objetos (ou exemplos) baseado nas características que esses objetos possuem, de tal maneira que seja maximizada a similaridade entre objetos pertencentes a um mesmo agrupamento e minimizada a similaridade entre os agrupamentos. Ou seja, procura-se a homogeneidade dos objetos dentro de um mesmo cluster enquanto se maximiza a diferença entre os distintos clusters.

A técnica de *clustering* é classificada como um método de aprendizagem não supervisionada, isto é, no conjunto de exemplos (objetos) considerado, não é informado um atributo especial (classe) na descrição do objeto. Portanto, os objetos que serão considerados pertencentes a um cluster (agrupamento), bem como o número total de clusters, serão dependentes dos critérios e procedimentos adotados para se avaliar a similaridade entre esses objetos. De acordo com esses procedimentos, muitas soluções (diferentes conjuntos de clusters) podem ser obtidas para um mesmo conjunto de dados.

A técnica de *clustering* em documentos foi investigada para melhorar a precisão em sistemas de recuperação de informação (van Rijsbergen, 1979; Kowalski, 1997) e como uma maneira eficiente de resolução do problema de se encontrar os vizinhos mais próximos de um documento (Buckley & Lewit, 1985). Além disso, *clustering* tem sido empregado em:

1. navegação em uma coleção de documentos(Cutting et al., 1992);
2. organização dos resultados de uma máquina de busca (Zamir & Etzioni, 1998);
3. geração automática de *clusters* hierárquicos de documentos (Koller & Sahami, 1997).

Atualmente existem disponíveis diversas técnicas de *clustering*. O *Clustering* Hierárquico Aglomerativo (CHA) e o *K-Means* (Macqueen, 1967; Hartigan & Wong, 1979)

são as técnicas mais conhecidas e utilizadas para *clustering* de documentos. As técnicas de CHA são caracterizadas de acordo com o método utilizado para definir as distâncias entre grupos. Os métodos mais conhecidos são *Single-link* (Sneath & Sokal, 1973), *Complete-link* (King, 1967), *Group average-link* (Voorhees, 1986) e *Minimum-variance* (Ward, 1963; Murtagh, 1984). Outra técnica utilizada com sucesso é o Mapa Auto-Organizável (SOM, do inglês *Self-Organizing Map*) (Kohonen, 1990; Kohonen et al., 1996; Kohonen, 2001). SOM é um método eficiente de agrupar dados de alta dimensão de modo que as entradas semelhantes sejam mapeadas proximamente. No mapa resultante, pode-se facilmente visualizar as relações entre objetos diferentes em função de distâncias entre eles, de uma maneira intuitiva (Lin et al., 1991; Kaski et al., 1996). Essa técnica foi empregada no WEBSOM (Honkela et al., 1996) gerando mapas com boa divisão dos documentos, permitindo uma fácil identificação das áreas.

Para realizar o *clustering* dos documentos é necessária uma etapa de pré-processamento nos textos. Este pré-processamento é composto de diversas tarefas com objetivo de possibilitar o uso das técnicas de Aprendizagem de Máquina (AM) para mineração de textos. As tarefas incluem: remoção de palavras que não são consideradas importantes na discriminação do conteúdo do texto (*stopwords*); eliminação de informações extras nas palavras restantes, como tempo verbal, gênero e número, reduzindo a palavra a um radical (técnica de *stemming*); atribuição de um peso a cada um dos termos; e conversão para uma representação vetorial, a partir da qual pode ser aplicada alguma das técnicas de AM.

A avaliação dos resultados é feita utilizando medidas indicadas pela literatura: entropia, pureza, além da análise da performance do algoritmo (tempo de execução e consumo de memória). Uma característica importante na avaliação de algoritmos hierárquicos é que sua aplicação, no conjunto de documentos, permite que sejam identificadas áreas e sub-áreas de acordo com os níveis de hierarquia.

Na próxima seção é apresentada brevemente a proposta da FIP, que é a motivação para este trabalho.

1.1 Motivação e Escopo do Trabalho

Um modo de auxiliar um pesquisador a enfrentar o problema da grande quantidade de material relevente e irrelevante na web é disponibilizar uma ferramenta que o ajude nessa mineração por informações importantes sobre a área ou tema de seu interesse. Um exemplo de informação útil para um aluno de iniciação científica ou mestrado, por exemplo, pode ser uma lista das publicações mais relevantes relacionadas com o tema de sua pesquisa. Essa lista, porém, nem sempre está disponível e com o desenvolvimento acelerado de determinadas áreas, quando existe, nem sempre está atualizada. Assim, é de significativo interesse um sistema que, automaticamente, busque na web artigos e publicações sobre um tema escolhido pelo usuário e organize, por exemplo, uma ontologia ou hierarquia dos principais conhecimentos na vizinhança desse tema, indicando ao usuário quais são as publicações mais importantes, quais são os temas relacionados, quais são as referências mais citadas, entre outras informações. Tal sistema pode apresentar um "mapa do terreno" a ser explorado pelo pesquisador.

Uma ferramenta com o objetivo de construir o mapa acima mencionado necessita fundamentalmente de três módulos principais: 1- um módulo de recuperação de informação (basicamente recuperação de artigos disponíveis na web e do armazenamento desses em um banco de dados relacional ou em um *data warehouse*); 2- um módulo de mineração, propriamente, para análise e avaliação das similaridades e relações entre os artigos, tais como precedência, importância, áreas e sub-áreas às quais pertence; 3- um módulo interação com o usuário e de visualização. Essa ferramenta foi denominada FIP (Ferramenta Inteligente de Apoio à Pesquisa) e está em desenvolvimento no LABIC/ICMC-USP (Melo & Lopes, 2004b; Brasil & Lopes, 2004; Melo & Lopes, 2005).

Uma Ferramenta Inteligente de Apoio à Pesquisa

O projeto de uma ferramenta inteligente de apoio à pesquisa (FIP) tem como objetivo disponibilizar uma ferramenta que auxilie um estudante ou pesquisador iniciante em uma área, a construir um mapa dessa área ou tópico de pesquisa. Esse mapa deve permitir a visualização das principais áreas, sub-áreas, artigos e autores principais, além de outros aspectos importantes relacionados com aquele tema/tópico

de pesquisa.

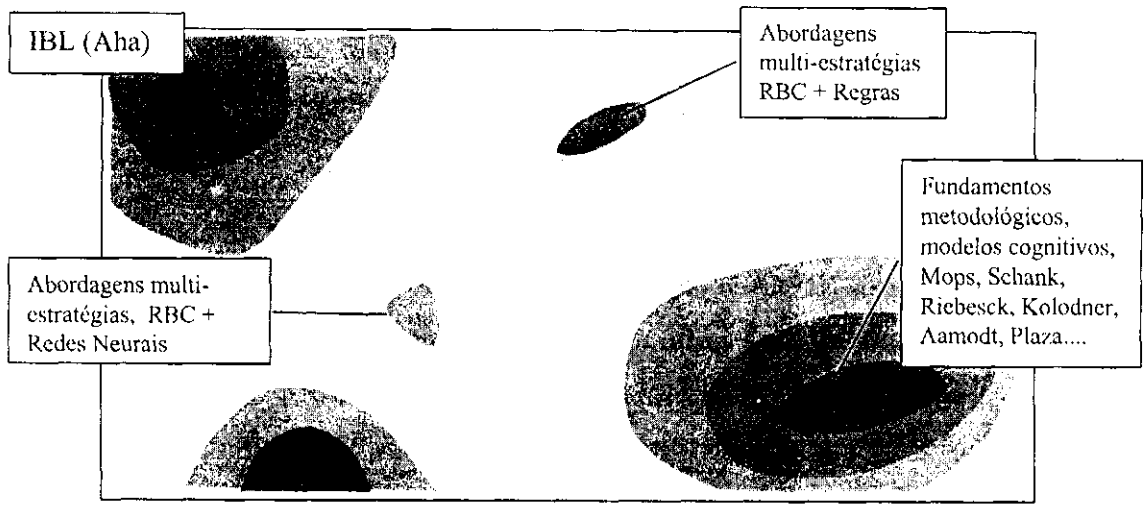
A informação de entrada para a construção do mapa é um conjunto de artigos recuperados da web relacionados com o tema proposto pelo usuário, armazenados no repositório da FIP. Um possível exemplo de resultado de uma pesquisa usando a FIP é ilustrado a seguir.

Supondo-se que o usuário da ferramenta inicie suas pesquisas na área de Raciocínio Baseado em Casos (RBC), o objetivo principal da ferramenta é conseguir construir um mapa semelhante ao da Figura 1.1. Após recuperar os artigos relacionados com o tópico proposto pelo usuário, a ferramenta deve selecionar os principais artigos, determinar relações de similaridade entre eles, determinar áreas e sub-áreas (estabelecer agrupamentos - clusters - mais importantes) e, finalmente, construir um mapa com essas informações. Vale citar que outras técnicas de redução de dimensionalidade e de projeção podem ser usadas. Porém, neste trabalho são avaliadas apenas as técnicas de *clustering*.

O mapa representado na Figura 1.1 é uma possibilidade de representação gráfica simplificada do conhecimento que um orientador ou especialista na área poderia passar para o aluno. Nesse mapa as curvas de contorno representam agrupamento (*clusters*) de artigos relacionados (similares), publicados em uma mesma época. As áreas mais escuras representam artigos pioneiros, que são referências fundamentais dentro de área e sub-áreas. O mapa pode ser visto como curvas de níveis de uma superfície onde os cumes representam publicações fundamentais e os montes os trabalhos relacionados que vieram a seguir. Uma maneira de detectar essas publicações mais relevantes pode ser em função da quantidade de vezes que são referenciados em outros artigos, importância do congresso, revista ou periódico em que foram publicadas, etc. Vale ressaltar que diversos outros critérios e pesos para as cotas podem ser usadas além de cores, sons e aura para facilitar a exploração do mapa. A idéia é construir esse mapa utilizando-se das informações de artigos e publicações, no caso do exemplo, da área de RBC, e de um conjunto de dados sobre cada publicação.

Para construção de tal mapa, um ponto fundamental é estabelecer a similaridade entre os artigos, além da importância relativa entre eles, precedência e referências comuns. Por meio desses dados, pode-se representar *clusters* que possivelmente estão relacionados com temas, abordagens ou sub-áreas dentro do tema de pesquisa selecio-

Figura 1.1: Um possível mapa da área de RBC



nado.

Uma vantagem da representação gráfica está na facilidade de se perceber os pesos e relações entre as diversas abordagens e sub-áreas, além da visualização das fronteiras e intersecções com outros campos. Naturalmente, outras representações poderão ser avaliadas, no momento em que dados reais forem coletados, juntamente com indicadores e filtros que permitam diferentes interpretações do mapa.

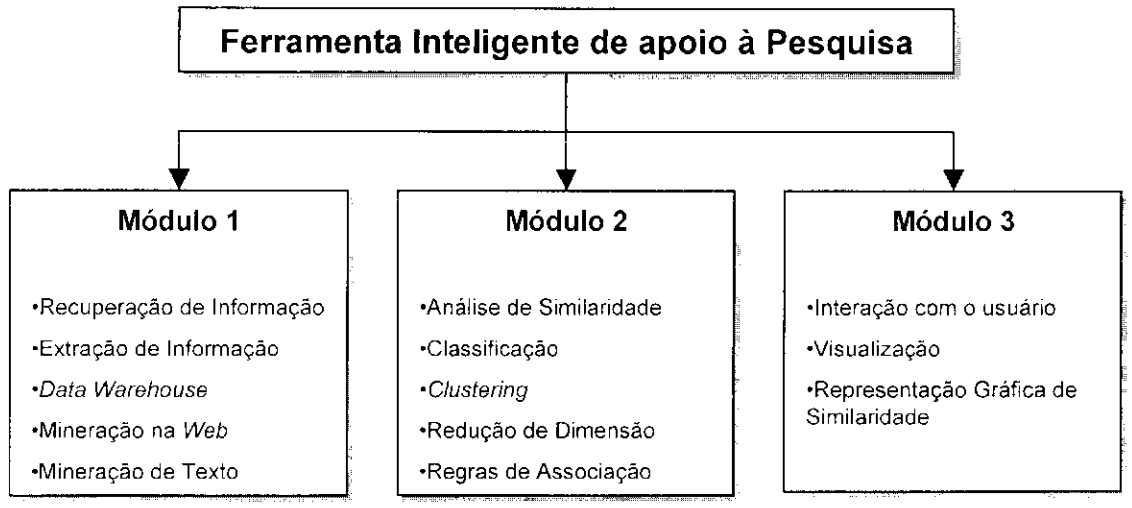
A ferramenta é composta, atualmente, por três módulos fundamentais (em desenvolvimento) relacionados com a recuperação de artigos científicos na web, a análise dos artigos (determinação da similaridade entre os artigos) e a visualização (construção do mapa) dos artigos recuperados (Figura 1.2).

Módulo 1: responsável pela recuperação de artigos científicos disponíveis na web por meio de técnicas de mineração na web. Dos artigos recuperados serão extraídas informações como: título, palavras-chave, resumo e conjunto de referências bibliográficas. Estas informações serão armazenadas em um repositório da FIP.

Módulo 2: trata da avaliação e análise da similaridade entre os artigos recuperados, além de encontrar relações de precedência, importância relativa, áreas e sub-áreas relacionadas com os documentos, por meio de técnicas como *clustering*, regras de associação e outras técnicas de Aprendizagem de Máquina.

Módulo 3: realiza a representação gráfica do resultado do módulo 2, ou seja,

Figura 1.2: Técnicas fundamentais dos módulos da FIP



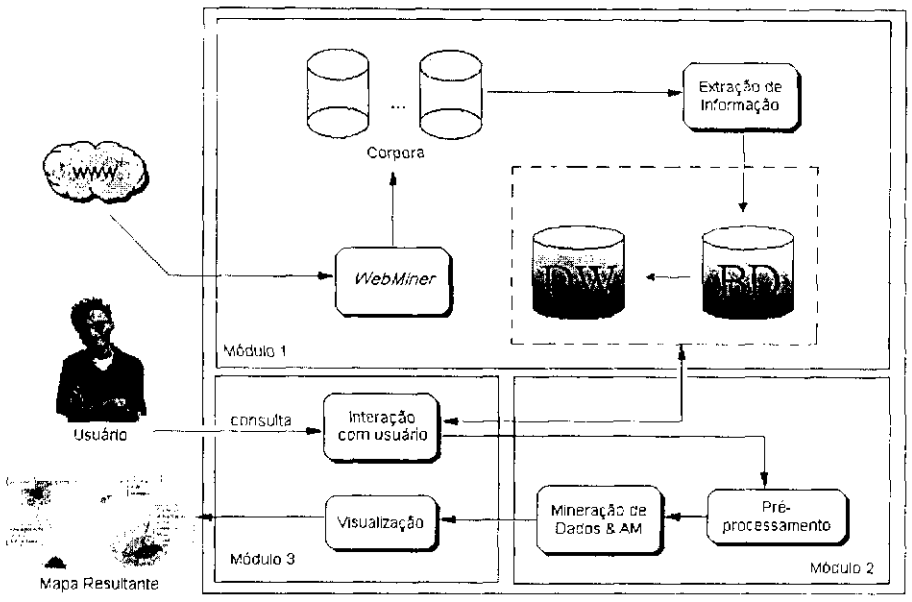
criação de um mapa que represente os níveis de similaridade entre os artigos científicos. Este módulo também é responsável pela interação com o usuário.

Entre as principais técnicas relacionadas com a FIP, destacam-se as técnicas de Aprendizagem de Máquina, de Mineração de Dados - (MD) (Han & Kamber, 2000; Berkhin, 2002), de Mineração de Textos - (MT) (Feldman et al., 1999; Visa, 2001), de Descoberta de Conhecimento em Banco de Dados - *Knowledge Discovery in Data Base* (KDD), de Mineração na *Web* - (MW) (Berendt et al., 2004) e de Recuperação de Informação - (RI) (van Rijsbergen, 1979; Frakes & Baeza-Yates, 1992; Kowalski, 1997).

Este trabalho de Mestrado está relacionado com o módulo 2. Porém, limitado à investigação das técnicas de *clustering* e decisão por aquela que melhor atenda os requisitos da FIP em termos de qualidade dos clusters, tempo de processamento e consumo de memória, visto que será tratada uma grande quantidade de documentos. Como os documentos a serem recuperados pela FIP são artigos científicos, pode-se, ainda, usar informações extras que ajudem no agrupamento dos documentos, como suas referências bibliográficas (Melo et al., 2003; Melo & Lopes, 2004a; Melo & Lopes, 2004b; Melo & Lopes, 2005), ou que ajudem o usuário a entender o conteúdo dos *clusters*.

Para treinamento e teste na FIP, foram construídos corpora de Raciocínio Baseado em Casos (RBC) e Programação Lógica Indutiva (PLI), por meio da digitalização (via

Figura 1.3: Representação da arquitetura básica da Ferramenta Inteligente de Apoio à Pesquisa.



scanner) e aplicação de uma ferramenta de *OCR*[1], de artigos publicados na *Lecture Notes on Artificial Intelligence*. Infelizmente, na conversão para documento digital, a ferramenta de *OCR* acaba acrescentando erros e nem todos são corrigidos. Foi também construído um córpus de Recuperação de Informação a partir de documentos recuperados da *web*. Atualmente o corpora possui 277 artigos sobre RBC, 118 artigos sobre PLI e 205 artigos sobre RI. Estes domínios foram escolhidos pelo fato de o ICMC possuir especialistas na área que podem ajudar na interpretação dos resultados. Nesta etapa do desenvolvimento também optou-se por considerar apenas publicações em inglês.

A Figura 1.3 representa a arquitetura básica da FIP e seu funcionamento é descrito a seguir:

1. **atualização dos corpora:** o sistema *WebMiner* atualizará de maneira automática e periódica o conteúdo dos corpora, sendo que, inicialmente, conterá apenas artigos das áreas de RBC, PLI e RI mas, futuramente, abrangerá outras áreas de Inteligência Artificial e Ciência da Computação. O processo é automatizado por técnicas de recuperação de informação (para recuperar os artigos PDF e PS) e mineração de texto (para determinar a relevância do artigo recuperado de acordo com o tema escolhido).

[1] *Optical Character Recognition*

2. **extração de informação:** informações como título, autores, resumo, referências, palavras-chave, serão extraídas dos artigos recuperados e armazenadas em um banco de dados.

3. **pré-processamento:** esse pré-processamento acessa os documentos da base de dados e realiza identificação de palavras-chave, remoção de *stopwords*, *stemming*, criação do ranking dos artigos ou autores mais citados. Estes dados são transformados para uma representação vetorial que possa ser tratada por algoritmos de AM e, no caso deste trabalho, pelos algoritmos de *clustering*.

4. **mineração de dados, AM e visualização:** os dados serão analisados e organizados, a fim de se prover uma representação visual dos artigos científicos ao usuário (mapa), bem como *links* com os títulos dos artigos, para que se possa ter acesso ao documento PDF/PS.

Atualmente, além deste aluno, três outros alunos de Mestrado trabalham no projeto FIP.

- Uma aluna está desenvolvendo a ferramenta de busca inteligente (*WebMiner*) para recuperar da web os artigos científicos que sejam relevantes para uma determinada área, fazendo uso das técnicas descritas para o módulo 1. Para tal, usa-se classificadores para gerar regras capazes de identificar um documento específico. Os classificadores foram treinados com o corpora da FIP, descritos anteriormente, e os resultados foram considerados bons, atingindo uma precisão de 98% (Brasil & Lopes, 2004).

- Um aluno está utilizando técnicas de PLI para induzir regras capazes de extrair informações dos artigos recuperados pelo *WebMiner*.

- Outro aluno está trabalhando na criação do *data warehouse* para fazer o armazenamento dos documentos recuperados pelo *WebMiner*. Além dos documentos completos serão armazenadas informações extraídas dos textos como autores, resumo, referências, etc. Com essas informações será possível mapear áreas, realizar ranqueamento de eventos, autores e artigos, de modo a facilitar o trabalho do pesquisador.

Ainda, o trabalho de Iniciação Científica realizado pelo aluno Marcus Secato envolveu a criação dos córpus de RBC e PLI e a implementação de um *parser* capaz de extrair as informações citadas no item 2 exposto anteriormente (Secato & Lopes, 2004). Um ex-aluno de mestrado, iniciou um trabalho sobre análise da similaridade entre artigos científicos, trabalhando também com visualização, utilizando técnicas para reduzir a dimensionalidade dos dados (*Latent Semantic Indexing*, seção 3.2.4) de modo que pudessem ser desenhados na tela e, por meio de um *clustering* hierárquico, construir mapas de similaridades entre pares de documentos (Fodra & Lopes, 2004).

1.2 Organização do Trabalho

Após essa contextualização do problema, o restante do documento está divido em duas partes principais, descritas a seguir.

1. Técnicas para resolução do problema de agrupamento de artigos científicos: no capítulo 2 é fornecida uma visão geral do uso da técnica de *clustering*, seu funcionamento, algumas métricas e métodos de avaliação. No capítulo 3 são tratadas as técnicas de *clustering* quando aplicadas a documentos, explicando sobre a representação dos documentos, bem como o uso das referências bibliográficas e sua representação. Também são detalhadas as métricas de similaridade que são usadas nesse tipo de problema. No capítulo 4 são apresentadas algumas das técnicas de *clustering* mais comuns na literatura da área.

2. Experimentos realizados: no capítulo 5 são expostos os experimentos e comentados os resultados. No capítulo 6 são apresentadas as considerações gerais sobre este trabalho, comentadas suas principais contribuições e possibilidades de trabalhos futuros.

CAPÍTULO
2

Clustering

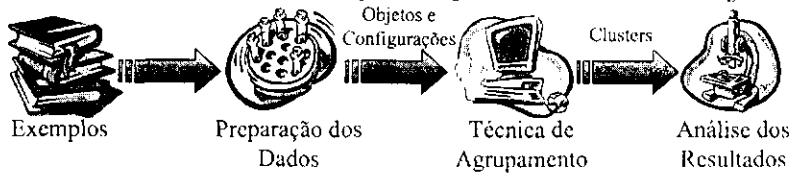
Neste capítulo é apresentada uma visão geral sobre a técnica de *clustering*, aplicada a dados que não sejam necessariamente documentos. A técnica para *clustering* de documentos é apresentada no capítulo seguinte, juntamente com algumas características particulares para tratamento de texto.

O objetivo da tarefa de *clustering* é agrupar objetos (ou exemplos) baseado nas características que esses objetos possuem, de tal maneira que seja maximizada a similaridade entre objetos pertencentes a um mesmo agrupamento e minimizada a similaridade entre agrupamentos distintos. Ou seja, procura-se a homogeneidade dos objetos dentro de um mesmo agrupamento enquanto maximiza-se a diferença entre os agrupamentos.

2.1 Estrutura do Processo de Clustering

O processo de *clustering* é dividido, basicamente, em três etapas: preparação dos dados (pré-processamento), aplicação da técnica de agrupamento (processamento) e, por último, avaliação dos resultados (pós-processamento), como representado na Figura 2.1. Essas etapas são detalhadas a seguir.

Figura 2.1: Representação do processo de *clustering*.



Exemplos: conjunto de exemplos que deverão ser agrupados.

Preparação dos Dados (Seleção e Extração de Características): escolha do melhor conjunto de características (atributos) que descrevem os exemplos, seguido por transformações e normalizações no conjunto de dados. Isso inclui eliminação de atributos que possam causar ruído e cálculos para normalização da escala dos valores, caso sejam muito diferentes.

Representação dos objetos e Configurações: representação dos exemplos em um formato que possa ser reconhecido pelo algoritmo, além de informações que devem ser utilizadas para que possa ser realizado o *clustering,* tais como número de grupos desejado, tamanho e formato dos padrões, matriz de distância, etc.

Técnica de Agrupamento: escolha e aplicação da técnica de *clustering* apropriada, como *Clustering* Aglomerativo Hierárquico, *K-Means* ou SOM, entre outras.

Clusters: agrupamento dos objetos de entrada de acordo com suas características.

Avaliação dos Resultados: validação das descobertas efetuadas pela aplicação da técnica de agrupamento. Pode ser realizada por meio de métricas de avaliação, ferramentas de visualização e pelo conhecimento de especialistas.

As técnicas de *clustering* são classificadas como métodos de aprendizagem não supervisionada, isto é, no conjunto de exemplos (objetos) considerado, não é informado um atributo especial (classe) na descrição do objeto. Portanto, os objetos que são considerados pertencentes a um cluster, bem como o número total de clusters, são dependentes dos critérios e procedimentos adotados para encontrar esses clusters. De acordo com esses procedimentos, muitas soluções (diferentes conjuntos de clusters) podem ser obtidas para um mesmo conjunto de dados.

As técnicas de *clustering* desenvolvidas nos últimos tempos têm apresentado novas estratégias, vantagens e desvantagens, tornando difícil sua classificação em uma taxo-

nomia. Berkhin, no entanto, propõe que uma boa técnica de *clustering* deve atender alguns pré-requisitos , comentados a seguir (Berkhin, 2002):

Escalável a alta dimensionalidade: habilidade para manipular dados de alta dimensão.

Versatilidade: os objetos da base de dados podem ser de formatos diversificados, tais como numérico, binário e categórico. O método de *clustering* ideal deve ser capaz de tratar todos os formatos.

Habilidade de encontrar clusters com diferentes formatos: esse é um requisito muito importante para *clustering* de dados espaciais. Alguns algoritmos conseguem descobrir apenas clusters de formato esférico.

Poucos parâmetros de entrada: o método deve requerer uma mínima quantidade de conhecimento de domínio para realizar o *clustering* corretamente. Contudo, a maioria dos algoritmos necessita de vários parâmetros (como quantidade de clusters e tipo dos dados).

Robusto com relação a ruído: esse é um fator importante, pois ruídos existem em qualquer problema prático e um bom algoritmo de *clustering* deve ser capaz de ser executado com sucesso mesmo na presença de grande quantidade de ruído.

Insensível à ordem de entrada dos dados: o método deve fornecer resultados consistentes, independentemente da ordem na qual os dados são apresentados.

Na próxima seção são explicadas com mais detalhes as etapas mostradas na Figura 2.1.

2.2 Síntese do Processo de Clustering

Esta seção, baseada em (Lopes, 1995), apresenta uma síntese do processo de *clustering*, que pode ser definida da seguinte maneira:

dado: um conjunto de objetos O.

meta: distinguir agrupamentos (i.é, subconjuntos de O) $S_1, ..., S_n$, tal que a similaridade entre objetos em um agrupamento seja maximizada e a similaridade entre

agrupamentos distintos seja minimizada.

Para realização dessa tarefa são necessárias três etapas distintas relacionadas com:

1. o tratamento dos dados;
2. a definição da métrica de similaridade;
3. e a definição do algoritmo de agrupamento.

Além dessas etapas, pode-se, também, considerar a etapa de avaliação dos resultados. Nas próximas seções são discutidas cada uma dessas etapas.

2.2.1 Tratamento dos Dados

Uma tarefa importante da engenharia de conhecimento é a definição dos objetos e a atribuição dos valores das suas características. Obtidos esses dados, o primeiro passo está relacionado com a análise das variáveis representadas. O outro passo é o tratamento desses dados. Por exemplo, se os objetos são agrupados segundo duas características com unidades distintas (centímetro e quilograma) esses mesmos objetos com valores expressos nas unidades milímetro e tonelada podem, eventualmente, formar agrupamentos distintos. São, portanto, necessários alguns cuidados no instante de agrupar as informações, sendo que, de algum modo, esses dados precisam ser relativizados.

Padronização dos Dados: de um modo geral é possível estabelecer coeficientes de similaridade para variáveis quantitativas, variáveis qualitativas nominais e variáveis qualitativas ordinais. A seguir, são descritas algumas transformações possíveis para os três tipos de variável, a fim de estabelecer os coeficientes de similaridade.

Variáveis quantitativas: um dos critérios pode ser a padronização estatística, ou seja, subtrair a média $\overline{x_i}$ de cada observação e dividir pelo respectivo desvio padrão s_i obtendo-se valores z_i na mesma ordem de grandeza para as diversas variáveis. Isso é:

$$z_i = \frac{x_i - \overline{x_i}}{s_i}, i = 1, ..., n. \tag{2.1}$$

Aplicando a padronização estatística aos elementos da Tabela 2.1 (que descreve cinco objetos em função de quatro atributos com seus respectivos valores) obtem-se a Tabela 2.2 com os dados padronizados.

Tabela 2.1: Dados Iniciais

Obj/Atr	x_1	x_2	x_3	x_4
O1	0,0	15,5	31,0	250
O2	0,0	19,2	15,4	100
O3	1.3	19,9	13,1	150
O4	18,0	2,2	0,1	90
O5	9,3	1,2	0,3	42

Tabela 2.2: Dados Padronizados

Obj/Zatr	z_1	z_2	z_3	z_4
O1	-0,87	0,76	1,88	1,86
O2	-0,87	1,17	0,57	0,05
O3	-0,67	1,24	0,37	0,65
O4	1,87	-0,71	-0,72	-0,08
O5	0,55	-0,82	-0,70	-0,66

Existem também técnicas de padronização de dados aplicáveis a variáveis qualitativas nominais e variáveis qualitativas ordinais. Por exemplo, a variável qualitativa nominal definida por $Y = j$, tal que $j = 1, 2, ..., n$ é transformada em n variáveis dicotômicas x_k, tal que se $Y = k$ então $x_i = 1$, para $i = k$ e $x_i = 0$ para $i <> k$. Assim, supondo a variável Sexo, que pode ser M ou F, tem-se, então, as variáveis SexoM e SexoF, tal que, se o sexo for M, o valor de SexoM é 1 e o de SexoF é 0.

2.2.2 *Métrica de Similaridade*

A maioria das técnicas *clustering* necessita usar uma medida de similaridade entre os elementos a serem agrupados, normalmente expressa como uma função distância, densidade ou uma métrica. É importante informar que uma função de distância ou densidade não é necessariamente uma métrica, como definida a seguir.

Seja M um conjunto de vetores em um espaço n-dimensional, uma métrica em M é uma função d: M×M $\rightarrow \Re$, tal que, para quaisquer A, B, C $\in M$, tem-se:

1. $d_{AB} > 0$ - para todo A $\neq$ B;
2. $d_{AB} = 0 \Leftrightarrow$ A = B;
3. $d_{AB} = d_{BA}$;

4. $d_{AB} \leq d_{AC} + d_{CB}$.

A proximidade de dois pontos $A(x_1, y_1)$ e $B(x_2, y_2)$ reflete a similaridade entre eles segundo as variáveis usadas para descrevê-los, ou seja, quanto mais próximos estiverem os pontos, maior a similaridade entre eles e, quanto maior a distância, maior a dissimilaridade. De fato, é justamente a distância euclideana dada por

$$d(A, B) = \left[(x_1 - x_2)^2 + (y_1 - y_2)^2\right]^{1/2}, \tag{2.2}$$

que é normalmente utilizada como coeficiente de parecença entre os dois pontos A e B. Uma medida de similaridade comumente usada é a generalização da distância euclideana média para um espaço de dimensão k (que pode ser vista na sub-seção 3.3.1), para calcular a distância entre os diversos pares de objetos.

Os estatísticos utilizam o termo parecença que, tecnicamente, pode ser dividido em duas categorias: medidas de similaridade e de dissimilaridade. Na primeira, quanto maior o valor observado, mais parecidos são os objetos. Na segunda categoria, quanto maior o valor observado, menos parecidos são os objetos (mais dissimilares).

Calculando a distância euclideana para todos os pares da matriz de dados padronizados $d(A, B)$, obtém-se a matriz de distância (Tabela 2.3).

Tabela 2.3: Matriz de similaridade (distância)

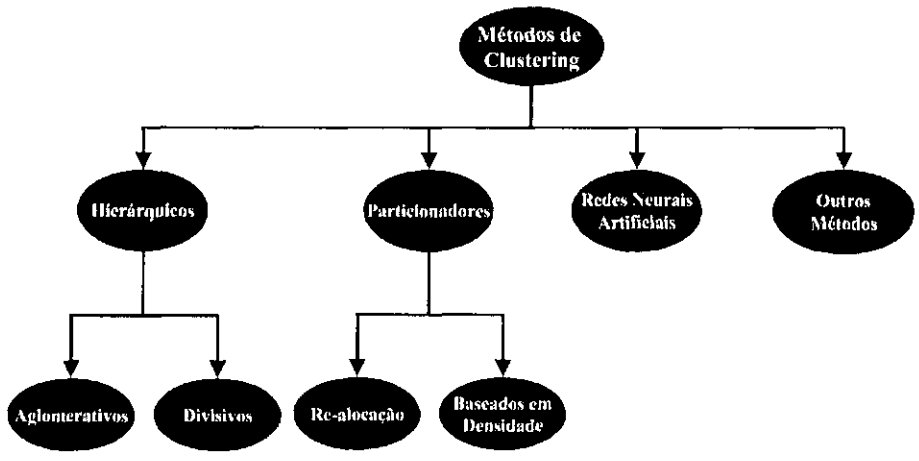
$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(n,1) & d(n,2) & \ldots & \ldots & 0 \end{bmatrix}$$

Como nesse cálculo é computada a distância entre os pares, deve ser observado que quanto menor o índice, maior é a similaridade entre os pares de objetos. Esse assunto é complementado na Seção 3.3, Métricas de Similaridade entre Documentos.

2.2.3 *Algoritmos de Clustering*

Nenhum dos algoritmos disponíveis atualmente é capaz de satisfazer todos os requisitos de uma técnica ideal (Berkhin, 2002). Assim, cada algoritmo possui particularidades que podem torná-lo eficaz para um determinado conjunto de exemplos e ineficaz para outro. Alguns desses algoritmos não podem ser utilizados com dados de alta dimensão, não sendo úteis para tratamento de documentos. Por esse motivo, o usuário deve, antes de escolher o algoritmo, conhecer o problema no qual se deseja realizar o *clustering*. Uma possível classificação dos algoritmos de *clustering* é exibida na Figura 2.2, que mostra alguns dos métodos mais citados na literatura.

Figura 2.2: Taxonomia dos Métodos de *Clustering*



- **Métodos Hierárquicos**

As técnicas de *Clustering* Hierárquico constroem hierarquias entre os objetos agrupados. O método aglomerativo (*bottom-up*) encontra os dois pares mais semelhantes e os agrupa formando um cluster. Os pares podem ser objeto/objeto, objeto/cluster ou cluster/cluster. Os objetos são agrupados até que se tenha apenas um cluster (raiz).

O método divisivo (*top-down*) inicia com um cluster contendo todos os objetos e vai dividindo os clusters recursivamente, de acordo com um determinado critério. O processo termina quando se alcança um critério de parada, geralmente o número de k clusters desejado pelo usuário. Algoritmos clássicos são SLINK (Sibson, 1973), COBWEB (Fisher, 1987) e CURE (Guha et al., 1998).

- **Métodos Particionadores**

Os métodos particionadores não aprendem gradualmente (de maneira hierárquica), mas diretamente, por meio da realocação iterativa dos objetos nos agrupamentos, ou tentando identificar clusters em áreas com muitos objetos. Algoritmos de re-alocação podem ser classificados como *K-medoids* (Kaufman & Rousseeuw, 1990), *K-means* (Macqueen, 1967) ou como probabilísticos (EM (Dempster et al., 1977), AUTOCLASS (Cheeseman & Stutz, 1996), MCLUST (Fraley & Raftery, 1999)). Tais métodos se concentram em quão bem os objetos se encaixam dentro dos clusters e tendem a construir clusters de forma convexa.

Algoritmos baseados em densidade tentam encontrar objetos conectados por densidade, sendo flexíveis em termos da forma dos clusters. A conectividade baseada em densidade é a técnica utilizada em algoritmos como DBSCAN (Ester et al., 1996) e OPTICS (Ankerst et al., 1999), sendo baseada no conceito de que dois pontos devem ser alcançáveis por densidade por um mesmo ponto em comum no cluster (Ester et al., 1996). Já a técnica de Funções de Densidade, usada no DENCLUE (Hinnenburg & Keim, 1998), utiliza funções como a Gaussiana ou uma *step function*[1] para medir a densidade em um espaço métrico de atributos. Algoritmos baseados em densidade são menos sensíveis a ruídos, sendo capazes de descobrir clusters de formato irregular. Porém, pela necessidade de se utilizar uma estrutura de dados que permita uma busca por vizinhos mais próximos (*Nearest Neighbors*), estes algoritmos têm dificuldade em trabalhar com dados de alta dimensão, visto que a partir de um determinado número de dimensões (10 a 15), a busca por vizinhos mais próximos se torna ineficiente (Beyer et al., 1999).

- **Redes Neurais Artificiais**

Uma técnica de aprendizado não supervisionado comumente utilizada para *clustering* é a rede SOM (Kohonen et al., 1996; Kohonen, 1990; Kohonen, 2001). Redes Neurais Artificiais (RNA) são técnicas importantes em AM, realizando *clustering* por meio da adaptação de seus neurônios aos padrões de entrada. O treinamento em vários tipos de RNA usa otimização de função, sendo uma técnica importante a *Gradient*

[1]Valores de ativação abaixo de um determinado limiar são modificados para 0 ou −1 e os demais valores são modificados para +1.

Descent, que utiliza a derivada dessa função para descer iterativamente por sua curva, com a finalidade de se encontrar um mínimo local (minimização do erro). Algoritmos de *clustering* têm sido utilizados, também, para segmentação de imagens (Cheng et al., 1996; Yao et al., 2000), visão computacional (Rogers et al., 1995; Zhang et al., 2004) e reconhecimento de padrões (Bishop, 1995; Jesan, 2004).

Os métodos evolucionários, que podem ser utilizados para se definir parâmetros para aplicações de mineração de dados, funcionam re-alocando um ponto de sua localização atual para um cluster escolhido aleatoriamente. Dessa maneira, o algoritmo tenta tratar o problema com uma função objetivo mais apropriada. Uma técnica conhecida é a busca tabu (Al-Sultan, 1995). Algoritmos genéticos também têm sido implementados para *clustering* de modo a encontrar melhores centróides para o algoritmo *K-Means* (Babu & Murty, 1993; Sarafis et al., 2002), por exemplo.

- **Outros Métodos**

Existem, ainda, diversos outros métodos menos conhecidos ou utilizados, por serem muito específicos ou estarem em processo de investigação. Alguns destes métodos são Baseados em *Grid* (BANG (Schikuta & Erhart, 1997), STING (Wang et al., 1997) e WaveCluster (Sheikholeslami et al., 1998)), Métodos Baseados na Co-Ocorrência Categórica de Dados (ROCK (Guha et al., 1999), SNN (*Shared Nearest Neighbors*) (Ertoz et al., 2003) e CACTUS (Ganti et al., 1999)) e *Clustering* baseado em Restrições ((Han et al., 2001; Tung et al., 2001)).

Os métodos baseados em *grid* trabalham com os dados de forma indireta, por meio da construção de sumários dos dados sobre subconjuntos do espaço de atributos, realizando segmentação do espaço e unindo segmentos apropriados. Estes algoritmos costumam ser rápidos e capazes de tratar ruídos.

Dados categóricos são comuns em aplicações reais de dados transacionais, em que o conceito simples de similaridade não é suficiente para agrupar tal tipo de dados, sendo, para isso, usada a idéia de co-ocorrência categórica. Tal tipo de dados pode ser exemplificado como compras em um mercado ou *logs* de *websites*. Estes dados costumam ser atributos binários, informando a ocorrência ou não de um determinado item em uma transação.

Clustering baseado em restrições é utilizado em ações particulares de negócio. As restrições podem ser em objetos individuais (um cliente que efetuou uma compra recentemente) e restrições de parâmetro (número de clusters), que podem ser escolhidas na etapa de pré-processamento ou podem ser parâmetros externos do cluster, restrições em clusters individuais, como limites de funções (mínimo, máximo, média), entre outras.

Escolhidos o algoritmo de *clustering* adequado ao conjunto de exemplos (convertidos para uma representação que possa ser tratada pelo algoritmo) e a métrica a ser utilizada, ainda permanece o problema de avaliação dos resultados. Os resultados são dependentes do conjunto de exemplos, do algoritmo usado e das configurações definidas pelo usuário.

Na seção a seguir, são apresentadas medidas comumente usadas para se avaliar a eficácia de algoritmos de *clustering*.

2.2.4 Avaliação dos Resultados

Na etapa de pós-processamento dos dados, é realizada a avaliação dos resultados para se medir a qualidade dos clusters encontrados pelo algoritmo. Isso permite que sejam feitos ajustes no processo como modificações nas configurações dos algoritmos ou até mesmo sua substituição a fim de se melhorar os resultados.

A avaliação pode ser feita por meio de duas medidas. Um tipo de medida permite comparar diferentes conjuntos de clusters sem uso de um conhecimento externo e é chamada de *medida de qualidade interna*. A medida de qualidade interna é geralmente a mesma função objetivo que o algoritmo de *clustering* tenta otimizar, como o critério de soma dos erros quadrados, usada pelo *K-Means* padrão, ou a coesão, baseada na similaridade entre pares de documentos em um cluster.

O outro tipo de medida permite avaliar o resultado fazendo a comparação entre os clusters produzidos e as *classes previamente conhecidas*. Esse tipo de medida é chamada de *medida de qualidade externa*. Para a avaliação de algoritmos de *clustering* de documentos são mais utilizadas as medidas de qualidade externa, quando é possível o uso de dados de treino, nos quais são conhecidas as classes dos documentos (apesar

de não serem utilizadas no processo de *clustering*). Com esse conjunto de treino, é possível ajustar os parâmetros a serem usados nos dados reais. Exemplos de medidas externas incluem a Matriz de Confusão, F-*measure*, Pureza e Entropia (Gosh, 2003).

Matriz de Confusão

Dado um conjunto de exemplos, a matriz de confusão exibe o número de classificações preditas por um classificador em oposição às classificações reais de cada exemplo, permitindo que sejam determinadas medidas de qualidade como precisão, cobertura e a F-*measure*, que são comentadas adiante. A matriz de confusão é possível de ser usada *apenas quando existe a informação da classe*. Naturalmente, deve-se observar que, em problemas reais, as classes associadas aos exemplos não existem na maioria das aplicações de *clustering*. Na Tabela 2.4 é apresentada uma matriz de confusão para problemas de classificação binária:

Tabela 2.4: Matriz de Confusão de Classificação Binária.

	Predito como Positivo	**Predito como Negativo**
Exemplos Positivos	v_p	f_n
Exemplos Negativos	f_p	v_n

Os valores da Tabela 2.4 correspondem a: v_p (verdadeiros positivos)- quantidade de exemplos positivos que foram preditos como positivos; f_p (falsos positivos)- quantidade de exemplos negativos que foram preditos como positivos; f_n (falsos negativos)- quantidade de exemplos positivos que foram preditos como negativos; e v_n (verdadeiros negativos)- quantidade de exemplos negativos que foram preditos como negativos. As taxas de erro da classificação são calculadas utilizando-se as fórmulas da Tabela 2.5.

Tabela 2.5: Taxas de erro calculadas a partir da Matriz de Confusão.

	Taxa de Erro da Classe	**Taxa de Erro Total**
Exemplos Positivos	$\frac{f_n}{(v_p+f_n)}$	$\frac{(f_p+f_n)}{n}$
Exemplos Negativos	$\frac{f_p}{(f_p+v_n)}$	$\frac{(f_p+f_n)}{n}$

A matriz de confusão para avaliar o desempenho de um sistema de *clustering* é criada na forma bi-valorada, como na Tabela 2.4, ou multi-valorada, como na Tabela 2.6.

Tabela 2.6: Matriz de Confusão Multi-Valorada.

	Classe 1	...	**Classe n**
Cluster 1	O_{11}	...	O_{n1}
...	...	...	...
Cluster n	O_{1n}	...	O_{nn}

Na Tabela 2.6, as linhas representam os clusters, as colunas representam a classe a que pertence o exemplo e as células internas representam o número de objetos da classe i no cluster j. Esta representação pode ser criada, também, quando o número de clusters difere do número de classes dos exemplos (Tabela 2.7).

Tabela 2.7: Matriz de Confusão para número diferente de classes e clusters.

	Classe 1	**Classe 2**	...	**Classe m-1**	**Classe m**
Cluster 1	**3020**	100	...	20	3
Cluster 2	**3547**	123	...	51	0
Cluster 3	3	37	...	**4598**	3894
...	...	...	...	...	...
Cluster n-1	15	40	...	235	**5896**
Cluster n	0	11	...	198	**5128**

Os números em negrito na Tabela 2.7, que representam a classe majoritária dentro do cluster, mostram o relacionamento existente entre classes e clusters. O cluster 3, por exemplo, indica proximidade entre as classes m-1 e m. Assim, pode-se concluir que os exemplos pertencentes a essas classes são bem semelhantes segundo a medida utilizada pelo algoritmo de *clustering*. Já os clusters 1 e 2 possuem resultados próximos, permitindo que se realize a fusão desses clusters, sendo que o mesmo ocorre com os clusters n-1 e n.

Como já comentado, a partir dessa matriz pode-se calcular a F-*measure* descrita a seguir.

F-measure

A F-*measure* (Larsen & Aone, 1999) combina os conceitos de precisão (do inglês *Precision*) e cobertura (do inglês *Recall*) da área de recuperação de informação (Kowalski, 1997; van Rijsbergen, 1979). Precisão é a razão de exemplos positivos que foram corretamente classificados (verdadeiros positivos) pelo número total de exemplos (positivos e negativos). Cobertura é a razão do número de exemplos positivos que foram corretamente classificados (verdadeiros positivos) pelo número de todos os exemplos positivos.

Para a avaliação, precisão e cobertura para a classe i e o cluster j são definidas como

$$Recall(i,j) = \frac{n_{ij}}{n_j} \tag{2.3}$$

$$Precision(i,j) = \frac{n_{ij}}{n_i} \tag{2.4}$$

tal que n_{ij} é o número de exemplos da classe i no cluster j, n_i é o número de exemplos da classe i, n_j é o número de exemplos no cluster j e n é o número total de exemplos.

A F-*measure* para a classe i e o cluster j é dada pela equação:

$$F(i,j) = 2\left[\frac{Recall(i,j) \times Precision(i,j)}{Recall(i,j) + Precision(i,j)}\right]. \tag{2.5}$$

A F-*measure* para todos os clusters é dada como um somatório ponderado pelas máximas F-*measures* de cada classe i como

$$F = \sum_i \frac{n_i}{n} \max_j F(i,j). \tag{2.6}$$

Pureza

A pureza fornece a razão do tamanho da classe dominante no cluster em relação ao tamanho do próprio cluster.

Supondo c categorias (rótulos de classes), enquanto o algoritmo de *clustering* produz k clusters, a pureza do cluster C pode ser definida como

$$P(C) = \frac{1}{n} \max_{h} \left(n^{(h)} \right), \tag{2.7}$$

tal que n é o número de exemplos do cluster C e $n^{(h)}$ é o número de exemplos em C que pertencem à classe h, $h = 1,\ldots, c$. Cada cluster pode conter exemplos de diferentes classes. Um valor alto de pureza implica que o cluster é um subconjunto "puro" da classe dominante. Essa medida está relacionada com a medida de precisão.

Entropia

A entropia define a homogeneidade dos clusters de acordo com o conceito de entropia usado na teoria da informação (Mitchell, 1997). Baixa entropia indica clusters mais homogêneos. Usando a mesma notação de pureza, a fórmula da entropia é dada por

$$H(C) = -\frac{1}{\log c} \sum_{h=1}^{c} \frac{n^{(h)}}{n} \log \left(\frac{n^{(h)}}{n} \right). \tag{2.8}$$

A entropia é uma medida mais compreensiva do que a pureza porque ao invés de considerar apenas o número de exemplos que "estão" e os que "não estão" na categoria mais freqüente, ela considera a distribuição como um todo. A fórmula acima é normalizada para obter valores entre 0, indicando que o cluster é composto inteiramente por uma classe e 1, indicando que o cluster possui uma divisão idêntica de classes.

2.3 Considerações Finais

No presente capítulo, além de uma visão geral sobre a técnica de *clustering*, etapas do processo de *clustering*, foram apresentadas as algumas métricas de similaridade, alguns dos algoritmos de *clustering* mais conhecidos e formas de avaliação dos resultados.

Como comentado, um passo importante é a preparação dos dados, que deve ser realizada de maneira correta para se evitar problemas com normalizações e com uma escolha incorreta do conjunto de atributos que melhor caracterizam os exemplos.

Com relação às métricas de similaridade, a mais adotada para *clustering* é a medida da distância euclideana. Outras técnicas são apresentadas na seção 3.3, complementando o assunto.

Durante a etapa de treino é possível a utilização de dados rotulados, permitindo uma avaliação automática e a definição das melhores técnicas e parâmetros. As medidas pureza e entropia, largamente usadas na literatura, são adotadas também nesse trabalho, para que possa ser feita uma comparação com os resultados de outros autores.

No capítulo a seguir, são apresentadas mais informações sobre a aplicação da técnica de *clustering* em documentos, foco deste trabalho.

Capítulo 3

Clustering de Documentos

Clustering de documentos vem sendo pesquisado para uso em uma variedade de áreas de mineração de texto e recuperação de informação. Inicialmente, essa técnica foi investigada para melhorar a precisão em sistemas de recuperação de informação (Kowalski, 1997; van Rijsbergen, 1979) e como uma maneira eficiente de resolução do problema de se encontrar os vizinhos mais próximos de um documento (Buckley & Lewit, 1985). Além disso, a técnica de *clustering* tem sido empregada para:

1. navegação em uma coleção de documentos (Cutting et al., 1992);
2. organização dos resultados de uma máquina de busca (Zamir & Etzioni, 1998);
3. geração automática de clusters hierárquicos de documentos (Koller & Sahami, 1997).

A principal etapa que diferencia o processo de *clustering* quando os dados a serem agrupados são documentos é a de pré-processamento, ou seja, na definição do modo como os dados serão representados (detalhada na próxima seção) de modo a serem processados de forma eficiente pelo algoritmo de *clustering*.

3.1 Representação de Documentos

A forma como os documentos são representados é uma característica muito importante para a aprendizagem de máquina. Tal representação deve ser de tamanho reduzido de modo a permitir um processamento eficiente de grandes conjuntos de documentos e deve conter todas as informações importantes desses documentos. A maioria das abordagens de *clustering* de documentos utiliza um modelo de espaço vetorial (van Rijsbergen, 1979; Bloedorn et al., 1996; Yan & Garcia-Molina, 1994; Moukas & Zacharia, 1997) para representar a coleção, na qual cada documento (todos no mesmo idioma) é representado como um vetor de características (atributos). Para cada característica nesse vetor é atribuído um peso que pode ser: binário, freqüência do termo no documento (*tf - Term Frequency*), freqüência do termo-freqüência inversa do documento (*tf-idf - Term Frequency-Inverse Document Frequency Weighting*[1]), dependendo da aplicação (Salton & McGill, 1986). Esses pesos são comentados adiante.

3.1.1 Bag-of-Words

Essa representação do modelo de espaço vetorial, também chamada de *bag-of-words*, é convertida em uma tabela, na qual cada linha corresponde a um documento e cada coluna corresponde a um termo (do conjunto de termos de todos os documentos) ou a um conjunto de termos (n-gramas) no documento. O uso de termos compostos (n-gramas) é aceitável quando expressa um conceito único, por exemplo "inteligência artificial". Os termos podem ser palavras ou radicais de palavras (*stems*). As demais informações contidas implicitamente no texto, como a semântica e a ordem das palavras, são perdidas quando se cria a representação vetorial do documento, pois os termos costumam ser ordenados pelo peso, de forma decrescente.

A conversão do texto em uma tabela do tipo atributo/valor considera os termos como atributos e os valores correspondentes a cada um são atribuídos de acordo com os critérios comentados a seguir:

Binário: informa a presença (1) ou não (0) do termo no documento. Essa representação é usada em recuperação de informação, basicamente em pesquisa, onde é feita

[1] A sigla em inglês (*tf-idf*) é comumente adotada na literatura.

uma combinação dos termos para selecionar documentos que satisfaçam a expressão fornecida pelo usuário. A representação é ilustrada na Tabela 3.1.

Tabela 3.1: Representação de documentos em tabela de peso binário.

Documentos	Termo 1	Termo 2	...	Termo n
d1	1	0	...	1
d2	1	1	...	0
...	...	...	...	...
dm	0	1	...	0

Freqüência do termo (tf): informa a freqüência do termo no documento (ver Tabela 3.2), possibilitando um ranqueamento dos documentos de acordo com a freqüência dos termos encontrados na pesquisa feita pelo usuário, ou seja, o documento que possuir a maior ocorrência dos termos fornecidos pelo usuário aparece primeiro na lista.

Tabela 3.2: Representação de documentos em tabela de peso por freqüência.

Documentos	Termo 1	Termo 2	...	Termo n
d1	3	15	...	0
d2	7	20	...	11
...	...	...	...	...
dm	0	2	...	8

Freqüência do termo-freqüência inversa do documento (tf-idf): informa a relação entre a freqüência do termo no documento (tf) e o inverso da freqüência do termo na coleção de documentos (idf_k). O peso de cada termo k no documento d (w_{kd}) pode ser calculado de acordo com as equações:

$$w_{kd} = tf.idf_k, \tag{3.1}$$

$$idf_k = log\left(\frac{N}{df_k}\right), \tag{3.2}$$

sendo N o número total de documentos pesquisados e df_k o número de documentos em que o termo k aparece. A idéia básica é que termos raros não são informativos para caracterização de um documento e os termos muito comuns têm seu peso reduzido. Como exemplo tem-se a Tabela 3.3.

Tabela 3.3: Representação de documentos em tabela de peso por *tf-idf*.

Documentos	Termo 1	Termo 2	...	Termo n
d1	0.10	0.70	...	0.30
d2	0.30	0.90	...	0.85
...	...	...	...	...
dm	0.25	0.33	...	0.67

A representação vetorial é criada na etapa de pré-processamento da coleção de documentos. Para se construir a tabela são realizados os passos a seguir.

1. **Extração de todas as palavras de todos os documentos (criação do dicionário):** usa-se um tokenizador[2] para realizar a extração das palavras. Números, figuras, tabelas, pontuação e equações são ignorados. Essa etapa pode ser realizada ao mesmo tempo em que as etapas 2 e 3, descritas a seguir.

2. **Remoção das *stopwords*:** conjunto de palavras consideradas não relevantes na análise dos textos. São palavras que, de um ponto de vista estatístico, agregam pouca capacidade à discriminação do texto, repetem-se em qualquer texto e podem ser desprezadas ao verificar e avaliar o conteúdo específico de um certo texto. Costumam ser palavras conectivas, preposições, pronomes, artigos e outras classes de palavras auxiliares. Podem fazer parte da *stoplist*, também, palavras de determinados domínios que podem se repetir durante o texto. Esse conjunto de palavras é removido do dicionário.

3. ***Stemming*:** processo de redução de cada uma das palavras do dicionário a um radical, quando possível. Por exemplo, as palavras CONSIDERAR, CONSIDERADO, CONSIDERAÇÃO, CONSIDERAÇÕES são reduzidas a CONSIDER.

[2]Programa de computador que realiza a extração de termos (palavras, números, pontuação, etc). Também conhecido como *parser* ou analisador sintático.

Os algoritmos de *stemming* correntes não usam, por exemplo, informações do contexto para determinar o sentido correto de cada palavra. Basicamente, os algoritmos realizam remoção de sufixos como "AR", "ADO", "AÇÃO", "AÇÕES", deixando apenas o radical comum. Casos em que o contexto ajuda no processo de *stemming* não são freqüentes (casa: verbo e substantivo) e a maioria das palavras pode ser considerada como apresentando um significado único. Erros resultantes de uma análise de sentido imprecisa das palavras não impactam no ganho obtido pelo aumento da precisão com *stemming*.

4. **Cálculo do peso de cada característica do vetor:** para finalizar a criação da tabela basta atribuir os pesos a cada um dos termos de acordo com o peso determinado pelo usuário (binário, freqüência, *tf-idf*).

3.1.2 Bag-of-References

Uma informação importante contida em artigos científicos e que não costuma ser tratada individualmente é sua seção de referências bibliográficas. Assim, além de elementos como palavras-chave, título, resumo e outras informações que existam no texto do artigo, é de se esperar que artigos semelhantes ou que abordem temas semelhantes, citem algumas referências em comum. Além disso, conhecendo-se as referências que cada artigo possui, tem-se acesso às informações como o impacto causado por um determinado artigo, autores, artigos mais citados e quantidade de publicações por ano. Referências em comum são uma indicação de similaridade ainda mais importante do que a informação expressa pela freqüência de um termo no documento, pois pode indicar a similaridade em termos de sub-áreas, ao invés de uma grande área, por exemplo: dois documentos sobre *Case-Based Reasoning* possuem, em geral, entre os termos mais significativos, *case* e *reasoning*, indicando a grande área da qual o artigo trata. Porém, analisando-se as referências, podemos encontrar indicações da sub-área específica em que o artigo se encaixa, ou seja, os trabalhos relacionados com o artigo em questão.

Para se usar a seção de referências bibliográficas é necessário realizar extração de informação e criar uma base de dados, na qual cada uma das referências devem possuir um identificador único, sendo que as referências iguais, em artigos diferentes, precisam ter o mesmo identificador. Com isso, tem-se uma nova representação denominada *bag-of-references* (ver Tabela 3.4), similar à *bag-of-words*, sendo que cada

atributo passa a ser uma referência ao invés de um termo/palavra, de modo a ser utilizada como informação extra no *clustering* de artigos científicos. Em (Melo & Lopes, 2004b) mostra-se que o uso das referências pode melhorar o resultado do clustering. Experimentos usando essa representação são apresentados na seção 5.2.3

- **Criação da *bag-of-references***

Referências a um mesmo artigo podem ter diferenças significantes de grafia, em diferentes artigos (ver Figura 3.1). Assim, cada referência citada em um artigo precisa ser identificada individualmente para que se possa saber quando uma mesma referência aparece em artigos distintos. A identificação é feita comparando-se atributos (autor e título) de duas referências. Se a similaridade entre elas for acima de um determinado limiar, as referências são consideradas iguais.

Figura 3.1: Diferentes formatos, em diferentes artigos, para uma mesma referência bibliográfica.

[28] Stephen Muggleton and Wray Buntine. Machine invention of first-order predicates by inverting resolution. In *Proc. of ICML 88*, pages 339-352. Morgan Kaufmann, 1988.
Muggleton, S., and Buntine, W., 1988. "Machine Invention of First-Order Predicates by Inverting Resolution", In Morgan Kaufmann, editor. *Proceedings of the 5th International Conference on Machine Learning*, pp. 339-352.
Muggleton, S., & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning*, pp. 339352 Ann Arbor, MI.

Este problema foi tratado em (Melo et al., 2003; Melo & Lopes, 2004a; Melo & Lopes, 2005) e em outros trabalhos (Hylton, 1996; Monge & Elkan, 1997) e é descrito brevemente a seguir.

A técnica utilizada para extrair informações e estruturá-las em registros é normalmente baseada em regras e em heurísticas definidas manualmente, podendo ser melhorada com uso de técnicas de aprendizado indutivo de regras (usado em (Geng &

Yang, 2003)) ou técnicas baseadas em uma estimação probabilística em conjuntos de exemplos de dados bibliográficos conhecidos (Lawrence et al., 1999).

Em geral, para cada registro na base de dados é feita uma pesquisa de modo a se encontrar todos os demais registros similares e atribuir a esses registros um mesmo código (ID). Por este motivo, este processo tem um alto custo computacional.

Para a identificação, é utilizada uma métrica de distância de edição, entre componentes de distintos registros, que fornece o grau de similaridade entre duas *strings*. Entre referências bibliográficas, as *strings* são comumente formadas pela concatenação dos campos autor e título (Melo et al., 2003b; Melo & Lopes, 2004a; Melo & Lopes, 2005; Hylton, 1996; Monge & Elkan, 1997). Tal métrica mede a similaridade de acordo com a quantidade de inserções, remoções e substituições necessárias para converter a *string* de origem na *string* de destino. Exemplos de algoritmos são o de Levenshtein (Levenshtein, 1966), Smith-Waterman (Smith & Waterman, 1981), Needleman-Wunsch (Needleman & Wunsch, 1970) em implementações de ferramentas como o *agrep* do *Unix*.

A fim de se reduzir o custo computacional desta tarefa foi adotada a seguinte abordagem:

1. extrai-se os campos autor(es), título e ano de publicação de cada uma das referências e os armazena em uma tabela;

2. para cada referência cria-se uma chave (*string*), de tamanho variável, composta pelo primeiro caracter de cada um dos termos existentes em autor(es) e título, extraídas algumas *stopwords* (*a, and, in, for, of, the, to*, etc), dígitos e símbolos delimitadores. Ordena-se os caracteres por campo e elimina-se os repetidos. Como exemplo, a chave **BCEMRCFIRS** é criada para a referência **E. Rosch and C.B. Mervis. Family resemblance studies in the internal structure of categories.**

3. ordena-se a tabela em ordem ascendente pelo ano, pela chave, pelos autor(es) e pelo título. Alguns artigos possuem referências sem ano de publicação. Tais referências são posicionadas no final da tabela para que possam ser identificadas quando alguma referência similar, que possua ano, já tenha sido identificada. Com essa ordenação, referências iguais, grafadas de forma variada, tendem a

ficar juntas.

4. a busca para indexação das referências é feita pela chave, de modo a selecionar um conjunto de referências similares a uma dada referência, que podem vir a serem idênticas. Calcula-se, então, a distância entre as referências inteiras (a que se deseja indexar e cada uma das referências retornadas pela busca).

5. como a similaridade é calculada por uma métrica, utilizamos uma estrutura de dados métrica para armazenar as chaves. Com isso, consegue-se fazer pesquisas pelas k chaves mais similares (*k Nearest Neighbors*) à chave de entrada, reduzindo consideravelmente o número de comparações entre as referências inteiras (Melo et al., 2003b; Melo & Lopes, 2004a; Melo & Lopes, 2005).

As referências, após serem identificadas, são processadas pela ferramenta *PreText* (Matsubara et al., 2003), apenas para se gerar a *bag-of-references* (Tabela 3.4), sem realizar *stemming*, remoção de *stopwords* ou cortes. Não se faz nenhum corte pois documentos de áreas recentes com referências pouco citadas, podem acabar possuindo vetores nulos.

Tabela 3.4: *Bag-of-words* e *Bag-of-references*

Documentos	Termo 1	Termo 2	...	Termo n	Ref 1	Ref 2	...	Ref n
d1	0.20	0.43	...	0.74	0.10	0.40	...	0.50
d2	0.15	0.56	...	0.60	0.13	0.38	...	0.55
...	...	...	...	...	...	...	...	...
dm	0.23	0.49	...	0.81	0.09	0.43	...	0.47

Pode-se, então, utilizar 3 tabelas no algoritmo de *clustering*:

1. *Bag-of-words*;

2. *Bag-of-references*;

3. *Bag-of-words* unida à *Bag-of-references*.

O uso do modelo de espaço vetorial como representação se deve, basicamente, ao fato de que operações utilizando vetores podem ser executadas rapidamente e existem algoritmos especializados para realizar a construção do modelo, a redução da dimensionalidade e a visualização de espaços de vetores.

Porém, a representação dos documentos utilizando tabela possui dois inconvenientes:

1. como os vetores de características são criados a partir do dicionário, que pode conter milhares de palavras, a dimensão dos vetores é extremamente alta;

2. cada documento não possui todas as palavras do dicionário. Pelo contrário, possui uma quantia substancialmente pequena (algo em torno de 1% a 2%). Isso significa que a grande maioria dos seus atributos terá valor igual a 0. Assim, a matriz gerada para a coleção de documentos é altamente esparsa.

Tais problemas podem ser tratados por meio de técnicas de redução de dimensionalidade, comentadas a seguir. Porém, como dito anteriormente, a redução aplicada a referências bibliográficas (corte) causa perda de informação, gerando vetores nulos.

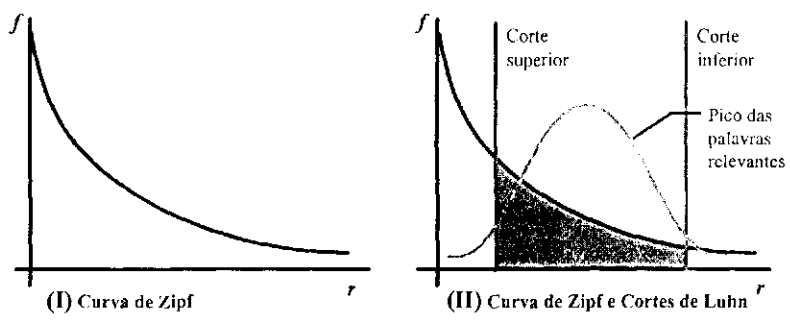
3.2 Redução de dimensionalidade

Existem diferentes maneiras de se referir ao mesmo objeto ou conceito (uso de sinônimos). Um outro problema é o fato de que várias palavras podem ter significados diferentes dependendo do contexto em que estão incluídas (polissemia). Métricas de similaridade assumem que as palavras são independentes, mas a ocorrência de uma determinada palavra em conjunção com outra não pode ser tomada como uma evidência independente de similaridade entre documentos, ou seja, a ordem e a localização de palavras possuem um significado importante e que deveria ser levado em consideração na comparação de documentos. Há técnicas que podem ser empregadas para realizar a redução de dimensionalidade do vetor de atributos explorando esse tipo de informação, tentando substituir vários atributos por apenas um. Vale notar que as técnicas de *stemming* e remoção de *stopwords* também realizam redução de dimensão.

3.2.1 Cortes de Luhn

Ordenando todos os termos de uma coleção de documentos de acordo com sua freqüência na coleção e fazendo-se o histograma dessa ordenação, tem-se a "curva de Zipf" para a dada coleção de documentos (Zipf, 1949). Luhn especifica dois pontos de corte na curva para excluir termos não relevantes (Luhn, 1958). Os termos acima do corte superior são muito freqüentes e considerados comuns por aparecerem em qualquer tipo de documento (*stopwords*). Já os termos abaixo do corte inferior são considerados raros, não sendo importantes na discriminação dos documentos. A Figura 3.2 ilustra a curva da Lei de Zipf (I) e os cortes de Luhn aplicados à Lei de Zipf (II), tal que f representa a freqüência das palavras e r as palavras correspondentes ordenadas pela freqüência.

Figura 3.2: Ilustração representando a curva de Zipf (I) e os cortes de Luhn (II) (Matsubara et al., 2003).



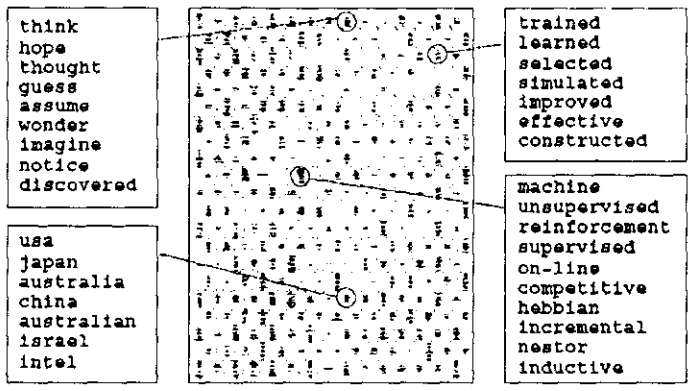
Luhn propõe que os termos mais significativos para discriminar o conteúdo do documento se encontram em um pico imaginário localizado no centro, entre os dois pontos de corte. Porém, os pontos de corte são definidos pelo usuário por tentativa e erro (van Rijsbergen, 1979).

Dessa maneira, o que costuma ser feito é eliminar os termos com freqüência inferior a um certo limiar, por exemplo, abaixo de duas ocorrências. Esse corte elimina boa quantidade dos termos, muitas vezes superando 50%. Outro tipo de corte é determinar qual a dimensão desejada da matriz. O corte pode ser feito, por exemplo, de modo a se utilizar apenas os 1000 termos mais freqüentes.

3.2.2 Mapa Semântico de Palavras

Mapas SOM semânticos (Ritter & Kohonen, 1989) permitem que documentos sejam representados por um conjunto consideravelmente menor de características, realizando o *clustering* de palavras em categorias (Figura 3.3). O SOM agrupa palavras similares de acordo com sua ocorrência no texto. Assim, palavras com contexto semelhante tendem a aparecer próximas entre si, permitindo representar documentos em função dos clusters de palavras ao invés de palavras individuais (Honkela et al., 1996). Dessa maneira, supondo-se que um vetor de 1000 palavras seja utilizado em um mapa semântico de tamanho 10x10, essas 1000 palavras podem ser agrupadas em 100 classes (10x10), reduzindo para 10% o tamanho do vetor de características que representa um documento, ou seja, cada documento será composto por um vetor de 100 posições, ao invés de 1000.

Figura 3.3: Exemplo de Mapa de Categorias de Palavras, de tamanho 15x21 nodos.



Dessa forma são formados grupos (categorias) de palavras que possam ter o mesmo significado (sinônimas) ou pertencer a um mesmo tipo de padrão (nomes de países, marcas de produtos, etc) em que uma das palavras é escolhida para representar o seu cluster (Ritter & Kohonen, 1989; Scholtes., 1993). Essa técnica foi utilizada no WEBSOM e seu resultado pode ser visto no final da seção **4.4.1**.

3.2.3 Thesaurus

Pode-se usar um *thesaurus* para se obter um resultado semelhante ao do Mapa Semântico de Palavras. A técnica empregada por Grefenstette (Grefenstette, 1994) usa

processamento de linguagem natural para derivar o contexto das palavras que aparecem nos textos. Esse *thesaurus* pode ser construído automaticamente, detectando termos relacionados entre si por meio de sua co-ocorrência em pares de documentos (Salton & McGill, 1986). Caso não se deseje que o *thesaurus* seja criado automaticamente a partir da coleção de textos, pode ser utilizada uma base de dados já pronta, que será acessada durante a criação da representação dos documentos para se evitar a inclusão de termos sinônimos no vetor de características.

3.2.4 Indexação por Semântica Latente

A LSI (*Latent Semantic Indexing*) (Foltz & Dumais, 1992; Berry et al., 1994) utiliza a noção de que relacionamentos entre palavras podem ser deduzidos pelo seu padrão de ocorrência entre documentos, por meio da aplicação da técnica SVD (*Singular Value Decomposition*) (Deerwester et al., 1990) na matriz de representação dos documentos, a fim de se obter uma projeção de documentos e palavras em um espaço conhecido como espaço latente.

A redução de dimensionalidade é alcançada retendo-se apenas as variáveis latentes (dimensões da projeção) com a maior variância. Cálculos de distância subseqüentes entre documentos ou termos são então realizados no espaço latente de tamanho reduzido. O algoritmo original de LSI tem uma complexidade computacional muito alta, $O(n^3)$, tal que n é o número de documentos, o que o torna problemático para uso em grandes conjuntos de dados. A Projeção Randômica (Drineas et al., 1999) tem sido usada por ser um método mais rápido do que a LSI e produzir resultados muito semelhantes. Além disso, a própria Projeção Randômica pode ser utilizada para se reduzir o tamanho da matriz antes de usá-la na LSI.

3.2.5 Fastmap

Faloutsos & Lin propuseram uma implementação alternativa do MDS (*Multi-Dimensional Scaling*) (Torgerson, 1958) que usa distância euclideana para aproximar uma métrica geral (Faloutsos & Lin, 1995). A heurística do algoritmo funciona considerando os objetos como pontos de algum espaço k-dimensional desconhecido. Tais pontos são projetados, de forma iterativa, nos hiper-planos perpendiculares a um con-

junto ortogonal de k-linhas passando através dos objetos mais dissimilares. A idéia básica considera que, dados N vetores com m atributos cada, encontre N vetores com k atributos, tal que $k < m$, de forma que as distâncias sejam mantidas o quanto for possível. Esse algoritmo possui complexidade de tempo linear com o número de objetos.

Na seção seguinte são apresentadas algumas métricas que podem ser utilizadas para se obter uma medida da similaridade entre dois documentos.

3.3 Métricas de Similaridade entre Documentos

Para que se obtenha clusters de boa qualidade, a métrica de similaridade deve ser invariante ao domínio do problema. Além disso, qualquer tipo de normalização aplicada aos dados pode modificar as propriedades da base de dados, de modo que o resultado do *clustering* é, também, modificado. Este resultado pode ser de qualidade superior ou inferior.

O conjunto de atributos (características) que representa cada um dos exemplos (documentos) também tem forte influência no *clustering*. Em uma grande quantidade de atributos pode haver muito ruído. Em uma pequena quantidade, as informações podem ser insuficientes para se encontrar um bom relacionamento entre os exemplos. Os atributos devem estar em escala comparável para que a métrica de similaridade consiga refletir as relações existentes no conjunto de dados.

As figuras nesta seção, extraídas de (Strehl et al., 2000), ilustram métricas de similaridade em 2 dimensões, sendo formadas por dois pontos $x_1 = (1, 2)$ e $x_2 = (3, 1)$, marcados com um $\times$. Para cada ponto são traçadas superfícies com valores de iso-similaridade $s = 0.25$, 0.5 e 0.75 (linhas sólidas). As linhas tracejadas correspondem à superfície equi-similar aos dois pontos. As figuras sombreadas ilustram espaços similares aos dois pontos (parte mais clara da figura). Os pontos de maior similaridade entre x_1 e x_2 são marcados com uma $\star$.

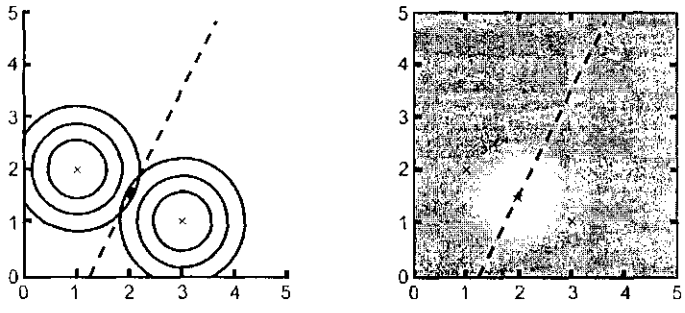
3.3.1 A distância de Minkowski

A distância de Minkowski, cuja fórmula é

$$d_k(x_i, x_j) = \left[\sum_{t-1}^{m} |x_{it} - x_{jt}|^k\right]^{1/k}, \tag{3.3}$$

é a métrica padrão utilizada na resolução de problemas geométricos, como o cálculo de distância entre dois pontos no espaço bidimensional. A distância de Manhattan é obtida quando $k = 1$. E, para $k = 2$, tem-se a distância euclideana. Usando similaridade euclideana, as iso-similaridades são hiper-esferas concêntricas em torno do ponto considerado (Figura 3.4), sendo que quanto maior a similaridade, menor o raio. Uma propriedade importante é que a única posição no espaço que possui similaridade igual a 1 é o próprio ponto. As demais posições possuem similaridade menor, à medida que se afastam do ponto. Porém, qualquer ponto, em uma posição finita, possui similaridade maior do que 0. Por esse motivo matrizes de similaridade nunca são esparsas.

Figura 3.4: Gráfico representando a similaridade euclideana.



Como já comentado, a medida de distância mais commumente utilizada nos algoritmos de *clustering* é a medida de distância euclideana. Porém, o uso dessa medida em um espaço vetorial não é vista como uma boa métrica para avaliar a similaridade de documentos (Strehl et al., 2000), porque o comprimento do vetor que representa um documento é determinado pelo comprimento do próprio documento e esse é um fator que não se leva em consideração ao se medir a similaridade entre documentos. Assim, uma medida que ignora o tamanho do vetor e que tem sido bastante utilizada em Recuperação de Informação é a medida do co-seno, sendo também indicada para uso em *clustering* de documentos.

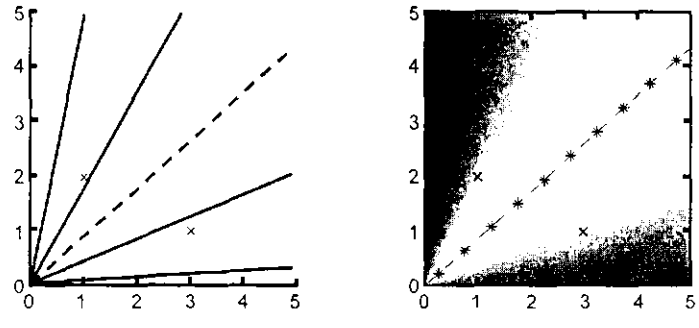
3.3.2 Medida do Co-seno

A similaridade pode ser definida também pelo co-seno do ângulo entre dois vetores. A medida do co-seno da similaridade é dada pela seguinte fórmula:

$$s(x_i, x_j) = \frac{x_i \cdot x_j}{|x_i|\,|x_j|}. \tag{3.4}$$

Uma propriedade importante é que o co-seno da entre dois vetores não depende do comprimento dos vetores. Um valor próximo a 1 indica que dois documentos são similares, enquanto que um valor próximo a 0 indica que são muito diferentes e o ângulo entre eles se aproxima de 90°. No espaço de termo-documento, os sentidos dos vetores são uma indicação mais confiável das similaridades semânticas dos objetos do que a distância entre os objetos (Frakes & Baeza-Yates, 1992).

Figura 3.5: Gráfico representando a similaridade com a medida do co-seno.



Para a medida do co-seno, as iso-similaridades são representadas como hiper-cones. Neste caso, todos os hiper-cones possuem seu ápice na origem (posição 0,0) e o eixo alinhado com o ponto considerado (Figura 3.5). As posições com similaridade 1 estão no sub-espaço unidimensional definido por esse eixo. Já os pontos com similaridade 0 são posicionados no hiperplano através da origem e perpendicular ao eixo.

3.3.3 Similaridade Estendida de Jaccard

O coeficiente binário de Jaccard é geralmente usado em aplicações que se deseja saber se um determinado evento ocorreu. No comércio, por exemplo, a similaridade

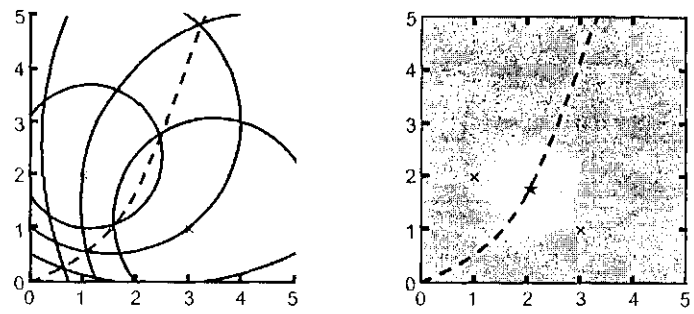
entre as compras de dois clientes pode ser computada sabendo-se quais são os produtos adquiridos. Assim, essa medida calcula o grau de sobreposição entre dois vetores (1 indica a presença do atributo e 0 indica sua ausência) como a divisão do número de atributos compartilhados pelo número total de atributos. Por exemplo, dados dois vetores binários $x_i = (0, 1, 1, 0)$ e $x_j = (1, 1, 0, 0)$, tem-se 1 atributo compartilhado pois o resultado de $x_i = (0, 1, 1, 0)$ AND $x_j = (1, 1, 0, 0)$ é $x_{i\,AND\,j} = (0, 1, 0, 0)$. Já o número total de atributos é 3, pois $x_i = (0, 1, 1, 0)$ OR $x_j = (1, 1, 0, 0)$ é $x_{i\,OR\,j} = (1, 1, 1, 0)$. Deste modo, coeficiente de Jaccard entre os vetores x_i e x_j é igual a 1/3.

A similaridade de Jaccard foi estendida (Equação 3.5) para ser capaz de computar atributos contínuos ou discretos não negativos. Nesta nova equação, x_i^T é a transposta do vetor $\overline{x_i}$ e os vetores são normalizados para se obter seu comprimento euclidiano. Essa fórmula é equivalente à versão binária quando as entradas forem binárias. A similaridade estendida de Jaccard retém a propriedade da esparsidade do co-seno ao permitir a discriminação de vetores colineares (Strehl et al., 2000).

$$s(x_i, x_j) = \frac{x_i^T x_j}{\|x_i\|_2^2 + \|x_j\|_2^2 - x_i^T x_j}, \tag{3.5}$$

A similaridade euclideana é invariante à translação, mas sensível à escala. Já a medida do co-seno é sensível à translação, mas invariante à escala. O Jaccard estendido supera esses problemas por possuir aspectos de ambas as propriedades. Para a similaridade estendida de Jaccard, as iso-similaridades são hiper-esferas não-concêntricas (Figura 3.6), sendo que a única posição com similaridade 1 é o próprio ponto. Diferentemente da distância euclideana, o ponto considerado não se localiza no centro da hiper-esfera. O raio da hiper-esfera aumenta com a distância do ponto considerado a partir da origem, de modo que vetores mais longos se tornam mais tolerantes em termos da similaridade do que vetores menores. Quando a similaridade se aproxima de 0 o raio tende ao infinito, representando a esfera no mesmo hiper-plano que foi obtido pela similaridade do co-seno. Já para a similaridade tendendo a 1, a similaridade estendida de Jaccard comporta-se como a distância euclideana.

Figura 3.6: Gráfico representando a similaridade com coeficiente estendido de Jaccard.



3.3.4 *Atributos Qualitativos*

Vale ressaltar, que além dos coeficientes de similaridade mostrados acima, adotados para atributos quantitativos, existem outras propostas para tratar outros tipos de valores. Deve ser observado que os coeficientes de similaridade, de um modo geral, são criados de acordo com o foco de interesse do pesquisador e de aspectos especiais relacionados aos atributos dos objetos analisados.

No caso de atributos qualitativos, existe a necessidade de coeficientes que definam o grau de similaridade entre objetos segundo variáveis desse tipo. São várias as propostas descritas na literatura (Wilson & Martinez, 1997; Sànchez-Marrè et al., 1998; Liao et al., 1998). Para ilustrar, considere a Tabela 3.5 com os resultados sobre a presença (1) ou não (0) de 10 atributos em dois objetos A e B (Lopes, 1995):

Tabela 3.5: Tabela de valores binários.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
A	1	1	0	1	0	1	1	0	0	1
B	1	0	0	1	0	1	1	1	0	0

Considere a Tabela 3.5 de dupla entrada computando o número observado de pares (1,1), (1,0), (0,1) e (0,0) para A e B:

Tabela 3.6: Tabela computando número de pares.

		A		
		1	0	
B	1	a = 4	b = 1	a + b = 5
	0	c = 2	d = 3	c + d = 5
		a + c = 6	b + d = 4	p = 10

Uma proposta para medida de similaridade é a proporção das coincidências, isto é,

$$s(A, B) = \frac{a + d}{a + b + c + d}. \tag{3.6}$$

Para o exemplo considerado na Tabela 3.6, o valor é igual a 0,7. Esse coeficiente varia entre 0 e 1, tal que valores próximos a 1 significam maior similaridade entre os objetos.

No caso da presença de variáveis de diferentes tipos (quantitativas, qualitativas nominais e qualitativas ordinais) uma possibilidade é construir coeficientes *cq*, *cn* e *co* para cada um dos três tipos e em seguida construir um único valor ponderado:

$$c(A, B) = w_1.co(A, B) + w_2.cn(A, B) + w_3.cq(A, B), \tag{3.7}$$

tal que w_i são pesos escolhidos convenientemente.

3.4 Considerações Finais

No presente capítulo foi apresentada uma visão geral sobre a técnica de *clustering* aplicada a documentos. Em *clustering* de documentos, geralmente, não se faz normalizações nos valores de atributos, visto que são todos baseados na freqüência dos termos. Uma normalização é feita na própria medida *tf-idf*, quando se relaciona o termo com a coleção de documentos. Por ser a técnica mais adotada na literatura, pelo fato da

normalização realizada, será também utilizada neste trabalho.

Com relação às métricas de similaridade, a distância euclideana é a mais utilizada. Porém, a similaridade do co-seno é mais indicada para uso em documentos e, mais recentemente, a similaridade estendida de Jaccard aparenta ser uma boa alternativa a essas duas, euclideana e do co-seno, por possuir características de ambas. Alguns softwares possuem várias opções e permitem que o usuário escolha entre elas. Assim, devem ser realizados testes para se definir qual métrica se mostra mais interessante em termos de qualidade dos clusters gerados e em tempo computacional, visto que a função de distância é responsável por grande parte do processamento em algoritmos de *clustering*.

CAPÍTULO
4

Algoritmos de *Clustering*

São apresentadas, neste capítulo, com um nível maior de detalhes, algumas técnicas de *clustering* mencionadas no capítulo anterior, descrevendo seus algoritmos ou conceitos, bem como seus pontos positivos e negativos.

4.1 Clustering *Hierárquico*

Existem vários métodos para análise de *clustering* e, de um modo geral, todos utilizam coeficientes de similaridade baseados em alguma medida de distância. As técnicas de *Clustering* hierárquico são variantes de técnicas como *Complete-link* (King, 1967), *Single-link* (Sneath & Sokal, 1973), *Minimum-variance* (Ward, 1963; Murtagh, 1984) e UPGMA (*Group average-link*) (Voorhees, 1986). Elas constroem hierarquias entre os objetos agrupados. O método aglomerativo (*bottom-up*) encontra os dois pares mais semelhantes e os agrupa formando um cluster. Os pares podem ser objeto/objeto, objeto/cluster ou cluster/cluster. Os objetos ou clusters são sucessivamente agrupados até que se tenha apenas um cluster (raiz). Esse tipo de algoritmo é bem flexível, sendo capaz de lidar com qualquer formato de métrica de similaridade ou de distância, bem como qualquer tipo de dados, permitindo que seja utilizado com documentos.

- Método das Médias das Distâncias (MMD)

O método MMD é um processo hierárquico *bottom-up* que em cada passo é diminuída uma dimensão da matriz de distância, reunindo os pares mais próximos (similares), até reunir todos em um único grupo. A seguir, uma descrição mais detalhada do MMD, baseada em Alneu Lopes (Lopes, 1995), é apresentada.

Um procedimento geral do Método das Médias das Distâncias é descrito no Algoritmo 1, em que *M_dist* é a matriz de distância; *Agrupamento* contém os objetos pertencentes aos clusters, bem como as distâncias entre esses objetos e *Num_grupo* é o número de clusters, inicialmente igual ao número de objetos.

Algoritmo 1: MMD

```
MMD (M_dist, Agrupamento, Num_grupo)
 Se Num_grupo = 2 Então
   Retornar Agrupamento
 Senão
   Escolher par mais próximo (Obj[i],Obj[j]);
   Reunir Obj[i] e Obj[j] no Agrupamento[k];
   Calcular a distância do Agrupamento[k] aos objetos (agrupamentos)
restantes;
   Recalcular M_dist;
   MMD(M_dist, Agrupamento, Num_grupo - 1);
 Fim se.
```

O MMD é ilustrado a seguir, considerando a matriz de distâncias construída com base nas distâncias entre pares de objetos. No exemplo apresentado é usada a distância euclidiana.

Passo 0: eliminar da matriz de distâncias a primeira linha e a última coluna por ter valores nulos.

	O1	O2	O3	O4	O5	O6
O2	1,14					
O3	1,00	0,34				
O4	2,25	1,78	1,73			
O5	2,09	1,42	1,47	0,72		
O6	2,13	1,39	1,47	0,87	0,16	
O7	2,08	1,31	1,43	1,20	0,48	0,36

Passo 1: agrupar o par de maior similaridade: O5 e O6.

Forma-se então os grupos: O2, O3, O4, O7, (O5-O6).

Reconstruir a matriz de distâncias:

	O1	O2	O3	O4	O7
O2	1,14				
O3	1,00	0,34			
O4	2,25	1,78	1,73		
O7	2,08	1,31	1,43	1,20	
O5-O6	2,11	1,41	1,47	0,79	0,42

Como os quatro primeiros grupos não sofreram alterações as distâncias entre eles também permanecem as mesmas. É necessário, agora, definir a distância entre o grupo (O5-O6) e os demais pontos. Nesse método, a distância é definida como sendo a média das distâncias individuais dos elementos de um grupo com os do outro.

Passo 2: repetir o passo anterior utilizando a nova tabela, agrupando O2 e O3.

	O1	O4	O7	O5-O6
O4	2,25			
O7	2,08	1,20		
O5-O6	2,11	0,79	0,42	
O2-O3	1,07	1,76	1,37	1,44

Passo 3: agrupar (O5-O6) e O7.

	O1	O4	O2-O3
O4	2,25		
O2-O3	1,07	1,76	
O5-O6-O7	2,09	1,00	1,40

Passo 4: agrupar (O5-O6-O7) e O4.

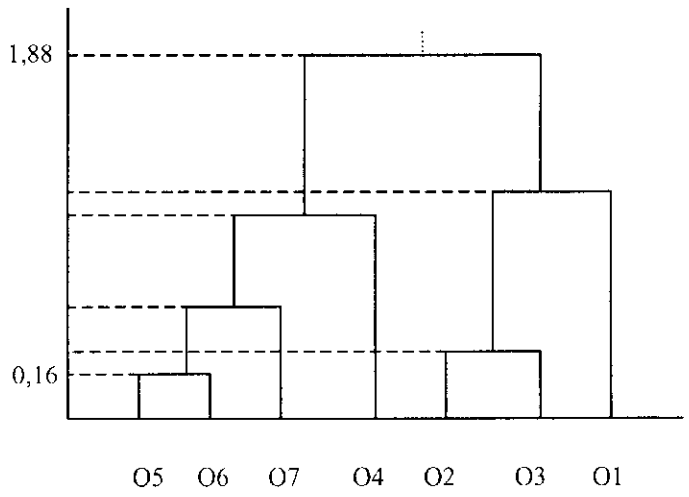
	O1	O2-O3
O2-O3	1,07	
O5-O6-O7-O4	2,17	1,58

Passo 5: agrupar (O2-O3) e O1.

	O2-O3-O1
O5-O6-O7-O4	1,88

Os clusters e os coeficientes de similaridades entre objetos e clusters, dão origem a uma representação gráfica denominada dendrograma. Na Figura 4.1, o dendrograma correspondente ao exemplo considerado.

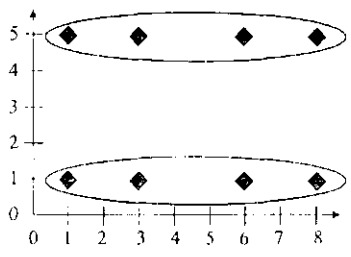
Figura 4.1: Dendrograma (Método das Médias das Distâncias).



- *Single-link*

Na técnica *Single-link*, a cada passo, são fundidos os dois clusters cujos dois membros mais próximos entre esses clusters possuem a menor distância (os dois clusters com a menor distância mínima entre si). Essa técnica gera clusters com boa *coerência local*, visto que a função de similaridade é definida localmente. Contudo, os clusters podem ser alongados ou dispersos, como mostrado na Figura 4.2. A tendência do *Single-link* de produzir este tipo de cluster alongado é chamado algumas vezes de *efeito de cadeia*, visto que ele segue uma cadeia de grandes similaridades sem levar em consideração o contexto global.

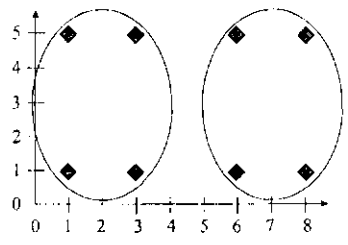
Figura 4.2: *Clustering* com *Single-link*.



- *Complete-Link*

Na técnica *Complete-link*, a cada passo, são fundidos os dois clusters cuja união possuir o menor diâmetro (os dois clusters com a menor distância máxima entre si). Sua função de similaridade é focada na qualidade *global* dos clusters, evitando clusters alongados e gerando clusters esféricos, como pode ser visto na Figura 4.3.

Figura 4.3: *Clustering* com *Complete-link*.



- *Minimun-Variance*

Com a técnica *minimum-variance*, a cada passo, são fundidos os dois clusters cuja união minimiza o aumento da variância do novo cluster, baseado na distância euclidiana entre os centróides de cada cluster.

- UPGMA

Na técnica UPGMA, a cada passo, são fundidos os dois clusters com a maior similaridade interna média (entre os documentos dentro do cluster). Esta estratégia é eficiente por evitar os clusters alongados e dispersos que ocorrem na técnica *Single-link*.

Algoritmos Hierárquicos são comumente usados por gerarem clusters de boa qualidade e por serem mais versáteis, permitindo uma navegação entre os clusters. Porém, possuem as seguintes desvantagens:

1. a complexidade de espaço é $O(n^2)$ por causa da matriz de similaridades (n x n) que deve ser armazenada, tal que n é o número de documentos;
2. a complexidade de tempo é de $O(n^2 logn)$ (Kurita, 1991);
3. dois documentos podem frequentemente ser vizinhos sem pertencerem à mesma classe semântica, ou seja, documentos de classes distintas podem ser identificados como muito similares.

Uma técnica com complexidade inferior e que é capaz de encontrar clusters de boa qualidade, é utilizada por algoritmos de particionamento.

4.2 Algoritmos de Particionamento

Além dos métodos hierárquicos, pode-se utilizar outra abordagem que também seja capaz de gerar clusters de boa qualidade. Tal abordagem pertence a uma classe de algoritmos que tenta minimizar a medida de dispersão dentro dos clusters. Inicialmente é definido o número de clusters desejados (k) e, durante o processo, cada um dos objetos é atribuído a um dos k clusters. Para minimizar a dispersão, faz-se uso, por exemplo, da soma das distâncias euclidianas da média de cada cluster.

4.2.1 K-Means

O método *K-means* (Macqueen, 1967; Hartigan & Wong, 1979), que é descrito nesta seção, pertence a essa classe de algoritmos que minimiza a dispersão dentro do *cluster*. Esse tipo de abordagem possui um alto custo computacional se for utilizado um número de *clusters* próximo da quantidade de exemplos. Para se encontrar a *melhor* solução, seria necessário testar todas os possíveis centróides (vetores representativos de cada cluster) e, nesse caso, o problema de se encontrar o melhor cluster se torna NP-Difícil (Megiddo & Supowit, 1984). Assim, costuma-se utilizar um método heurístico para se inicializar as sementes (centros iniciais de clusters), que torna o processo mais rápido e que, geralmente, produz boas (mas não necessariamente ótimas) soluções. Algumas dessas heurísticas são comentadas adiante.

O algoritmo *K-means* começa com uma partição inicial dos objetos em k clusters, com suas sementes (centróides iniciais) geradas de maneira aleatória. Passos subsequentes modificam a partição para minimizar a soma das distâncias de cada objeto à média do cluster ao qual o objeto pertence. A intenção é assinalar cada objeto à partição cuja média (centróide) está mais próxima do objeto. Esse passo gera uma nova partição em que a soma das distâncias tende a ser menor do que no particionamento anterior. O passo de redução da soma das distâncias é repetido até que essa melhora seja muito pequena ou nula. No caso da redução levar a menos do que k partições, uma das partições (geralmente a que possui a maior soma de distâncias da média) é dividida em duas ou mais até se atingir o número de k partições desejadas pelo usuário.

Como se pode perceber, o fundamento do método *K-Means* é minimizar a variância no cluster, isto é, os objetos pertencentes a um cluster precisam estar o mais próximo possível do centro do cluster (centróide). A função a ser minimizada é a Soma dos Erros Quadrados (no inglês, SSE: *Sum of the Squared Error*),

$$SSE(C) = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - C_j\|^2, \tag{4.1}$$

tal que C_j é o centróide do cluster j e x_i é o objeto em questão.

Cada iteração tenta reduzir o valor de SSE até que se torne estável, ou seja, não sofra mais alterações. É importante lembrar que para cada configuração de cluster

(posição dos centróides) existe um valor distinto para SSE. Dependendo do ponto inicial nesse espaço de solução, o algoritmo pode convergir para diferentes soluções finais.

Como a configuração inicial do cluster é importante para a execução do algoritmo, alguns cuidados são necessários e diferentes abordagens têm sido propostas para tentar minimizar esse problema:

1. utilizar um dos níveis de uma árvore, resultante de um *clustering* hierárquico, cujas ramificações produzam K partições (Milligan & Sokol, 1980);
2. uso de algoritmos genéticos para encontrar as melhores sementes (centróides iniciais) (Babu & Murty, 1993; Sarafis et al., 2002);
3. executar o algoritmo várias vezes com sementes diferentes, escolhendo a melhor solução.

Apesar do problema dos centróides iniciais, o algoritmo apresenta bons resultados e sua execução é rápida, principalmente, se comparado ao *clustering* hierárquico. Isso ocorre devido ao fato de que o cálculo de distância é feito de cada objeto x_i para cada centróide C_j, sendo que a quantidade de C_j << quantidade de x_i. Já no *clustering* hierárquico, o cálculo de distância é realizado entre todos os objetos.

O algoritmo simples do *K-Means*, dado k *(*clusters*)*, é implementado basicamente em 4 passos, descritos no Algoritmo 2.

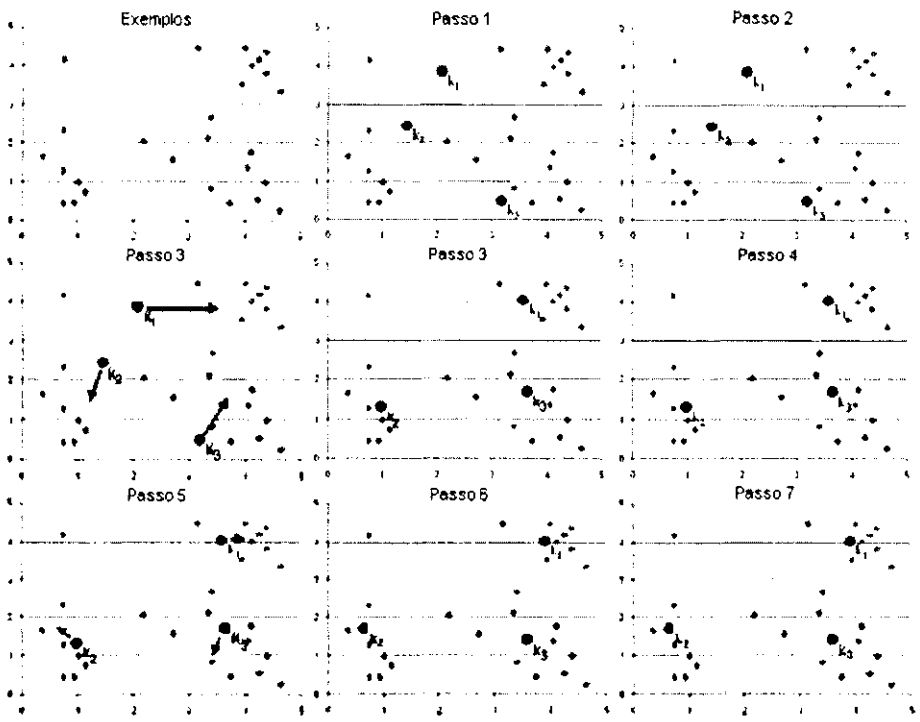
Algoritmo 2: *K-Means*

```
K-Means (k)
 Gerar randomicamente k sementes;
 Faça
   Atribuir cada exemplo ao centróide mais próximo formando k clusters;
   Recalcular centróides utilizando a média dos valores dos objetos
pertencentes ao mesmo cluster;
 Enquanto os centróides sofrerem modificação;
```

Na Figura 4.4, tem-se uma sequência de imagens que ilustram o funcionamento

do algoritmo. Nessa sequência de imagens é mostrada a identificação de 3 clusters pelo algoritmo *K-Means*.

Figura 4.4: Funcionamento do algoritmo *K-Means*.



Passo 1: inicializa os k centróides.

Passo 2: atribui cada objeto ao cluster mais próximo.

Passo 3: move cada centróide para o ponto médio do cluster.

Passo 4: re-assinala os exemplos ao centróide mais próximo.

Passos 5 e 6: re-calcula os centróides.

Passo 7: centróides estáveis. Retorna os três clusters.

Essa técnica de *clustering* é geralmente a mais citada na literatura e aplicada comercialmente. Algumas das razões por trás da sua popularidade são:

1. sua complexidade é $O(nkl)$, tal que n é o número de padrões, k é o número de clusters e l, o número de iterações que o algoritmo leva para convergir. Como o número de clusters e de iterações costuma ser substancialmente baixo (poucas

dezenas), o algoritmo acaba tendo um tempo linear, de acordo com a quantidade de padrões (Day, 1992);

2. para uma dada semente, o algoritmo gera a mesma partição de dados independente da ordem na qual os dados são apresentados.

Seus pontos fracos também são bem conhecidos:

1. o número de clusters (k) precisa ser especificado antes de se executar o algoritmo e esse valor não é óbvio na maioria dos problemas;

2. a estatística utilizada é a média dos valores de cada cluster. Os objetos individuais do cluster podem ter uma grande variância e a média pode não ser um bom indicativo do seu conteúdo;

3. se o número de clusters for da ordem de milhares, o custo computacional do algoritmo *K-Means* torna-se elevado, aproximando-se a $O(n^2)$ comparações, sendo n o número de objetos.

4. o *K-Means* clássico pode ser utilizado apenas com atributos numéricos. Porém, outras implementações permitem uso de dados categóricos;

5. os clusters resultantes podem ter tamanhos muito diferentes.

4.2.2 *Bi-Secting K-Means*

A diferença do *Bi-Secting K-Means* (Steinbach et al., 2000) para o algoritmo *K-Means* original é o passo de divisão. O algoritmo particiona um cluster em 2 sub-grupos utilizando o *K-Means* padrão. Esse processo é feito i vezes para cada cluster e o melhor resultado é mantido. O processo se repete até se alcançar o número desejado de clusters. O algoritmo trabalha da seguinte forma:

1. selecionar um cluster para ser dividido.

2. encontrar 2 sub-clusters usando o algoritmo do *K-Means* (passo do *Bi-secting*).

3. repetir o passo 2 i vezes e utilizar a divisão que gerou os 2 clusters com maior valor de similaridade total [1].

4. repetir os passos 1, 2 e 3 até que o número desejado de clusters seja alcançado.

O cluster que será particionado pode ser escolhido como o maior cluster a cada passo, o que possui menor similaridade total, ou ambos.

As duas principais vantagens sobre o *K-Means* padrão são que ele tende a produzir clusters de tamanho uniforme e os clusters resultantes possuem menor medida de entropia.

4.2.3 Spherical K-Means

Essa variante do algoritmo *K-Means*, desenvolvida por Dhillon & Modha (Dhillon & Modha, 2001), utiliza a medida do co-seno ao invés da medida de distância Euclidiana. O *Spherical K-Means* particiona uma esfera de alta dimensão usando uma coleção de grandes hiper-esferas. O algoritmo computa uma partição disjunta dos vetores de documento e, para cada partição, computa um centróide com norma Euclidiana. Estes centróides normalizados podem conter informação semântica valiosa sobre os clusters e são chamados de *vetores de conceito*.

Esse algoritmo possui algumas vantagens do ponto de vista computacional: como utiliza a medida do co-seno, consegue explorar melhor a esparsidade dos dados; pode ser paralelizado de forma eficiente; converge rapidamente para um ótimo local e os vetores de conceito podem servir como classificadores (Dhillon & Modha, 2001).

4.3 Algoritmos Multi-Visão

Algoritmos de Multi-visão treinam duas hipóteses independentes provendo, uma à outra, rótulos para os dados sem rótulo. Os algoritmos de treinamento tendem a maximizar a concordância entre as duas hipóteses independentes. Dasgupta e colegas

[1] *Overall Similarity* é uma medida de qualidade interna que mede a coesão dos objetos contidos em um cluster.

(Dasgupta et al., 2001) mostraram que a discordância entre duas hipóteses independentes é um limite superior na taxa de erro de uma hipótese, explicando a boa qualidade dos resultados da aplicação de aprendizagem multi-visão. Bickel (Bickel & Scheffer, 2004) aplica esta abordagem a dois métodos de *clustering* comuns da literatura: aglomerativo hierárquico e *K-Means*, descritos a seguir.

4.3.1 Multi-View Agglomerative Clustering

O clustering aglomerativo é baseado em uma medida de distância entre clusters. Na configuração de multi-visão tem-se dois conjuntos de atributos $V^{(1)}$ e $V^{(2)}$ e duas medidas de distância $d^{(1)}C_iC_j$ e $d^{(2)}C_iC_j$. O *clustering* aglomerativo de uma única visão inicia com cada exemplo em seu próprio cluster. Então, funde-se iterativamente os clusters, de acordo com uma função de distância, construindo um dendrograma. Já o *clustering* aglomerativo multi-visão funde os clusters mais próximos intercalando as visões. Assim, cria-se um novo cluster de acordo com os documentos mais próximos na visão 1. Repete-se o processo como um novo par de documentos mais similares na visão 2. Volta-se para a visão 1 e continua-se o processo. As operações de fusão trabalham em um dendrograma combinado para ambas as visões, resultando em um dendrograma final.

4.3.2 Multi-View Spherical K-Means

Existe um problema que ocorre com documentos que possuem igual composição de palavras mas com diferentes freqüências. Isto leva a diferentes modelos que informam a probabilidade de um documento ser assinalado a um determinado cluster. Este problema pode ser superado normalizando cada vetor de documento em relação ao seu comprimento. Um algoritmo de *clustering* capaz de trabalhar com vetores normalizados é o *Spherical K-Means*.

Tanto os vetores de conceito (centróides - $c_j^{(v)}$) quanto os vetores de exemplo ($x_i^{(v)}$), tal que v é o número de visões, têm comprimento igual a 1. Começa-se o algoritmo com vetores de conceito inicializados randomicamente $c_j^{(2)}$. Um passo de expectativa assinala os documentos que são mais próximos do vetor de conceito $c_j^{(v)}$ para a partição correspondente $p_j^{(v)}$. A seguir, um passo de maximização computa os novos vetores de

conceito (parâmetros do modelo).

Após realizar os passos de maximização e expectativa em uma visão da partição $p_j^{(v)}$, realiza-se os passos de maximização e de expectativa na outra visão. A cada iteração computa-se uma função objetivo para cada visão. Finaliza-se o processo de otimização se a função objetivo não alcançar um novo mínimo após um determinado número de iterações em cada visão.

Terminado o processo, as partições $p_j^{(1)}$ e $p_j^{(2)}$ não contêm necessariamente os mesmos exemplos. Para obter um resultado combinando as partições é necessário assinalar cada exemplo a um cluster distinto que é determinado pelo vetor de conceito mais próximo. Para fazer isto, computa-se uma média de consenso para cada cluster e visão. Apenas os exemplos em que ambas as visões concordam são incluídos no cluster.

No final, assinala-se cada exemplo ao cluster que provê o vetor de consensos mais semelhante. Este vetor é determinado calculando-se a média do co-seno em ambas as visões.

4.4 Redes Neurais Artificiais

Redes Neurais Artificiais (RNA) são sistemas computacionais baseados no sistema nervoso biológico, que usam um grande número de neurônios artificiais simples e interconectados entre si, para tentar reproduzir algumas habilidades do cérebro como reconhecimento de imagens, sons e agrupamento de informações.

Os elementos básicos de processamento são os neurônios artificiais, que recebem vetores de entrada como treinamento, nos quais cada vetor é ponderado pelos pesos da conexão de entrada correspondente e possui uma saída calculada por uma função de ativação, geralmente não linear. Inicialmente os pesos são definidos de forma aleatória, por por alguma função definida pelo usuário, e seu ajuste durante o processo de treinamento corresponde à forma pela qual a rede aprende e representa seu conhecimento, ou seja, o conhecimento é armazenado nos pesos das conexões e não nos neurônios. Se o ajuste de pesos for realizado comparando-se a saída da rede com o resultado esperado para um determinado padrão de entrada, diz-se que esse é um modelo de aprendizagem supervisionada. Porém, se o ajuste de pesos for realizado apenas tendo como base os

padrões de entrada, sem que haja qualquer referencial de saída esperada, diz-se que esse é um modelo de aprendizagem não supervisionada. Como, na prática, o número de grupos é desconhecido, o treinamento é feito sob o paradigma não supervisionado, dificultando a definição da topologia da rede, isto é, a quantidade de neurônios que ela deve ter. Esse problema ocorre em RNAs de tamanho fixo (Kohonen, 1990; Kohonen et al., 1996; Kohonen, 2001). Para resolver essa situação, foram criadas redes de tamanho variável, capazes de inserir (Alahakoon et al., 1998) ou remover neurônios (Martinetz & Schulten, 1994), ou ambos (Fritzke, 1995), além de poderem criar mapas hierárquicos (Dittenbach et al., 2000; Rauber et al., 2002). Nesta seção são apresentadas duas dessas técnicas: uma estática e uma construtiva hierárquica.

4.4.1 Mapas Auto-Organizáveis (SOM)

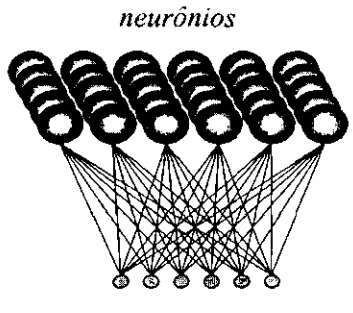
Esta seção, baseada em (Haykin, 1999), trata de um modelo de Redes Neurais Artificiais comumente utilizado em aplicações de *clustering*. Os Mapas Auto-Organizáveis (*Self-organizing Maps* - SOM)(Kohonen, 1990; Kohonen et al., 1996; Kohonen, 2001) são redes neurais artificiais competitivas, capazes de formar mapeamentos que tentam preservar a distribuição entre os espaços de entrada e de saída. Sua inspiração é biologicamente baseada na idéia de que uma organização de alto nível pode ser criada no cérebro durante o aprendizado por meio de uma auto-organização e a representação do conhecimento em categorias pode assumir a forma de um mapa de características, organizado geometricamente sobre partes correspondentes do cérebro. Desse modo, SOM mapeiam um conjunto de objetos de entrada, cada um representado por um vetor n-dimensional, em uma malha ou um mapa de nodos, revelando a frequência e distribuição dos dados nesse arranjo espacial de nodos. Portanto, objetos semelhantes são posicionados proximamente, refletindo a estrutura interna desses dados de entrada, preservando boa parte dos relacionamentos de distâncias entre os objetos.

Geralmente, agrupa-se os neurônios de uma rede SOM em uma malha bidimensional (Figura 4.5), mas dimensões maiores podem ser utilizadas, bem como uma rede unidimensional. Um mapa quadrado ou retangular de neurônios (superior), onde cada um possui um vetor de pesos (características) do mesmo tamanho do vetor de entrada (inferior), que é conectado a todos os neurônios da rede.

O funcionamento de uma rede SOM é caracterizado por 3 etapas: competição,

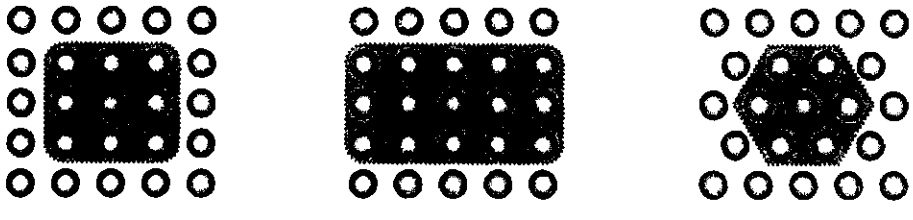
cooperação e adaptação. Quando uma entrada é apresentada à rede, os neurônios de saída **competem** entre si para serem ativados. Essa competição é feita medindo-se a similaridade do padrão de entrada com todos os vetores de pesos dos neurônios da rede, utilizando-se uma função de distância, que geralmente é a distância euclidiana. O neurônio que vence tal competição é chamado neurônio vencedor e possui a menor distância do padrão de entrada, ou seja, é o neurônio que mais se assemelha ao objeto. Definido o neurônio vencedor tem-se início a etapa de **cooperação,** onde se determina uma vizinhança em torno desse neurônio vencedor, na qual haverá a cooperação de neurônios para responder ao objeto que foi apresentado à rede, ou seja, essa região (não apenas o neurônio vencedor) é a que melhor responderá ao estímulo desse objeto. Ocorre então a fase de **adaptação**. Nesse processo, apenas o neurônio vencedor e seus vizinhos dentro de um certo raio ou área de vizinhança terão seus pesos atualizados, aumentando o grau de similaridade entre tais neurônios e a entrada. Cada vez que um novo padrão de treinamento é apresentado à rede, os neurônios competem entre si para ver qual deles responde melhor ao estímulo. Devido ao ajuste dos pesos do neurônio vencedor e de seus vizinhos topológicos, dois neurônios fisicamente próximos na rede representarão padrões similares do conjunto de dados de entrada.

Figura 4.5: Estrutura de uma rede do tipo SOM.



Para que a construção desse mapa seja realizada de maneira adequada, o processo de cooperação exige que a vizinhança do neurônio vencedor também sofra uma adaptação, como visto anteriormente. Essa vizinhança pode ter formas do tipo quadrada, retangular ou hexagonal, em torno do neurônio vencedor (Figura 4.6).

Figura 4.6: Tipos de vizinhança em uma rede do tipo SOM.



A atualização dos pesos é diferente para o neurônio vencedor e para a vizinhança. O vencedor tem seus pesos ajustados mais fortemente movendo esse vetor de pesos na direção do vetor de entrada, tornando seu aprendizado, ou seja, sua adaptação ao vetor de entrada, maior, enquanto que os neurônios da vizinhança têm um aprendizado menor. Durante o treinamento, a taxa de aprendizado e o raio da vizinhança são continuamente decrementados. Essas duas variáveis podem ser decrementadas linearmente, exponencialmente ou utilizando-se alguma outra função.

Devido aos ciclos de apresentação dos dados de treino, os vetores de pesos dos neurônios da rede tendem a seguir a distribuição dos padrões de entrada, graças a atualização da vizinhança. Isso gera uma ordenação topológica do mapa de características no espaço de entrada, pois os neurônios que são adjacentes na malha tendem a ter vetores de pesos similares.

A adaptação mencionada anteriormente fica visível na Figura 4.7, onde uma rede SOM tenta se ajustar aos exemplos de treinamento, fazendo com que a rede se espalhe. Nesse exemplo os pesos dos neurônios são inicializados aleatoriamente, como pode ser visto no primeiro quadro, que corresponde à época 0. À medida que o treinamento prossegue, o mapa vai se ajustando aos exemplos de treinamento até a iteração final estabelecida pelo usuário.

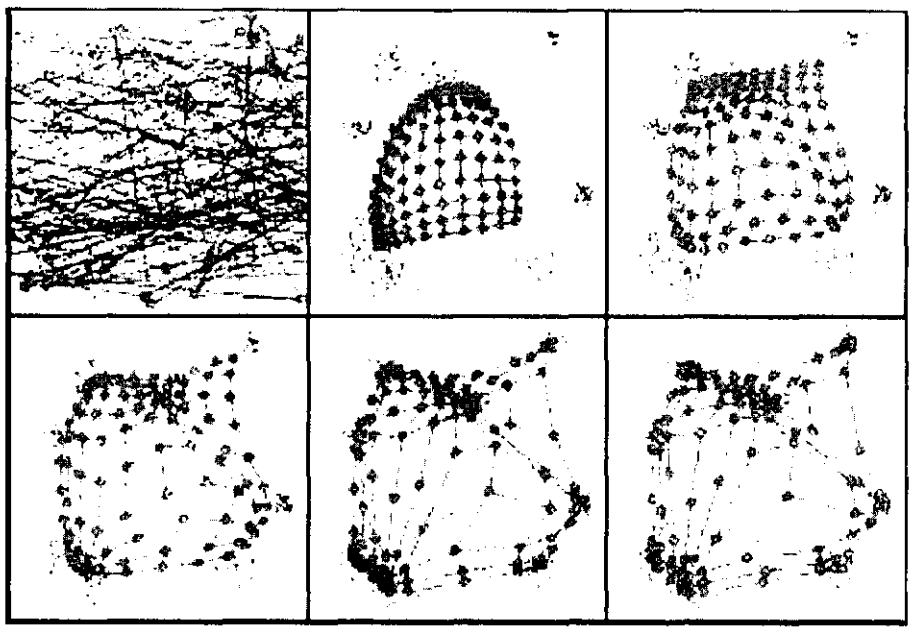
O treinamento de redes tipo SOM é relativamente simples e pode ser visto no Algoritmo 3. A complexidade do método padrão é dada em função do número de cálculos de distância:

$$NCD = NU \times NP \times NC,$$

tal que NU é o número de unidades (neurônios) da rede, NP é o número de padrões apresentados e NC é o número de ciclos. Ou seja, para cada padrão apresentado à

rede, são realizados NU cálculos de distância para se encontrar o neurônio vencedor. O processo é repetido para cada ciclo, causando um alto custo computacional. A vantagem do uso de redes do tipo SOM em *clustering* de documentos é exatamente o mapa gerado, que pode ser utilizado para exibir graficamente as relações entre documentos próximos e a definição de áreas entre documentos distantes.

Figura 4.7: Treinamento de uma rede SOM.



Algoritmo 3: SOM

```
SOM()
 Converter os exemplos I_i em vetores X_i;
 Inicializar os m_j vetores de pesos, associados a cada nodo de saída,
com valores aleatórios.
 Faça
   Apresentar um vetor de entrada;
   Computar a distância d_ij entre o vetor de entrada X_i e cada vetor de
peso m_j para encontrar o nodo vencedor;
   Atualizar os pesos no nodo vencedor e na vizinhança;
 Enquanto o limite de épocas não foi atingido.
```

- **WEBSOM**

WEBSOM é uma ferramenta de navegação e recuperação de informação baseada em SOM que foi aplicada para organizar documentos de *Newsgroups* para facilitar sua navegação e exploração (Honkela et al., 1996). Apesar de ter sido utilizada nesse domínio, essa ferramenta pode ser aplicada em qualquer coleção de documentos.

A arquitetura do WEBSOM é composta hierarquicamente por duas redes SOM inter-relacionadas (Kaski et al., 1996), associadas ao Mapa de Categoria de Palavras e ao Mapa de Documentos, descritos a seguir:

1. mapa de Categorias de Palavras: descreve as relações entre as palavras dos documentos, que foram categorizadas em clusters, utilizando um mapa semântico de palavras (seção 3.2.2);

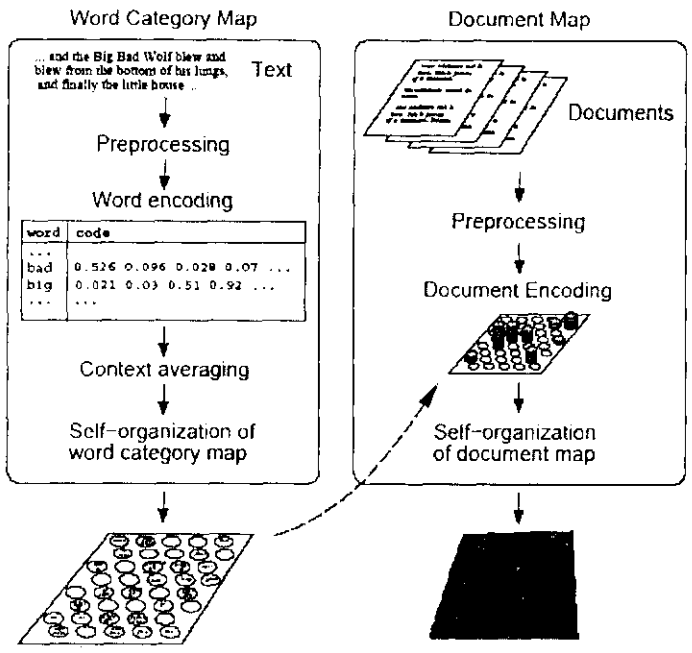
2. mapa de Documentos: utiliza o mapa de categorias para criar um mapa refletindo as relações entre documentos.

Essa arquitetura pode ser melhor entendida observando-se a Figura 4.8, apresentada por Honkela et. al. Apesar de ser realizada em duas etapas, a construção final do mapa de documentos é substancialmente mais rápida por utilizar o mapa de palavras que reduz consideravelmente o tamanho do vetor de características dos documentos e, conseqüentemente, o tempo de processamento do algoritmo (Honkela et al., 1996).

A etapa de pré-processamento realiza a remoção de *stopwords*, figuras e o tratamento de expressões numéricas e de códigos especiais com regras heurísticas. Além disso, é executado o processo de *stemming* para gerar o vocabulário final.

A codificação do documento é feita mapeando-se cada uma das palavras do texto no mapa de categorias, formando um histograma das colisões (cada vez que um neurônio responde pela entrada de uma palavra). Utiliza-se uma *Gaussiana* para reduzir a sensitividade do histograma a pequenas variações no conteúdo do documento.

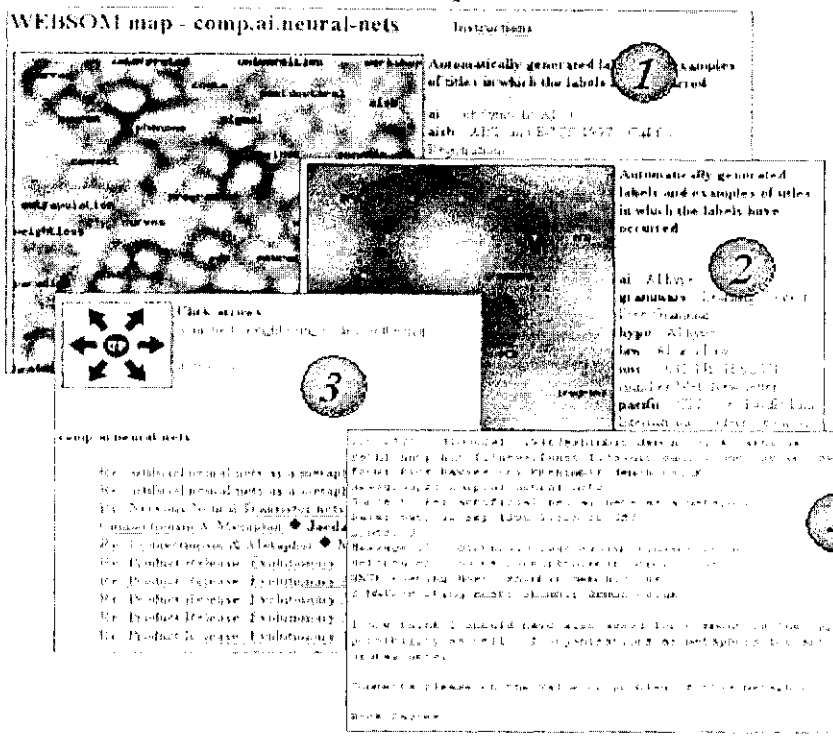
Figura 4.8: Arquitetura básica do WEBSOM.



Após isso, os padrões de entrada (documentos codificados) são utilizados para o treinamento da rede SOM. Finalizado o treinamento, o mapa obtido fornece uma informação visual de como se apresenta a organização espacial dos documentos e pode ser utilizado para sua exploração e navegação. Esse mapa é extremamente útil quando o usuário tem pouco conhecimento sobre o domínio ou sobre o conteúdo dos documentos, como pode ser visto na Figura 4.9.

O item 1 é o mapa gerado pela aplicação. As partes mais claras indicam alta concentração de documentos, enquanto as partes escuras correspondem a espaços vazios. Se fosse mapeado para 3D, as partes claras seriam cumes e as escuras vales. Ao se clicar no mapa é gerada uma nova imagem, com uma aproximação na região clicada (item 2). Quando uma aproximação não é mais possível, é exibida a lista de documentos do (item 3) e, por fim, pode-se ter acesso aos documentos (item 4) por meio dos *links* do item 3.

Figura 4.9: Exemplo do WEBSOM.

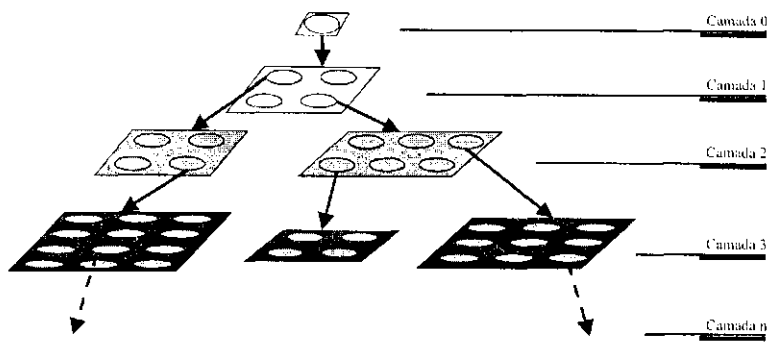


4.4.2 Growing Hierarchical Self-Organizing Map *(GHSOM)*

Apesar da estabilidade e popularidade do SOM, pelo menos duas limitações devem ser citadas: a arquitetura estática deste modelo e a capacidade limitada para a representação de relações hierárquicas dos dados. O *Growing Hierarchical Self-Organizing Map* (GHSOM) (Dittenbach et al., 2000; Rauber et al., 2002) supera essas limitações . O GHSOM é um modelo de rede neural artificial com arquitetura hierárquica composta de SOMs independentes. Uma navegação através das ramificações é facilitada graças aos SOMs independentes nas camadas individuais da hierarquia (ver Figura 4.10).

Um dos problemas do SOM é justamente sua arquitetura fixa que tem que ser definida *a-priori*. Variantes do SOM que crescem dinamicamente, por outro lado, tendem a produzir mapas enormes que são difíceis de controlar. O GHSOM cresce de um modo vertical, de acordo com a distribuição dos dados, permitindo uma navegação hierárquica em sub-conjuntos dos dados, e de um modo horizontal, significando que o tamanho de cada mapa bi-dimensional individual, se adapta às exigências da dimensão (quantidade de atributos) dos exemplos de treinamento.

Figura 4.10: Arquitetura de uma GHSOM treinada.



A idéia básica é que cada camada do GHSOM é responsável por explicar alguma porção do desvio dos dados de entrada como apresentado em sua camada anterior. Isto é feito acrescentando neurônios aos SOMs em cada camada até que seja alcançado um tamanho satisfatório para o mapa. Mais precisamente, permite-se que os SOMs em cada camada cresçam até que o desvio presente no neurônio (na camada anterior) correspondente ao SOM da camada atual, seja reduzido a pelo menos uma porcentagem fixa definida pelo usuário.

O processo de treinamento e de inserção de neurônios nos SOMs recentemente criados ocorre apenas em uma fração dos dados de entrada. Tal fração é representada pelos dados agrupados no neurônio correspondente na camada anterior. A estratégia para inserção de linha ou coluna, como também o critério de parada, é essencialmente igual ao usado para o primeiro mapa de camada. O mesmo procedimento é aplicado para qualquer camada subseqüente do GHSOM.

O processo de treinamento do GHSOM é finalizado quando nenhum neurônio necessitar de expansão adicional. Este processo de treinamento não conduz, necessariamente, a uma hierarquia equilibrada, ou seja, uma hierarquia com profundidade igual em cada ramo. Pode ocorrer de alguns clusters serem mais densos (neurônios com muitos exemplos) do que outros e precisarem se ramificar mais profundamente.

4.5 Considerações Finais

Neste capítulo foram apresentados alguns dos principais algoritmos de *clustering*, bem como suas vantagens e desvantagens. Devido à menor complexidade, o algoritmo *K-Means* pode ser mais eficiente, em relação ao custo computacional, se comparado aos demais algoritmos. Porém, como o produto da FIP é um mapa, uma técnica interessante a ser empregada é o SOM. Visto que essa última possui alto custo computacional, existe a necessidade de se encontrar uma maneira de aumentar seu desempenho. Isto pode ser feito pela redução de dimensionalidade dos dados, como no mapa de categorias de palavras do WEBSOM.

Para *clustering* de documentos, os algoritmos mais indicados são os que permitem tratamento de dados de alta dimensão, como os hierárquicos e os particionadores, que são os mais citados na literatura. Entretanto, outras técnicas menos usadas para documentos, mas que suportem alta dimensionalidade, também podem ser eficientes no tratamento desse tipo de dados, como algoritmos de redes neurais artificiais.

CAPÍTULO

5

Experimentos e Resultados

Neste capítulo são apresentadas as ferramentas, atributos de configuração e os conjuntos de dados (*datasets*) utilizados para realizar os experimentos durante este trabalho mestrado, bem como os resultados obtidos e comentários. Tais experimentos foram feitos utilizando o cluster do LABIC (nós com Intel Xeon 2.8 GHz) com OpenMosix e compilador GCC 3.3. Foram utilizadas as versões para Linux dos softwares escolhidos para os experimentos de modo que pudessem ser executadas no cluster do LABIC.

5.1 Descrição Geral dos Experimentos

Nesta seção é apresentada uma descrição geral dos experimentos, descrevendo as ferramentas, os conjuntos de dados e a etapa de pré-processamento.

5.1.1 Ferramentas

As ferramentas de Aprendizado de Máquina não supervisionado, em nosso caso os algoritmos de *clustering*, utilizadas para o agrupamento dos conjuntos de exemplos (documentos) são os *frameworks* CLUTO e gCLUTO e os softwares Gmeans, Co-*K-Means*, Som_Pak e GHSOM, descritos a seguir. Quando a ferramenta permite,

optou-se pelo uso da medida do coseno, visto que esta apresentou melhores resultados em testes previamente realizados.

- O *framework* CLUTO

Disponível gratuitamente no site do grupo desenvolvedor[1], fornece vários algoritmos de *clustering* e várias opções de métrica de similaridade e de funções de otimização (Karypis, 2002).

Os algoritmos e funções utilizados nos experimentos são descritos a seguir:

RB: neste método, a solução desejada é computada, realizando-se uma seqüência de k-1 *repeated bisections*. Nesta abordagem, a representação vetorial (tabela atributo-valor) é inicialmente particionada em dois clusters, então um desses clusters é selecionado e particionado. O processo continua até se encontrar o número desejado de clusters. A cada passo, o cluster que sofre a divisão resulta em dois novos clusters, otimizando *localmente* uma determinada função.

RBR: este método utiliza uma solução obtida pelo método RB e tenta otimizá-la globalmente de acordo com uma determinada função de critério.

AGGLO: neste método, a solução é computada usando o paradigma aglomerativo para otimizar localmente uma função de critério. Tal solução é obtida quando restam apenas k clusters.

BAGGLO: nesta abordagem, o processo de aglomeração é guiado por uma solução encontrada previamente pelo uso do método RB. Assim, o *bagglo*, como pode ser visto nos experimentos, costuma encontrar clusters de qualidade superior ao *agglo*.

As funções de otimização que o CLUTO fornece são mostradas a seguir, sendo k o número total de clusters, C o número de exemplos a serem agrupados, C_j o conjunto de exemplos assinalados ao cluster j, n_j o número de objetos no cluster j, x e y representam dois exemplos quaisquer e $s(x, y)$ é a similaridade entre esses dois exemplos. Informações mais detalhadas sobre cada uma das seguintes funções são encontradas em (Zhao & Karypis, 2002).

[1]http://www-users.cs.umn.edu/~karypis/cluto/

Tabela 5.1: Funções de otimização oferecidas pelo CLUTO.

i1:	maximiza a soma das similaridades médias entre pares de documentos (de acordo com a medida do coseno) assinalados a cada cluster, ponderado pelo tamanho de cada cluster.
i2:	maximiza a similaridade entre cada documento e o centróide do cluster ao qual foi assinalado.
e1:	minimiza o coseno entre o centróide de cada cluster e o centróide da coleção de documentos, tentando uma boa separação dos grupos.
g1:	minimiza o *edge-cut* de cada uma das partições.
h1:	maximiza i_1/e_1.
h2:	maximiza i_2/e_1.
slink:	função *single-link* tradicional em que, a cada passo, são fundidos os dois clusters cujos dois objetos mais próximos entre esses clusters possuírem a menor distância (os dois clusters com a menor distância mínima entre si).
wslink:	função *single-link* ponderada por cluster. As funções ponderadas por cluster encontram uma solução a partir de uma já existente, sendo que o peso do *link* entre dois clusters C_i e C_j é calculado como a similaridade entre os objetos de C_i e C_j dividido pela similaridade total entre os objetos em $C_i \cup C_j$.
clink:	função *complete-link* tradicional em que, a cada passo, são fundidos os dois clusters cuja união possuir o menor diâmetro (os dois clusters com a menor distância máxima entre si).
wclink:	função *complete-link* ponderada por cluster.
upgma:	função UPGMA tradicional em que, a cada passo, são fundidos os dois clusters cuja similaridade média, entre todos os documentos de cada cluster, seja maior.
wupgma:	função UPGMA ponderada por cluster.

A execução desta ferramenta em nossos experimentos foi realizada utilizando a configuração padrão, modificando apenas o método de *clustering* e a função de otimização, como segue:

```
#./vcluster -clmethod=Método -crfun=Função -rclass=ArqClasses Matriz
    NClusters
```

Método: rb, rbr, agglo ou bagglo.

Função: i1, i2, e1, g1, h1, h2, slink, wslink, clink, wclink, upgma ou wupgma.

ArqClasses: nome do arquivo texto que contém a correta classificação dos exemplos, na mesma ordem em que aparecem na Matriz.

Matriz: nome do arquivo contendo a tabela atributo/valor, sendo que a primeira linha informa a quantidade de objetos e a quantidade de atributos. Cada uma das demais linhas representa um objeto, com os atributos separados por um espaço em branco.

NClusters: número de clusters desejados.

A geração dos mapas em 3D foi realizada utilizando uma implementação gráfica do CLUTO, chamada gCLUTO[2]. Nesta forma de representação topográfica (em montanhas), cada cluster é representado como um pico no terreno em 3D. A localização, o volume, a altura e a cor da montanha fornecem informações sobre o cluster associado a ela. A forma da montanha é uma curva gaussiana, usada como uma estimativa da distribuição dos dados dentro de cada cluster. A altura da montanha é proporcional à similaridade interna do cluster. Assim, mesmo um cluster com pureza elevada pode ter altura baixa e um cluster com pureza mediana pode ter altura elevada. O volume da montanha é proporcional ao número de elementos dentro do cluster. Finalmente, a cor do topo da montanha é proporcional ao desvio padrão interno do cluster, sendo que vermelho indica baixo desvio, enquanto que azul indica um alto desvio (Documentação do gCluto). Porém, analisando os mapas e as tabelas, percebemos que a essa informação está invertida, ou seja, vermelho indica alto desvio e azul indica baixo desvio.

[2]http://www-users.cs.umn.edu/~karypis/cluto/gcluto/download.html

É importante mencionar, ainda, que quando duas montanhas tentam ocupar um mesmo espaço, o software faz a correção por meio da criação de uma única montanha com dois picos, volume e altura maiores. Porém, as cores dos picos são alteradas. Em geral, as cores descem um ou mais níveis no espectro: vermelho se torna laranja, laranja se torna amarelo, amarelo se torna amarelo esverdeado, ..., verde azulado se torna azul. Assim, um pico que inicialmente era vermelho pode se tornar amarelo, gerando um gráfico que não corresponde exatamente ao resultado do *clustering*. Isso será indicado quando necessário.

- *Spherical K-Means* (*Gmeans*)

Disponível gratuitamente no site do grupo desenvolvedor[3], o Gmeans é uma implementação do algoritmo *K-Means*, porém utilizando a medida do cosseno para calcular a similaridade entre os exemplos (Dhillon & Modha, 2001). Um de seus parâmetros de configuração é o método de inicialização dos centróides:

random ID: assinala aleatoriamente cada exemplo a um cluster;

random perturb: gera o centróide para todo o conjunto de dados e, então, de acordo com um grau de perturbação, modifica valores deste centróide para que ele seja movimentado no espaço;

initializing for a file: lê o cluster a que corresponde cada exemplo a partir um arquivo texto e calcula os centróides de acordo com os exemplos contidos em cada cluster;

random seed: seleciona aleatoriamente exemplos do conjunto de dados como centróides;

farthest_ 0: seleciona o primeiro centróide aleatoriamente do conjunto de exemplos. Em seguida, seleciona o exemplo mais distante dos centróides já escolhidos, repetindo o processo até selecionar os k centróides;

farthest_ 1: similar ao *farthest_ 0*, porém o primeiro centróide escolhido é o mais distante do centróide correspondente a todo o conjunto de exemplos.

[3]http://www.cs.utexas.edu/users/yguan/datamining/gmeans.html

Duas opções são utilizadas nos experimentos: *random seed* e *random ID*. Em alguns *datasets* a função *random seed* não consegue gerar bons centróides e o algoritmo não converge. Quando isso ocorre, passa-se a utilizar a função *random ID*, que converge mas não encontra clusters de boa qualidade. A função utilizada em cada experimento é indicada no texto.

O Gmeans foi executado em sua configuração padrão, modificando apenas a inicialização dos centróides, como segue:

```
#./gmeans -i InitCentróide -T ArqClasses -F t Matriz -c NClusters
```

InitCentróide: método de inicialização dos centróides (c = *random seed* e r = *random ID*).

ArqClasses: nome do arquivo texto que contém a correta classificação dos exemplos, na mesma ordem em que aparecem na Matriz.

Matriz: nome do arquivo contendo a tabela atributo/valor, sendo que a primeira linha informa a quantidade de objetos e a quantidade de atributos. Cada uma das demais linhas representa um objeto, com os atributos separados por um espaço em branco.

NClusters: número de clusters desejados.

- Co-*K-Means* (*Multi-View Clustering*)

Implementação do algoritmo *Spherical K-Means* capaz de trabalhar com duas visões simultaneamente para agrupar conjuntos de exemplos (Bickel & Scheffer, 2004). Após contato por correio eletrônico, esta ferramenta foi cedida gratuitamente por Stefen Bickel.

O Co-*K-Means* foi executado em sua configuração padrão, escolhendo-se o algoritmo e o número de visões, como segue:

```
#./clustering -rn ArqSaida -v Verbose -alg Algoritmo -nv NVisões
    -f Matriz -k NClusters
```

ArqSaida: arquivo no qual será gravado o ID do cluster em que cada um dos exemplos foi assinalado.

Verbose: nível de detalhe das informações exibidas pela ferramenta.

Algoritmo: 1 = *K-Means*.

NVisões: Número de visões (1 ou 2). A primeira visão equivale aos atributos da posição 1 até 99.999. Já a segunda visão vai de 100.000 a 199.999.

Matriz: nome do arquivo contendo a tabela atributo/valor em um formato de matriz esparsa definido como: Classe Coluna_do_Atributo:Valor_do_Atributo. Exemplo:

```
0 52:1.97 86:1.85 273:1.71 ...
⋮
1 27:1.85 36:1.91 110:1.80 ...
⋮
2 20:3.81 65:2.05 171:3.70 ...
```

NClusters: número de clusters desejados.

Todos os algoritmos comentados até o momento utilizam a medida do coseno para avaliar a similaridade dos objetos.

Para análise subjetiva, os algoritmos utilizados são derivados do SOM e usam medida de distância Euclidiana.

- Som_Pak

Implementação do algoritmo de Mapas Auto-Organizáveis (Kohonen et al., 1996), disponível gratuitamente no site do grupo desenvolvedor[4]. A execução do Som_Pak é feita em 5 etapas:

1. Criação do Mapa (ferramenta *randinit*): definição das dimensões, topologia, função de atualização da vizinhança, inicialização dos pesos e do gerador de números aleatórios;

[4]http://www.cis.hut.fi/research/som_pak/

2. Etapa Inicial do Treinamento (ferramenta *vsom*): etapa inicial de adaptação do mapa aos padrões de entrada. Definição da quantidade de épocas de treinamento, razão de aprendizagem e raio da vizinhança;

3. Etapa Final do Treinamento (ferramenta *vsom*): etapa de ajuste fino do mapa aos padrões de entrada. Definição da quantidade de épocas de treinamento, razão de aprendizagem e raio da vizinhança. Nesta etapa a razão de aprendizagem e o raio da vizinhança são menores para se procurar uma estabilização do mapa. Por isto é necessário um número maior de épocas de treinamento;

4. Cálculo do Erro (ferramenta *qerror*);

5. Criação da U-Matrix (ferramentas *vcal* e *umat*);

A configuração desta ferramenta para uso em todos os experimentos é descrita na Tabela 5.2 a seguir.

Tabela 5.2: Configuração do SOM

Criação do Mapa	**Etapa Inicial do Treinamento**
Vetores de pesos: inicializados aleatoriamente	**Épocas:** 100
Dimensão do mapa: a quantidade de neurônios é	**Razão de Aprendizagem (alpha):** 0.05
cerca de 10% da quantidade de documentos	**Raio Inicial da Vizinhança:** metade da
Topologia: hexagonal	altura do mapa
Função de atualização da vizinhança: bubble	
Parâmetro do gerador de números aleatórios: 123	**Etapa Final do Treinamento**
	Épocas: 100
	Razão de Aprendizagem (alpha): 0.02
	Raio Inicial da Vizinhança: 1

O Som_Pak foi executado em sua configuração padrão, adotando as opções descritas na Tabela 5.2:

```
#./randinit -din Matriz -cout ArqCod -xdim Xdim -ydim Ydim
     -topol Topologia -neigh Vizinhança -rand Semente

#./vsom -din Matriz -cin ArqCod -cout ArqCod -rlen Épocas -alpha Alpha
     -radius RaioV

#./qerror -din Matriz -cin ArqCod
```

```
#./vcal -din Matriz -cin ArqCod -cout ArqCod

#./umat -cin ArqCod -portrait -ps 1 -fontsize TamFonte -average Avg
```

Matriz: nome do arquivo contendo a tabela atributo/valor, sendo que a primeira linha informa a quantidade de exemplos e cada uma das demais linhas representa um exemplo, com os atributos separados por um espaço em branco.

ArqCod: arquivo *codebook* que contém a rede neural que está sendo treinada.

Xdim e Ydim: dimensões do mapa, preferenciando um mapa retangular.

Topologia: hexa = Hexagonal e rect = Retangular.

Vizinhança: método de atualização dos pesos dos vizinhos dos neurônios vencedores (bubble = Linear e gaussian = Gaussiana).

Semente: número utilizado como semente para o gerador de números aleatórios.

Épocas: número de repetições necessárias para o treinamento da rede.

Alpha: valor da razão de aprendizagem.

RaioV: raio inicial da vizinhança.

Avg: o valor 1 indica que a ferramenta deve calcular a média dos valores do mapa para suavizar a coloração do mesmo.

- *Growing Hierarchical* SOM (GHSOM)

Disponível gratuitamente no site do grupo desenvolvedor[5], esta ferramenta é capaz de gerar SOMs convencionais, ou seja, de tamanho fixo, *growing* SOMs, que são mapas SOM cuja dimensão sofre incremento ao longo do processo de treinamento, e *growing hierarchical* SOM, que são mapas SOM conectados hierarquicamente como níveis de detalhamento de um mapa principal (superfície) (Dittenbach et al., 2000; Rauber et al., 2002). A ferramenta SOM_PAK foi usada para gerar mapas convencionais e a GHSOM para gerar mapas hierárquicos. Os parâmetros principais do *arquivo de configuração* são descritos a seguir:

[5]http://www.ifs.tuwien.ac.at/~andi/ghsom/download.html

EXPAND_CYCLES: número de épocas de treinamento necessárias para que o mapa seja verificado para uma eventual expansão;

MAX_CYCLES: número máximo de épocas de treinamento;

TAU_1: porcentagem do erro restante que tem que ser explicado por cada mapa, ou seja, o critério de parada para o crescimento horizontal. Quanto menor este valor, maior será o mapa, e mais achatada será a hierarquia;

TAU_2: grau final de granularidade representado pelos mapas na camada mais baixa. Quanto menor, mais detalhada será a respresentação de dados e maior será a estrutura total de GHSOM;

INITIAL_X_SIZE e INITIAL_Y_SIZE: dimensões iniciais do mapa, preferenciando um mapa retangular.

Existem ainda parâmetros como taxa de aprendizagem, semente para o gerador de números aleatórios, método de normalização dos vetores, quantidade de *labels* em um nó, entre outros. A configuração desta ferramenta para uso em todos os experimentos é descrita na Tabela 5.3.

Tabela 5.3: Configuração do GHSOM.

Criação do Mapa	Etapa Inicial do Treinamento
Vetores de pesos: inicializados aleatoriamente **Dimensão do mapa:** a quantidade de neurônios é cerca de 4% da quantidade de documentos **Função de atualização da vizinhança:** bubble **Parâmetro do gerador de números aleatórios:** 123	**Expand_Cycles:** 20 **Tau_1:** 0.5 **Tau_2:** 0.05 **Razão de Aprendizagem (alpha):** 0.3 **Raio Inicial da Vizinhança:** metade da altura

Na seção seguinte são apresentadas informações sobre o pré-processamento dos córpus utilizados nos experimentos.

5.1.2 Pré-Processamento dos Conjuntos de Dados Textuais

O pré-processamento dos textos foi realizado utilizando a ferramenta computacional PreTexT (Matsubara et al., 2003) desenvolvida no Labic. O PreTexT, implementado na linguagem Perl, é uma ferramenta que contém uma implementação do

algoritmo de *stemming* do Porter para a língua inglesa. Tal ferramenta pode ser utilizada para:

1. extrair radicais de palavras em português, inglês e espanhol;

2. gerar 1, 2 ou 3-*grams*;

3. ignorar palavras que não são significativas para os textos (*stopwords*), como conjunções e artigos;

4. calcular medidas de atribuição de valores ao atributos: binário, *tf* e *tf-idf*;

5. aplicar a Lei de Zipf e cortes de Luhn;

6. criar a tabela atributo-valor para uso nos algoritmos de *clustering*.

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

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

2. binário e *tf-idf*, como medidas para atribuição de valor aos atributos sem nenhuma normalização;

3. 1-*gram*, como a quantidade de *grams*, ou seja, foram consideradas apenas palavras simples para representar os atributos;

4. Alguns trabalhos sugerem o uso apenas dos k termos mais freqüentes nos documentos para uso no *clustering* (Boley et al., 1999; Huang et al., 2003). Essa sugestão foi adotada em nossos experimentos, variando o corte de Luhn para se o mais próximo possível de 1.000 termos mais freqüentes.

Este pré-processamento foi realizado apenas nos córpus que estavam em formato texto. Para testar os algoritmos escolhemos córpus e *datasets* comuns da literatura, listados na próxima sub-seção.

5.1.3 *Córpus* e Datasets

Os conjuntos de dados textuais (córpus) e de *datasets* pré-processados utilizados nos experimentos são:

1. 20 *Newsgroups*: o córpus 20 *Newsgroups* (20NG) contém 19997 artigos retirados da coleção de *newsgroups* da USENET[6]. Cada artigo é designado em uma ou mais categorias semânticas, sendo 20 categorias, todas com aproximadamente o mesmo tamanho. Foram selecionadas 10 classes de acordo com o experimento realizado em (Bickel & Scheffer, 2004).

2. WebKB: *World Wide Knowledge Base* (WebKB). Este *dataset* é uma coleção de 8.282 páginas web obtidas de quatro domínios acadêmicos, rotuladas primeiramente de acordo com o tópico (*course*, *department*, *faculty*, *project*, *staff*, *student* e *other*) e depois de acordo com o domínio web, como: *student*: *cornell*, *texas*, *washington* entre outros.

3. LNAI/ *WebMiner*: este córpus contém 277 artigos sobre Raciocínio Baseado em Casos (RBC) e 118 sobre Programação Lógica Indutiva (PLI) publicados na série *Lecture Notes on Artificial Intelligence* (LNAI) e 205 documentos sobre Recuperação de Informação (RI) que foram recuperados por uma ferramenta de mineração de dados na web (*WebMiner*) (Brasil & Lopes, 2004);

4. *Classic*: este *dataset* foi obtido combinando-se os *abstracts* dos *datasets* CACM, CISI, CRANFIELD, e MEDLINE, que foram usados para avaliar sistemas de recuperação de informação[7]. Neste *dataset*, cada conjunto de *abstracts* compõe uma classe. Esse *dataset* foi obtido no site do *framework* CLUTO;

5. FBIS: dados do *Foreign Broadcast Information Service* do *dataset* TREC-5. Esse *dataset* foi obtido no site do *framework* CLUTO;

6. *Reuters*: o córpus *Reuters*-21578 contém 21.578 artigos retirados da agência de notícias *Reuters*[8]. Cada artigo é designado tipicamente em uma ou mais categorias semânticas como *Earn*, *Trade*, *Corn*, entre outras. Esse *dataset* foi obtido

[6]http://kdd.ics.uci.edu/
[7]ftp://ftp.cs.cornell.edu/pub/smart
[8]http://www.research.att.com/~lewis.

no site do *framework* CLUTO, separado em RE0 e RE1, sendo que utilizamos o RE0. Assim, deve-se considerar as referências ao *Reuters* como sendo apenas a partição RE0;

7. WAP: projeto WebACE (WAP). Cado documento corresponde a uma página web listada na hierarquia de assuntos do site *Yahoo!*. Esse *dataset* foi obtido no site do *framework* CLUTO;

Na Tabela 5.4 são mostradas mais informações sobre a dimensão dos *datasets*.

Tabela 5.4: *Datasets* utilizados nos experimentos.

Dataset	Documentos	Classes	Termos
20NG	2.000	10	1.015
WebKB	1.639	5	1.007
LNAI/ *WebMiner*	564	3	1.033
Classic	7.089	4	12.009
FBIS	2.463	17	2.000
Reuters	1.504	13	2.886
WAP	1.560	20	8.440

Os *datasets* do CLUTO[9] foram obtidos em um formato de matriz esparsa, em que apenas os elementos diferentes de zero e sua posição correspondente no vetor, são armazenados. Além disso, é necessário um cabeçalho informando a quantidade de exemplos, a quantidade de colunas (total de atributos) e o número de elementos contidos na matriz esparsa (elementos diferentes de zero). O formato adotado pelo *framework* CLUTO difere do formato dos algoritmos Gmeans e do Co-K-means. Um outro problema é o fato dos algoritmos SOM e GHSOM não trabalharem com essa representação de matriz. Neste caso, foi necessário criar um conversor de matriz esparsa para matriz densa, em que todos os elementos são armazenados (inclusive os de valor 0). Essa representação de matriz densa possui tamanho em disco muito maior, causando, ainda, um custo maior de processamento. Assim, foi necessário criar conversores para os demais algoritmos.

[9]http://www.cs.umn.edu/karypis/cluto/files/datasets.tar.gz

Para os córpus, que foram obtidos em forma de texto, foi necessário realizar a etapa de pré-processamento, descrita na sub-seção anterior.

A seguir são mostrados os experimentos realizados com os algoritmos e *datasets* já comentados neste capítulo.

5.2 Experimentos Realizados

As tabelas atributo-valor geradas pelo PreTexT consistem dos exemplos e atributos selecionados pelos parâmetros definidos anteriormente. Como o objetivo deste trabalho é a análise dos resultados obtidos por algoritmos de *clustering*, aplicados a documentos textuais, a classificação dos documentos foi omitida na execução dos algoritmos nos experimentos realizados e utilizadas, posteriormente, na avaliação dos clusters obtidos.

A metodologia adotada consiste, basicamente, em submeter os exemplos a um algoritmo de *clustering* para agrupá-los em clusters. Após a descoberta dos clusters é feito o cálculo das medidas de qualidade Pureza, que mede a quantidade de exemplos da classe majoritária no cluster em relação ao tamanho do próprio cluster, e Entropia, que calcula como as várias classes estão distribuídas dentro de cada cluster. Em geral, quanto maior for a Pureza e quanto menor for a Entropia, melhor a qualidade do cluster. Os valores mostrados nas tabelas/gráficos são as médias dessas medidas em relação a todos os clusters encontrados para cada *dataset*, sendo que o valor 1,000 corresponderia a 100% e 0,000 a 0%. Como o *framework* CLUTO fornece diversas funções de critério, foram executados experimentos com cada uma delas para se avaliar seu comportamento em cada *dataset*. Já no caso de algoritmos do tipo SOM, o mapa gerado é avaliado subjetivamente. São analisados, também, o consumo de memória e de tempo computacional.

É importante notar, ainda, que cada *dataset* possui uma quantidade distinta de classes pré-definidas por quem o criou. Assim, foi solicitado ao algoritmo de *clustering* que encontre esta exata quantidade de clusters visto que, à medida que se aumenta esse valor, a qualidade dos clusters pode melhorar, devido ao fato de que mais grupos serão formados, ou pode piorar, visto que podem existir clusters com pequenas quantidades de documentos de classes distintas, gerando um alto valor de entropia.

Nos experimentos dos córpus 20 *Newsgroups*, WebKB e LNAI/*WebMiner* foram executados os algoritmos particionadores (RB e RBR) e aglomerativos (Agglo e Bagglo) do *framework* CLUTO e o *Spherical K-Means* da ferramenta Gmeans. Este último foi executado 100 vezes, visto que o *K-Means* apresenta resultados distintos a cada execução, diferentemente dos algoritmos particionadores e aglomerativos. Ainda, nos experimentos com o córpus LNAI/*WebMiner* foram avaliadas combinações envolvendo resumo, texto das referências, *bag of references*, com pesos binário e *tf-idf*. Ainda, foi realizado um experimento para avaliar o comportamento dessas combinações em um algoritmo multi-visão. Em cada um destes experimentos são apresentados:

- o córpus;
- os algoritmos utilizados;
- uma tabela e dois gráficos contendo os resultados de pureza e entropia médias dos clusters encontrados usando o CLUTO;
- uma tabela com o tempo e a memória consumidos pelo CLUTO no experimento;
- duas matrizes de confusão: uma com o melhor resultado e a outra com o pior resultado. É importante informar que nestas tabelas a primeira linha representa as classes do córpus e a primeira coluna representa os clusters encontrados pelo algoritmo. Cada uma das células informa a quantidade de exemplos da classe i no cluster j. Ainda, as células vazias significam que não contém nenhum documento daquela classe naquele cluster;
- um gráfico das 100 execuções do *Spherical K-means*, com os resultados em ordem decrescente de pureza;
- uma tabela com as melhores entropia e pureza usando CLUTO e entropia e pureza médias usando o *Spherical K-Means;*
- conclusões sobre os experimentos relatados;
- um mapa topográfico, gerado pelo gCLUTO;
- um mapa gerado com as ferramentas do Som_Pak;
- um mapa gerado pelo GHSOM;

- conclusões sobre os mapas.

Nos experimentos com os *datasets* foram utilizados apenas o *framework* CLUTO. Como não se tem disponível os córpus referentes aos *datasets*, não foram realizados experimentos cujos resultados pudessem ser avaliados subjetivamente (SOM e GHSOM). Para estes experimentos são apresentados:

- o *dataset*;
- os algoritmos utilizados;
- uma tabela contendo os resultados de pureza médias dos clusters;
- uma tabela com o tempo e a memória consumidos pelo CLUTO no experimento;
- conclusões sobre os experimentos;

5.2.1 *Experimento 1: Córpus 20* Newsgroups

Seguindo o trabalho de Bickel e Scheffer (Bickel & Scheffer, 2004), utilizamos 10 das 20 classes para os experimentos: *comp.graphics* (*Grafx*), *comp.os.ms-windows.misc* (*WinMisc*), *comp.sys. ibm.pc.hardware* (*IBM*), *misc.forsale* (*MiscSale*), *rec.autos* (*Autos*), *rec.motorcycles* (*Motorcyles*), *rec.sport.hockey* (*Hockey*), *sci.space, sci.med* e *soc.religion.christian* (*Religion*). Foram selecionados aleatoriamente 200 documentos de cada uma das classes, totalizando 2.000 documentos. A ferramenta PreTexT foi utilizada para gerar a tabela atributo/valor com a configuração descrita na seção 5.1.2.

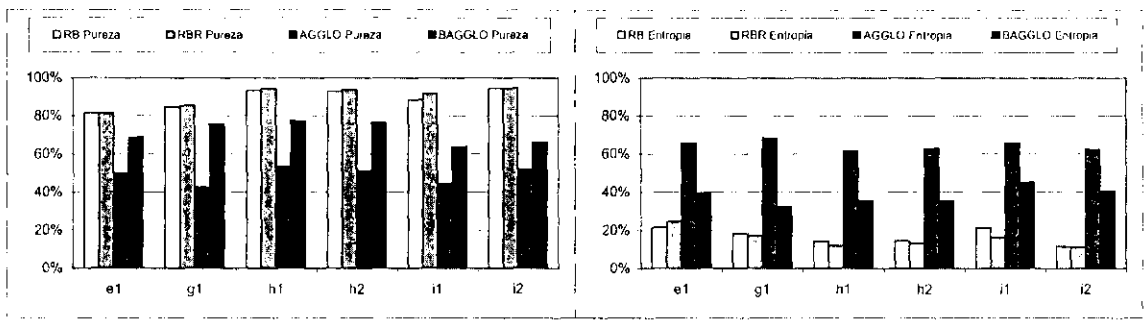
Percebe-se pela Tabela 5.5 que algoritmos hierárquicos possuem dificuldade em encontrar os clusters deste córpus. A tabela é incompleta pois as funções de otimização de *clink* até *wupgma* não são compatíveis com o algoritmo particionador no *framework* CLUTO. Os valores em negrito indicam o melhor resultado (maior pureza com menor entropia) dentre os algoritmos utilizando a mesma função de critério. Isto é válido para todas as tabelas com resultados de *clustering*, em todos os experimentos.

O algoritmo RBR, que tenta uma otimização global, superou o RB neste córpus porém, sem diferença significativa, a não ser pela função *i1*. O algoritmo Bagglo obteve

Tabela 5.5: Pureza e Entropia do córpus 20 *Newsgroups* usando CLUTO.

	RB		RBR		AGGLO		BAGGLO	
Função	**Entropia**	**Pureza**	**Entropia**	**Pureza**	**Entropia**	**Pureza**	**Entropia**	**Pureza**
e1	0,216	**0,815**	0,245	0,814	0,657	0,500	0,396	0,688
g1	0,182	0,846	0,170	**0,853**	0,683	0,428	0,324	0,756
h1	0,141	0,933	0,121	**0,944**	0,615	0,533	0,354	0,772
h2	0,146	0,929	0,134	**0,937**	0,628	0,512	0,353	0,763
i1	0,212	0,883	0,161	**0,917**	0,657	0,446	0,452	0,638
i2	0,116	0,944	0,114	**0,946**	0,622	0,518	0,405	0,662
clink	-	-	-	-	0,708	0,377	0,381	**0,633**
slink	-	-	-	-	0,993	**0,106**	0,994	0,106
upgma	-	-	-	-	0,798	0,292	0,322	**0,779**
wclink	-	-	-	-	0,708	0,377	0,381	**0,633**
wslink	-	-	-	-	0,993	**0,106**	0,994	0,106
wupgma	-	-	-	-	0,613	0,544	0,264	**0,863**

Figura 5.1: Gráfico dos resultados da execução de algoritmos de *clustering* no córpus 20 *Newsgroups* (funções *e1* até *i2*).



resultados melhores do que o Agglo devido ao fato de ser guiado por um resultado do algoritmo RB. A seguir, são mostrados os gráficos equivalentes à Tabela 5.5.

Fica claro, pela Figura 5.1, a superioridade das técnicas particionadoras neste córpus. Os algoritmos RB e RBR possuem resultados semelhantes enquanto que Agglo e Bagglo, além de resultados bem distintos são, também, ruins. Já na Figura 5.2, pode-se perceber que Agglo e Bagglo conseguem resultados semelhantes apenas usando as funções *slink* e *wslink*, que geraram os agrupamentos de pior qualidade deste experimento.

O tempo e a memória consumidos pelo CLUTO neste *dataset* são mostrados na Tabela 5.6. Os algoritmos particionadores consomem muito menos tempo e memória do que os aglomerativos. É interessante notar o tempo gasto ao se usar as funções *h1* e *h2* nos algoritmos aglomerativos. Mesmo com a quantidade de tempo de processamento dispendida, a qualidade dos clusters não foi boa.

Nas Tabelas 5.7 e 5.8 tem-se as matrizes de confusão para duas configurações:

Figura 5.2: Gráfico dos resultados da execução de algoritmos de *clustering* no córpus 20 *Newsgroups* (funções *clink* até *wupgma*).

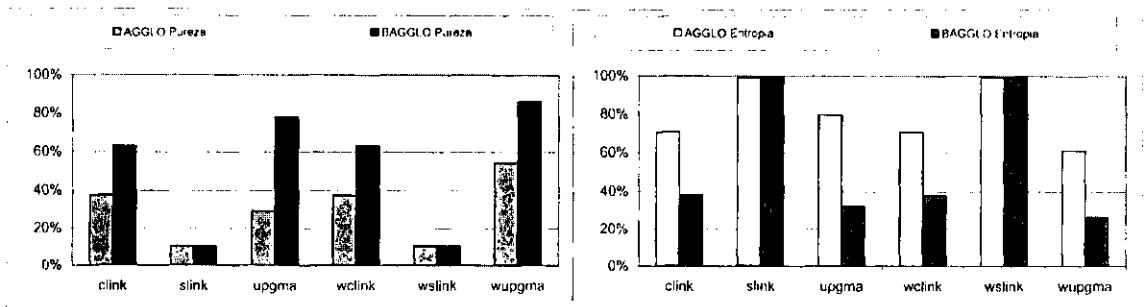


Tabela 5.6: Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus 20 *Newsgroups*.

	RB		RBR		AGGLO		BAGGLO	
Função	**T**	**M**	**T**	**M**	**T**	**M**	**T**	**M**
e1	7,01	40,00	7,55	40,00	37,99	62,00	35,89	62,00
g1	10,53	45,00	11,17	45,00	38,47	62,00	35,94	62,00
h1	7,65	45,00	7,97	45,00	299,91	62,00	274,41	62,00
h2	6,59	42,00	7,15	42,00	291,57	62,00	271,48	62,00
i1	7,74	45,00	8,77	45,00	36,38	62,00	35,07	62,00
i2	6,41	45,00	7,15	45,00	38,15	62,00	35,79	62,00
clink	-	-	-	-	29,37	62,00	34,53	62,00
slink	-	-	-	-	29,62	62,00	34,73	62,00
upgma	-	-	-	-	28,49	62,00	34,85	62,00
wclink	-	-	-	-	29,56	62,00	34,55	62,00
wslink	-	-	-	-	29,48	62,00	34,54	62,00
wupgma	-	-	-	-	29,87	62,00	35,23	62,00

Agglo com a função *slink*, cujos clusters encontrados possuem maior valor de entropia e RBR com a função *i2*, com o menor valor de entropia do experimento. É importante informar que os clusters são ordenados pelo valor de similaridade interna. Assim, o cluster com documentos mais similares é o cluster 0, e o cluster com menor similaridade entre seus documentos é o cluster n, neste caso, o cluster 9.

O algoritmo hierárquico agrupou praticamente todos os documentos em um mesmo cluster, gerando um resultado de qualidade extremamente baixa. Esse problema, que se repete nos demais experimentos, é causado aparentemente pelo fato de a função *slink* ter dificuldade de trabalhar com vetores muito esparsos, apesar de o algoritmo Agglo

Tabela 5.7: Matriz de confusão (Agglo - slink)

cluster	grafx	winm	ibm	misc	auto	moto	hock	scis	scim	reli
0		3								
1	1									
2			1							
3								1		
4						2				
5	1									
6		1								
7								1		
8								2		
9	198	196	199	200	200	198	200	196	200	200

não ter obtido bons resultados com nenhuma das funções de critério neste experimento. Isso não ocorre com o algoritmo particionador (ver Tabela 5.8).

Tabela 5.8: Matriz de confusão (RBR - $i2$)

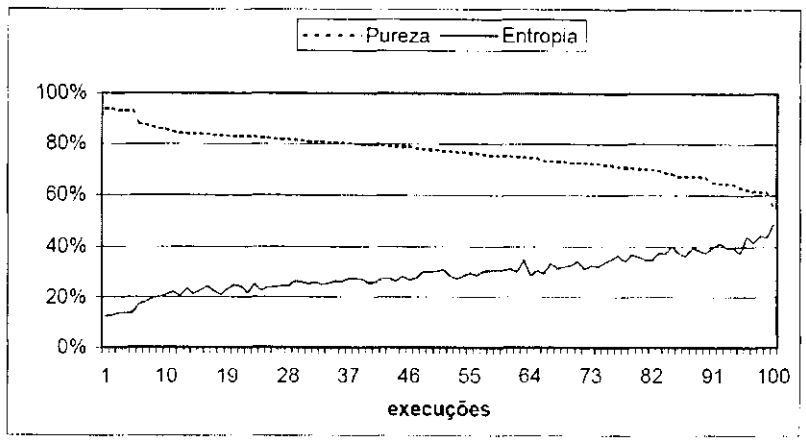
cluster	graf	winm	ibm	misc	auto	moto	hock	scis	scim	reli
0		1	1							200
1	1		2	185	5	3				
2		1			1	3		2	195	
3	6	178	5	1		1		1		
4			1	2	1		197	1	1	
5	180	5	1			1				
6				4	187	2		1		
7	13	13	190	8	2	2		1	3	
8		1			1		1	193	1	
9		1			3	188	2	1		

Nota-se que a classe *Religion* não foi particionada, sendo completamente agrupada no cluster 0. As classes *Hockey*, *SciMed* e *SciSpace* também tiveram pouca dispersão de seus documentos. Já as classes *Grafx* e *WinMisc* tiveram cerca de 12% de seus documentos alocados erroneamente. Nesta solução um dos clusters agrupou documentos de 8 das 10 classes, ocasionando um entropia elevada. Porém, isto não significa que este cluster seja ruim, visto que o cluster com menor similaridade interna é o cluster 9 e que a entropia é calculada de acordo com a classificação prévia dos documentos.

Na Figura 5.3 tem-se a execução do algoritmo *Spherical K-Means*. Este algoritmo

foi executado 100 vezes neste córpus com a opção *random seed* para inicializar os centróides. Os resultados foram ordenados decrescentemente de acordo com o valor de pureza.

Figura 5.3: Gráfico de 100 execuções do *Spherical K-Means* no córpus 20 *Newsgroups*.



Calculando-se a média da pureza, entropia e seus respectivos valores de desvio padrão (D.P.), tem-se um resultado de boa qualidade, mas inferior aos melhores obtidos com o CLUTO (Tabela 5.9). O tempo computacional e o consumo de memória, correspondentes a uma execução, ficaram entre os particionadores e os hierárquicos.

Tabela 5.9: Melhores Entropia e Pureza do córpus 20 *Newsgroups* usando CLUTO e Entropia e Pureza médias *Spherical K-Means*.

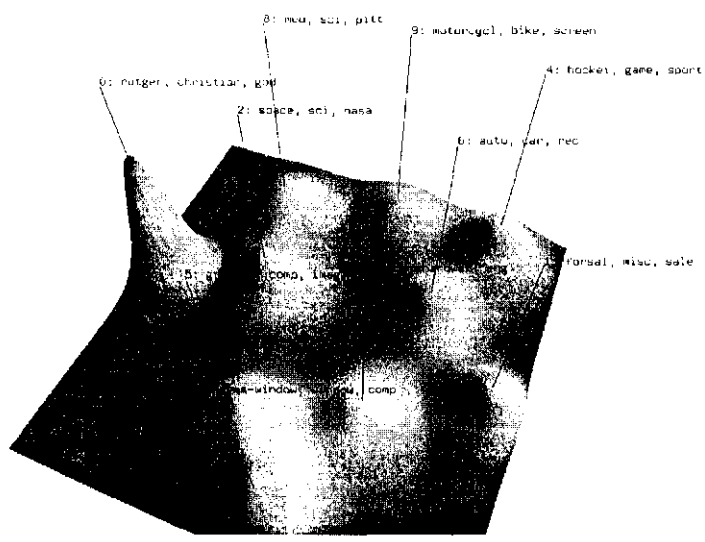
Algoritmo	**Entropia**	**D.P.**	**Pureza**	**D.P.**	**T**	**M**
RBR (*i2*)	0,114	-	0,946	-	7,15	45,00
BAGGLO (*wupgma*)	0,264	-	0,863	-	35,23	62,00
Spherical K-Means	0,291	0,073	0,767	0,078	6,87	7,92

Para este *dataset*, o algoritmo RBR com a função *i*2 obteve os melhores resultados, seguido pelo RB com a mesma função. Dada a pequena diferença nos valores de entropia e pureza e o fato de o algoritmo RBR ter um custo computacional maior do que o RB, visto que tenta uma otimização global após a execução do próprio RB, o algoritmo mais indicado para o córpus 20NG é o RB com a função *i2*.

Mapas

Na Figura 5.4 tem-se o mapa em 3D gerado pela ferramenta gCLUTO a partir da melhor solução encontrada para este córpus: RBR com a função *i2*. Este mapa foi criado em **13,62 segundos**, somado o tempo do RBR. Analisando-se a topologia do mapa, percebe-se que o algoritmo conseguiu uma boa divisão dos exemplos. Os clusters 0 e 8 possuem 99% de pureza. A cor vermelha no topo cluster 0 e sua altura elevada devem-se ao menor valor de desvio padrão interno (entre os documentos do cluster) e ao maior valor de similaridade interna, respectivamente. Já o cluster 8, com baixo valor de similaridade interna, tem pouca altura e topo verde azulado. Ainda, vale notar o posicionamento dos clusters no mapa. Os clusters 3, 5 e 7, que são da área de computação, estão próximos entre si. Os clusters 2 (*SciSpace*) e 8 (*SciMed*) também estão posicionados próximos. Já o cluster 1 - *Misc* - ficou em uma borda do mapa, do lado oposto do cluster 0 - *Religion*. Assim, o mapa gerado para este córpus apresenta uma boa representação dos documentos.

Figura 5.4: Mapa 3D do córpus 20 *Newsgroups*.

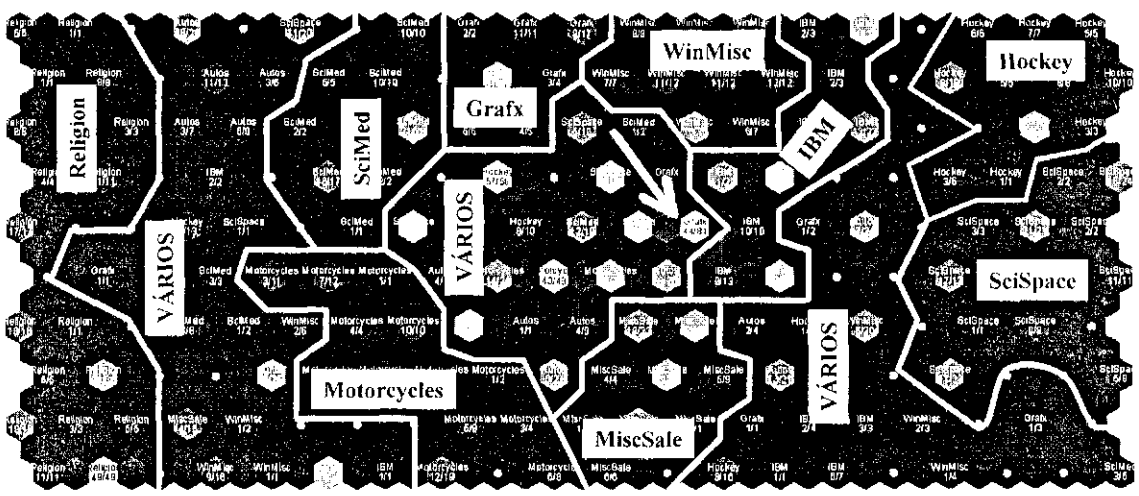


Já o algoritmo SOM não foi capaz de agrupar corretamente os documentos. O mapa da Figura 5.5 foi criado pela ferramenta *umat* com a dimensão de 20x10. Esta dimensão foi adotada de modo que cada neurônio representasse, em média, 10 documentos. Foram necessários **15min e 23s** e aproximadamente **9,5 Mb** de RAM para se gerar este mapa. A cor do neurônio indica a quantidade de documentos que ele representa, tal que quanto mais clara, maior a quantidade. O texto no neurônio é a

classe majoritária dos documentos que ele agrupa e os valores logo embaixo são o número de documentos da classe majoritária e o número de documentos no neurônio. Já os círculos brancos são neurônios sem documentos. Vale ressaltar que todos os mapas criados pelo SOM foram editados e que os limites, rótulos e setas foram adicionados manualmente.

É possível notar na Figura 5.5 que alguns grupos puderam ser delimitados mas existem duas áreas com grande número de documentos de diversas classes - rótulo VÁRIOS. Aparentemente, a única classe que manteve seus documentos próximos foi a *Religion*. As demais tiveram seus documentos espalhados pelo mapa. Na região central do mapa existe uma grande concentração de documentos. Um problema nesta concentração é o fato de um neurônio agrupar documentos de diversas classes.

Figura 5.5: Mapa do córpus 20 *Newsgroups* gerado usando SOM.

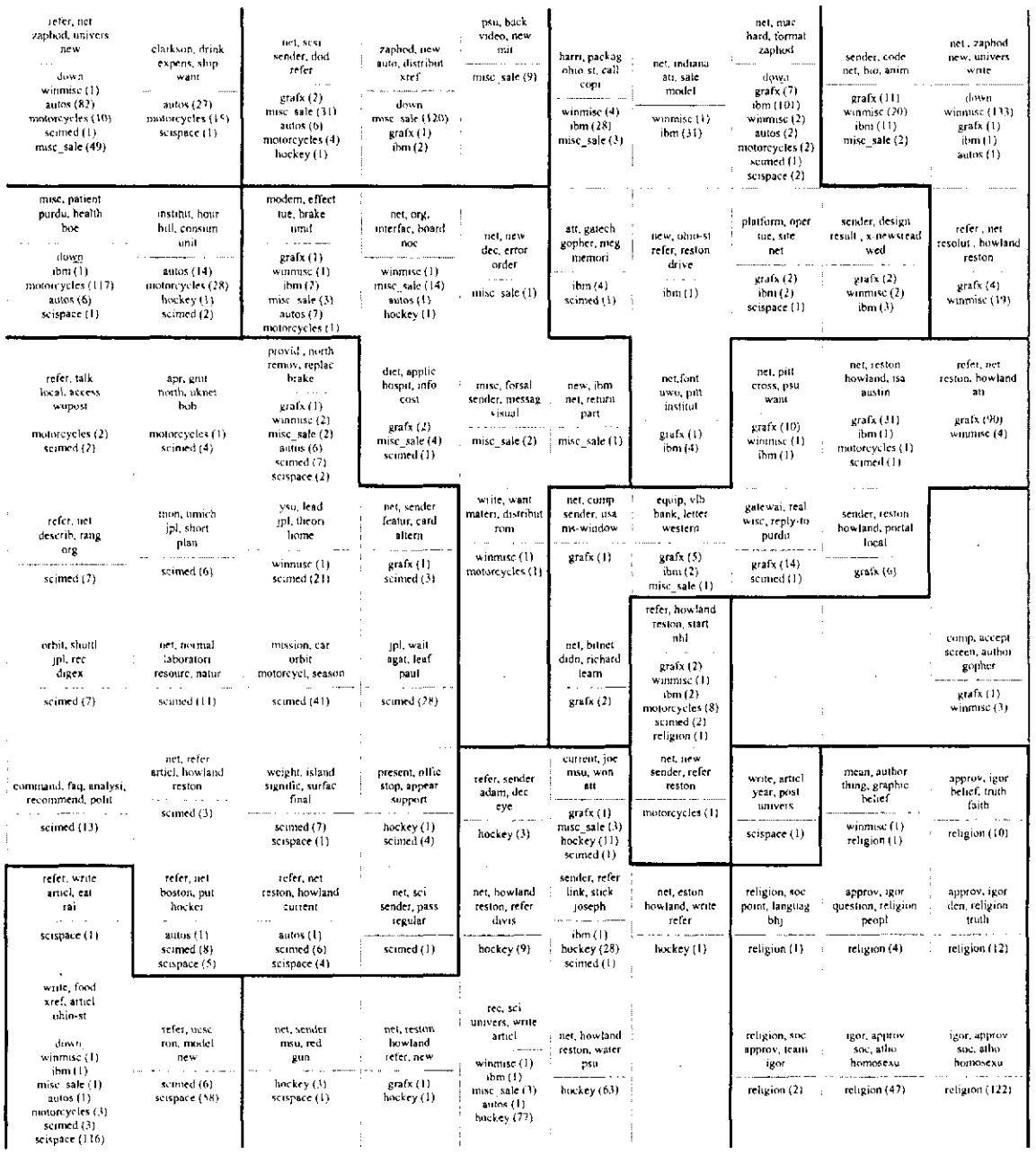


Como exemplo tem se o neurônio apontado pela seta branca - na região central do mapa - que possui 81 documentos, sendo 44 da classe *Grafx* e ao redor tem-se documentos das classes *IBM*, *Autos* e *SciMed*. Foram executados experimentos com mapas de diferentes tamanhos, taxa de aprendizagem e raio de vizinhança, mas o algoritmo SOM não conseguiu preservar a distribuição dos documentos.

A seguir, no mapa gerado pelo GHSOM, cada célula contém os 5 termos mais descritivos do neurônio e, logo embaixo, a quantidade de documentos que cada classe possui naquele neurônio. O termo *doum* indica que foi criado um novo mapa para distribuir os documentos representados por aquele neurônio. A contagem que aparece

nesta célula com o termo *down*, equivale aos documentos dos mapas dos outros níveis deste neurônio. Células vazias - que possuem apenas um '.' são neurônios sem documentos. Os mapas criados pelo GHSOM também foram editados. Nos mapas originais tem-se *links* para todos os documentos ao invés da contagem de documentos por classe. Tal contagem foi inserida manualmente.

Figura 5.6: mapa do córpus 20 *Newsgroups* gerado usando GHSOM.



O algoritmo GHSOM, foi iniciado com um mapa de tamanho 10x8, visto que uma de suas características é a expansão do mapa. O tempo necessário foi de **30,9**

segundos e a ferramenta consumiu **8,89 Mb**. O resultado gerado foi similar ao do SOM. No mapa da Figura 5.6 pode-se perceber o mesmo problema de documentos espalhados pelo mapa e neurônios com documentos de diversas classes. Novamente, a classe que foi melhor representada foi a *Religion*.

O mapa obtido com a ferramenta gCLUTO, que utiliza o resultado de um algoritmo de *clustering* para plotar o mapa, apresenta uma boa representação dos dados. Os clusters de áreas semelhantes foram posicionados próximos e houve uma boa distância entre clusters distintos. O mapa gerado usando o algoritmo SOM é desorganizado. Os grupos não ficaram bem definidos, existem documentos espalhados pelo mapa, além de neurônios agrupando vários documentos de diversas classes. O mesmo ocorre com o mapa gerado com o GHSOM. Apesar de se ter novas camadas quando os neurônios agrupam muitos documentos, esta nova camada não é capaz de corrigir os erros da camada anterior. Ainda, levando-se em consideração o custo computacional, o mapa gerado com a ferramenta gCLUTO em menos de 14 segundos superou consideravelmente as outras duas ferramentas.

5.2.2 *Experimento 2: Córpus WebKB*

Este córpus possui 6 classes principais: *course*, *department*, *faculty*, *project*, *staff* e *student*. Para este experimento foi selecionada apenas a maior pasta: *student*, contendo 5 sub-pastas *cornell*, *misc*, *texas*, *washington* e *wisconsin*, num total de 1.639 exemplos. Foram executados os algoritmos particionadores e aglomerativos hierárquicos do *framework* CLUTO em várias configurações e do algoritmo *Spherical K-Means*.

A execução dos algoritmos de *clustering* neste córpus resultaram em clusters de boa qualidade. É importante destacar o desempenho do algoritmo Agglo. No experimento anterior este algoritmo não foi capaz de lidar com os dados de maneira eficaz. Entretanto, obteve o melhor resultado deste experimento, com a função *i1*, superando os demais algoritmos também usando outras funções (Tabela 5.10).

Em geral, todos os algoritmos se saíram bem neste córpus. A menor pureza foi de 66%, sendo que 8 resultados foram de 66% a 70% de pureza e 4 acima de 90%. Nas figuras 5.7 e 5.8 tem-se os gráficos da Tabela 5.10. Comparando-se os gráficos 5.7 e 5.8 pode-se perceber que as soluções de menor qualidade foram encontradas com as

Tabela 5.10: Pureza e Entropia do córpus WebKB usando CLUTO.

	RB		RBR		AGGLO		BAGGLO	
Função	**Entropia**	**Pureza**	**Entropia**	**Pureza**	**Entropia**	**Pureza**	**Entropia**	**Pureza**
e1	0,370	0,688	0,373	0,68	0,372	0,732	0,295	**0,742**
g1	0,367	0,752	0,368	0,752	0,236	**0,890**	0,367	0,807
h1	0,149	**0,904**	0,151	0,904	0,189	0,893	0,163	0,898
h2	0,218	**0,822**	0,256	0,770	0,351	0,729	0,300	0,735
i1	0,250	0,885	0,263	0,881	0,159	**0,949**	0,157	0,945
i2	0,223	0,805	0,225	0,805	0,293	**0,817**	0,324	0,747
clink	-	-	-	-	0,361	0,732	0,277	**0,773**
slink	-	-	-	-	0,689	0,660	0,689	**0,661**
upgma	-	-	-	-	0,683	0,660	0,637	**0,66**
wclink	-	-	-	-	0,361	0,732	0,277	**0,773**
wslink	-	-	-	-	0,689	0,660	0,689	**0,661**
wupgma	-	-	-	-	0,251	0,885	0,209	**0,896**

funções de critério adotadas apenas em algoritmos hierárquicos.

Figura 5.7: Gráfico dos resultados da execução de algoritmos de *clustering* no córpus WebKB (funções *e1* até *i2*).

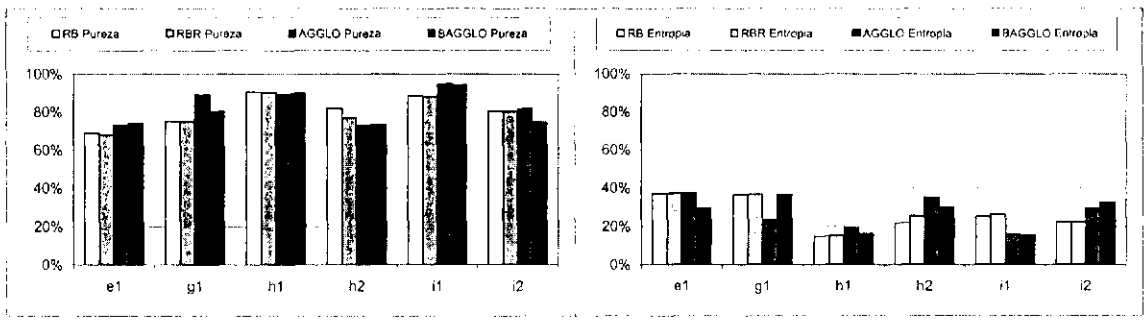
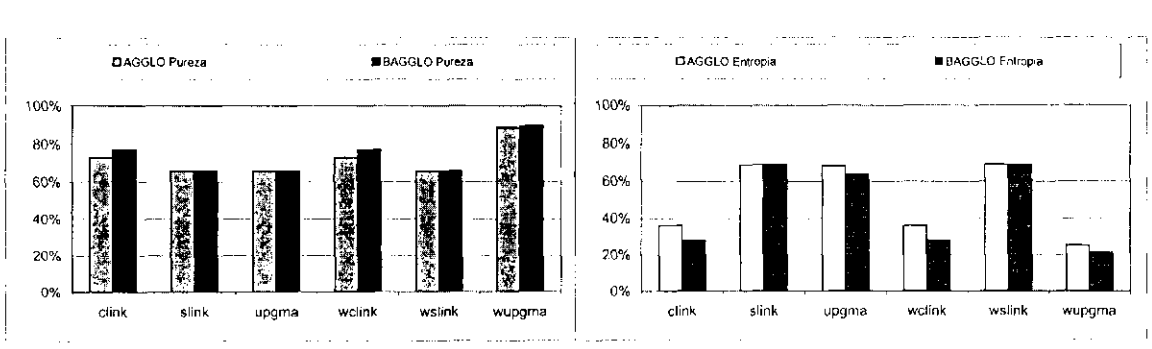


Figura 5.8: Gráfico dos resultados da execução de algoritmos de *clustering* no córpus WebKB (funções *clink* até *wupgma*).



Apesar do bom desempenho dos algoritmos hierárquicos em termos de pureza e entropia dos clusters, seu custo continua sendo muito elevado se comparado aos algoritmos particionadores (Tabela 5.11). Curiosamente, a diferença do consumo de memória

neste córpus, entre os algoritmos particionadores e os aglomerativos, foi pequena, se comparada aos valores no experimento anterior.

Tabela 5.11: Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus WebKB.

	RB		RBR		AGGLO		BAGGLO	
Função	**T**	**M**	**T**	**M**	**T**	**M**	**T**	**M**
e1	3,38	32,00	3,68	32,00	22,89	48,00	28,62	48,00
g1	3,51	36,00	3,80	36,00	24,55	48,00	29,39	48,00
h1	3,06	33,00	3,18	33,00	158,39	48,00	161,09	48,00
h2	3,51	33,00	3,66	33,00	152,04	48,00	153,71	48,00
i1	4,47	37,00	4,68	37,00	25,55	48,00	29,92	48,00
i2	2,93	33,00	3,08	33,00	22,93	48,00	27,17	48,00
clink	-	-	-	-	26,31	48,00	26,09	48,00
slink	-	-	-	-	20,86	48,00	27,39	48,00
upgma	-	-	-	-	22,51	48,00	24,93	48,00
wclink	-	-	-	-	22,02	48,00	25,52	48,00
wslink	-	-	-	-	21,60	48,00	26,08	48,00
wupgma	-	-	-	-	23,44	48,00	24,87	48,00

As matrizes de confusão para o pior resultado (Agglo / *slink*) na Tabela 5.12 e o melhor resultado (Agglo / *i1*) na Tabela 5.13. Mais uma vez a função *slink* mostrou-se desaconselhável.

Tabela 5.12: Matriz de confusão (Agglo - *slink*).

cluster	**cornell**	**misc**	**texas**	**washington**	**wisconsin**
0		1			
1		1			
2		1			
3		1			
4	127	1078	148	126	156

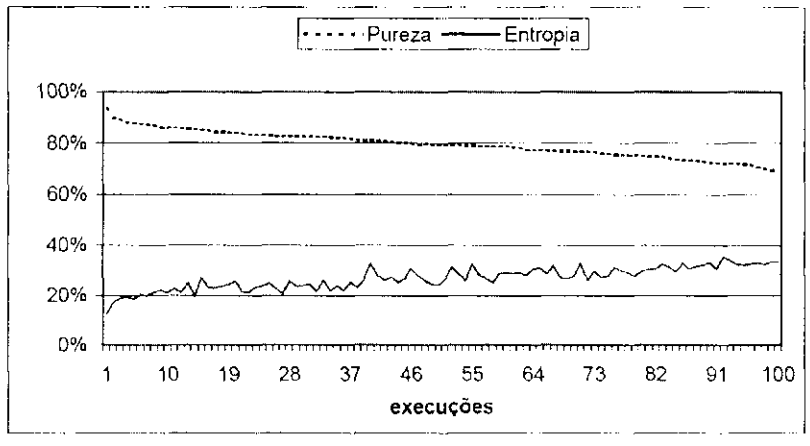
Fica claro na Tabela 5.13 que o resultado não foi melhor graças ao cluster 4, que agrupou exemplos de todas as classes. Como a classe *Misc* engloba documentos que, teoricamente, não se encaixam em nenhuma das outras 4 classes, era de se esperar que o cluster característico desta classe fosse genérico o suficiente para agrupar documentos pertencentes a outras classes.

Tabela 5.13: Matriz de confusão (Agglo - *i1*).

cluster	cornell	misc	texas	washington	wisconsin
0					136
1				111	
2	101	1			
3			126		
4	26	1081	22	15	20

A execução do *Spherical K-Means* neste córpus foi realizada com a opção *random seed*. Os resultados foram ordenados em ordem decrescente de pureza e podem ser vistos na Figura 5.9.

Figura 5.9: Gráfico de 100 execuções do *Spherical K-Means* no córpus WebKB.



De acordo com a média e o desvio padrão das medidas de pureza e entropia, percebe-se um resultado de boa qualidade, apesar de bem inferior aos melhores obtidos com o CLUTO (Tabela 5.14). O tempo computacional foi próximo ao dos algoritmos particionadores e o consumo de memória foi bem inferior.

Como o *Spherical K-Means* teve uma pureza média muito baixa, será desconsiderado deste experimento, apesar de ser rápido e eficiente no consumo de memória. Os resultados com o algoritmo Agglo usando a função *i1* foi próximo ao ideal desejado para este córpus. O algoritmo RB encontrou clusters de boa qualidade mas ligeiramente inferiores aos do Agglo. O ponto principal neste caso é, novamente, o custo

computacional. O algoritmo hierárquico levou 25,55s contra apenas 3,06s do RB. Esta diferença parece pequena, pois o tempo está em segundos. Assim, se o usuário optar pelo benefício, o Agglo é a opção correta para este córpus. Caso o usuário escolher pelo custo, o algoritmo RB é, sem dúvida, o ideal.

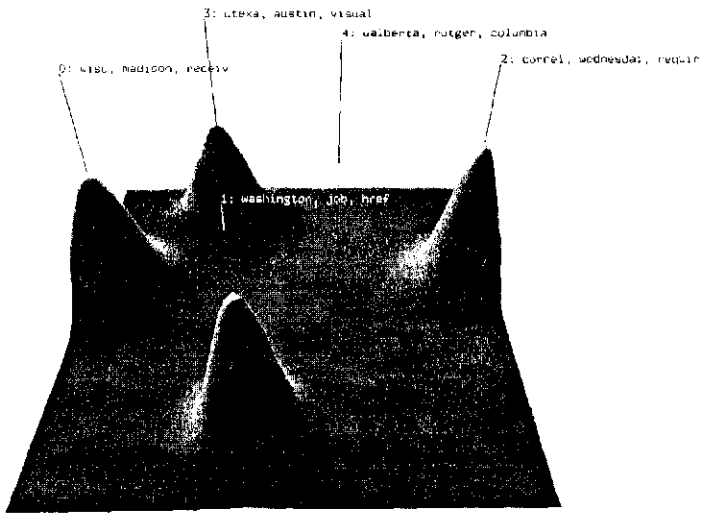
Tabela 5.14: Melhores Entropia e Pureza do córpus WebKB usando CLUTO e Entropia e Pureza médias *Spherical K-Means*.

Algoritmo	Entropia	D.P.	Pureza	D.P.	T	M
RB (*h1*)	0,149	-	0,904	-	3,06	33,00
AGGLO (*i1*)	0,159	-	0,949	-	25,55	48,00
Spherical K-Means	0,268	0,045	0,795	0,050	5,29	6,42

Mapas

Na Figura 5.10 tem-se o mapa em 3D do córpus WebKb para o melhor resultado obtido pelo CLUTO. O mapa foi gerado em **7,63 segundos**, somado o tempo do clustering RBR/*h1*. Analisando-se as montanhas neste mapa, percebe-se que o algoritmo conseguiu uma divisão muito boa dos exemplos. Uma característica que deve ser notada é o conjunto de termos mais descritivos nos clusters. Como pode ser visto na Figura 5.10, o termo do cluster 0 é *wisc*, o do cluster 1 é *washington*, o do cluster 2 é *cornel* e o do cluster 3 é *utexa*. Essa característica explica a boa divisão dos documentos.

Figura 5.10: mapa 3D do córpus WebKB.

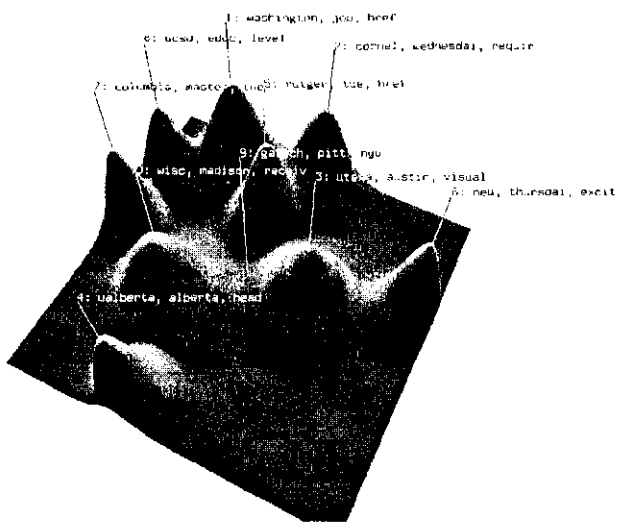


O cluster 3, de topo vermelho, é 100% puro e possui o menor desvio interno desta

solução. Porém, sua similaridade interna é baixa. A maior similaridade interna é do cluster 2, que possui, ainda, um valor pequeno de desvio interno e pureza próximo a 100%. O cluster 0, também 100% puro, possui o segundo menor valor de desvio, mas sua similaridade interna é de valor médio. A maior similaridade é do cluster 1. Nota-se, na Figura 5.10, que ele ficou bem afastado dos demais clusters. Finalizando, o cluster 4, cuja classe majoritária é a Misc, foi o pior resultado desta solução, com o menor valor de similaridade interna, maior valor de desvio interno e menor pureza (apesar de ser um valor muito bom - cerca de 92%). Essa situação já era prevista, dado o nome da classe. É interessante notar no mapa que este cluster não tem altura.

O fato de haver uma divisão eficaz dos dados e de os clusters estarem distantes uns dos outros permite que o processo seja refeito com um número maior de clusters, de modo a se encontrar sub-áreas nos agrupamentos (ver Figura 5.11). Neste novo mapa a pureza manteve-se em 94,9%. A única classe que sofreu particionamento foi a *Misc*, criando os clusters de 4 a 9.

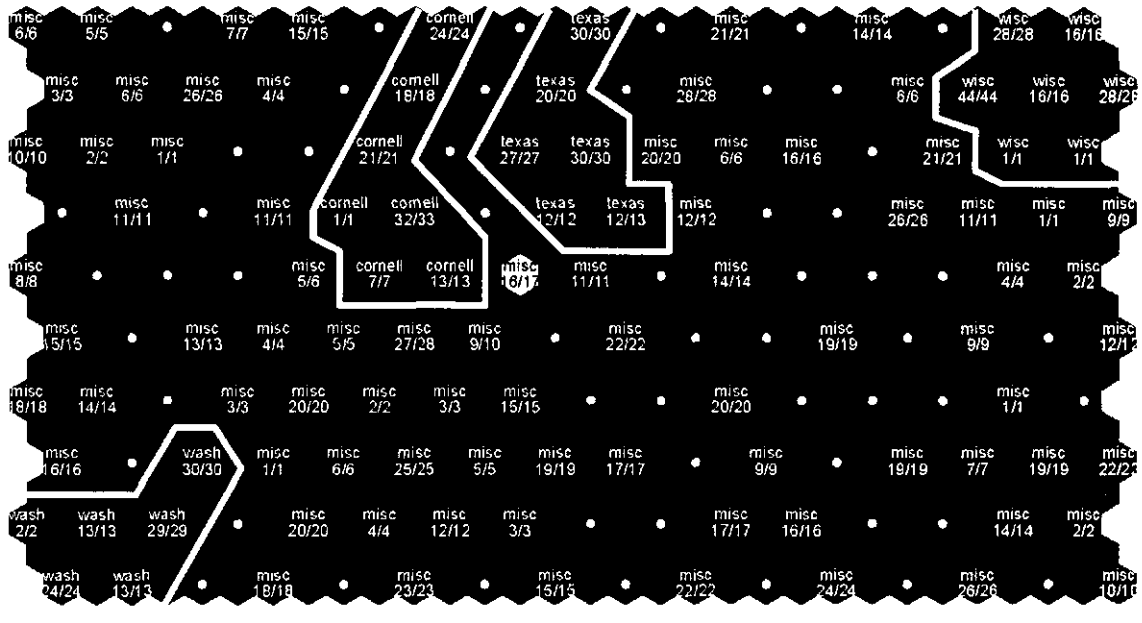
Figura 5.11: Mapa 3D do córpus WebKB - 10 clusters.



Utilizamos a ferramenta Som_Pak para gerar um mapa da distribuição dos documentos. A dimensão do mapa seguiu o mesmo critério do experimento anterior, ou seja, a quantidade de neurônios é cerca de 10% da quantidade de documentos. Este mapa possui 160 neurônios (16x10) para distribuir 1.639 documentos. A ferramenta consumiu **11min 44s** e cerca de **8 Mb** para gerar o mapa final. O resultado pode ser visto na Figura 5.12.

Diferentemente do que ocorreu no experimento anterior, o algoritmo SOM gerou um resultado interessante. Os documentos das classes *cornell*, *texas*, *washington* e *wisconsin* ficaram bem agrupados e com fácil delimitação de suas fronteiras. Fazendo-se a contagem dos documentos em cada um desses grupos percebe-se que o SOM dispersou menos os documentos das classes *cornell* (116 documentos) e *texas* (131 documentos) do que o algoritmo hierárquico (ver Tabela 5.13). A distância entre eles também pode ser considerada satisfatória. O problema deste experimento foi a classe *Misc* que se espalhou pelo restante do mapa. Neste caso, ao contrário do que aconteceu com o mapa gerado pelo gCLUTO, a saída é reduzir a dimensão do mapa de modo que a classe *Misc* não se espalhe tanto. O novo mapa, de tamanho 8x6 pode ser visto na Figura 5.13.

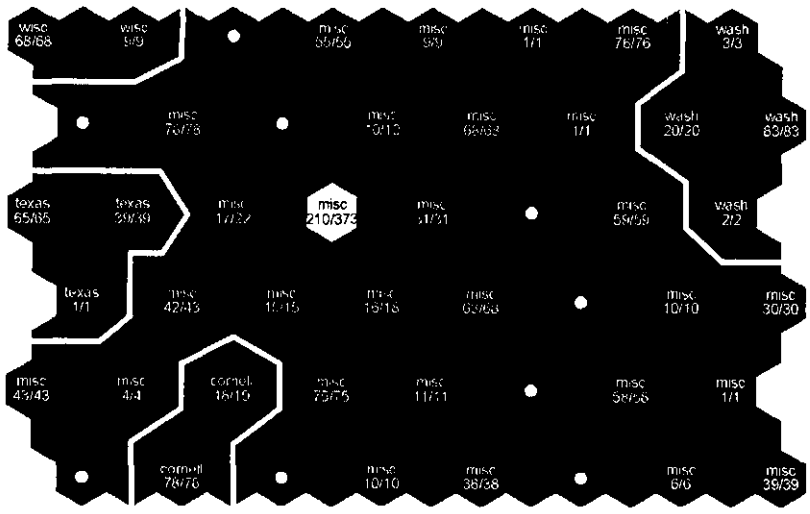
Figura 5.12: Mapa do córpus WebKB gerado pela ferramenta Som_Pak.



Este novo mapa reduz nitidamente a dispersão da classe *Misc*. Ainda, como existem menos neurônios, as células vazias também estão em quantidade menor. Porém, essa redução da dimensão do mapa causou o acúmulo de 373 documentos em um único neurônio, sendo 210 da classe *Misc* (neurônio claro na região central do mapa). Assim, o mapa de dimensão maior apresentou um melhor resultado.

O algoritmo GHSOM foi inicializado com um mapa de tamanho 10x6. O GHSOM levou **34,51 segundos** e consumiu **8,9 Mb**. O mapa possui muitos neurônios vazios

Figura 5.13: Mapa do córpus WebKB de dimensão 8x6.

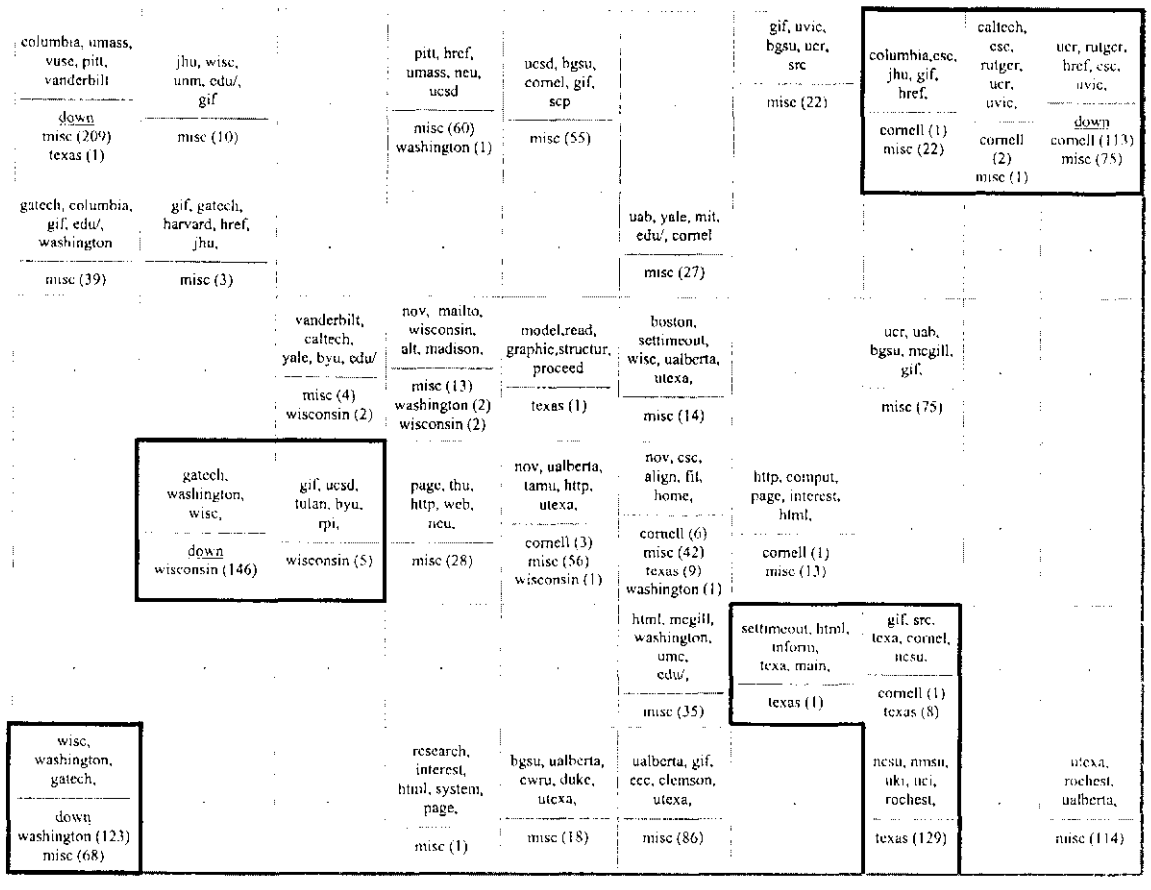


(ver Figura 5.14), neurônios com muitos documentos sem criação de uma nova camada e neurônios com alto valor de entropia (grande quantidade de documentos de 2 classes: *cornell*(113)/*Misc*(75) e *washington*(123)/*Misc*(68)). Apesar disso, os documentos das 4 classes menores ficaram bem próximos. Analisando os grupos delimitados manualmente na Figura 5.14 e calculando a soma dos documentos das classes majoritárias, chega-se a conclusão de que o resultado foi melhor do que o alcançado pelo algoritmo particionador, pois o grupo da classe *cornell* agrupou 116 documentos; o da classe *texas* 138 documentos, o da classe *washington* 123 documentos e o da classe *wisconsin* 151 documentos.

O primeiro mapa criado pelo gCLUTO, com os 5 clusters originais da classe, mostrou que os dados foram bem divididos e que a distância inter-clusters era grande o suficiente para que houvesse particionamento dos clusters maiores. Isto foi verificado no segundo mapa, gerado com 10 clusters. Este novo mapa manteve a taxa de pureza do primeiro enquanto particionou a classe *Misc* em 5 clusters, facilitando a visualização de seu conteúdo.

No mapa inicial gerado com a ferramenta Som_Pak tem-se uma boa delimitação das classes menores. Porém, como o mapa apresentou vários neurônios sem documentos e uma grande dispersão da classe *Misc*, optou-se por um mapa de dimensão reduzida a fim de se minimizar a dispersão. Apesar de atingido o propósito com este novo mapa, surgiu o problema de um neurônio com quantidade substancialmente elevada de

Figura 5.14: Mapa do córpus WebKb gerado pelo GHSOM.



documentos, invalidando o ganho com a redução da dispersão.

A ferramenta GHSOM foi capaz de agrupar os documentos das classes pequenas com mais eficácia do que os algoritmos da ferramenta CLUTO. Isto pode ser verificado na soma das classes majoritárias dos grupos delimitados manualmente no mapa gerado pelo GHSOM. Os problemas detectados neste mapa foram a quantidade de neurônios vazios, o fato de se ter um grande número de documentos em um mesmo neurônio, sem a criação de uma nova camada e os dois neurônios com alta entropia. Apesar disso, consideramos este mapa como a melhor representação dos exemplos deste córpus.

*5.2.3 Experimento 3: Córpus LNAI/*WebMiner

O córpus LNAI/WebMiner foi construído digitalizando-se 277 artigos sobre Raciocínio Baseado em Casos (RBC) e 118 sobre Programação Lógica Indutiva (PLI)

publicados na série *Lecture Notes on Artificial Intelligence* (LNAI) e aplicando-se uma ferramenta de reconhecimento de caracteres - *OCR*. Além disso, foram recuperados da web 205 documentos sobre Recuperação de Informação (RI), em formato PDF/PS, que foram convertidos para texto usando as ferramentas *pdftotext*[10] e *pstotext*[11] . Removendo-se os documentos de RI que não possuem referências bibliográficas, restaram 171 documentos. Ainda, dois dos documentos de RBC também não possuem referências. No total o córpus possui 564 artigos. Cada documento contém o cabeçalho do artigo (título, autores, e-mails, universidades, etc), abstract (resumo) e a seção de referências bibliográficas. O córpus foi pré-processado em diferentes configurações para se poder analisar o resultado do *clustering*:

Resumo: *bag-of-words* composta pelo cabeçalho e pelo *abstract* do artigo;

Texto das Referências: *bag-of-words* da seção de referências bibliográficas foi pré-processada como texto;

Resumo e Texto das Referências: *bag-of-words* criada com o cabeçalho, *abstract* e referências, sendo o documento inteiro tratado como texto, ou seja, sem a extração e indexação das referências;

Bag of References: uso apenas da *bag-of-references*;

Resumo + *Bag of References*: criação da *bag-of-words* do cabeçalho e *abstract* e, após isso, foi feita a junção com a tabela de referências indexadas.

Neste córpus foram encontradas 10.269 referências bibliográficas sendo 6.636 distintas. A *bag-of-words* foi criada com cerca de 1.000 atributos (dependente da configuração) e o vetor da *bag-of-references* não sofreu cortes para se evitar vetores nulos. Tais vetores ocorrem quando o artigo possui referências que são muito pouco citadas no córpus. Essas referências são excluídas quando se faz o corte por freqüência.

Inicialmente, aplicou-se o algoritmo Agglo com cada uma das funções de critério na *bag-of-references* com as medidas binária e *tf-idf*. Esperava-se conseguir bons resultados com este algoritmo, de modo que fosse criada uma árvore pela qual se pudesse navegar encontrando ramificações de sub-áreas de acordo com as referências citadas nos artigos. Os resultados são mostrados na Tabela 5.15.

[10]http://www.foolabs.com/xpdf/download.html
[11]http://www.cs.wisc.edu/~ghost/doc/pstotext.htm

Tabela 5.15: Medida de Pureza média dos Clusters.

	AGGLO (*binário*)		BAGGLO (*binário*)	
Função	**Entropia**	**Pureza**	**Entropia**	**Pureza**
e1	0,922	**0,509**	0,939	0,493
g1	0,937	0,496	0,928	**0,505**
h1	0,527	**0,766**	0,527	**0,766**
h2	0,828	0,539	0,727	**0,663**
i1	0,934	0,500	0,929	**0,505**
i2	0,938	0,495	0,929	**0,505**
clink	0,915	**0,488**	0,943	0,488
slink	0,935	0,498	0,923	**0,507**
upgma	0,938	0,495	0,935	**0,498**
wclink	0,920	**0,488**	0,943	0,488
wslink	0,930	0,504	0,923	**0,507**
wupgma	0,933	**0,502**	0,934	0,500

O melhor resultado foi de cerca de 66% de pureza, um valor razoável e bem abaixo do esperado. A seguir, foram executados os algoritmos particionadores e aglomerativos hierárquicos do *framework* CLUTO na *bag-of-words* criada a partir da configuração Resumo e Texto das Referências e os resultados são mostrados na Tabela 5.16.

Tabela 5.16: Pureza e Entropia do córpus LNAI/*WebMiner* usando CLUTO.

	RB		RBR		AGGLO		BAGGLO	
Função	**Entropia**	**Pureza**	**Entropia**	**Pureza**	**Entropia**	**Pureza**	**Entropia**	**Pureza**
e1	0,068	**0,986**	0,137	0,961	0,446	0,823	0,499	0,690
g1	0,067	**0,986**	0,085	0,982	0,248	0,927	0,241	0,918
h1	0,078	0,982	0,076	**0,982**	0,355	0,883	0,289	0,906
h2	0,067	**0,986**	0,081	0,980	0,311	0,897	0,223	0,938
i1	0,194	**0,940**	0,194	**0,940**	0,250	0,931	0,470	0,807
i2	0,067	**0,986**	0,085	0,982	0,234	0,933	0,300	0,904
clink	-	-	-	-	0,584	**0,730**	0,667	0,677
slink	-	-	-	-	0,943	**0,489**	0,943	**0,489**
upgma	-	-	-	-	0,911	0,518	0,46	**0,775**
wclink	-	-	-	-	0,584	0,73	0,667	0,677
wslink	-	-	-	-	0,943	**0,489**	0,943	**0,489**
wupgma	-	-	-	-	0,243	0,934	0,097	**0,979**

Os melhores resultados foram obtidos com o algoritmo RB, que atingiu 98% de pureza em 5 das 6 execuções. Utilizando a função *h1*, o valor de entropia foi um pouco superior ao do resultado encontrado pelo algoritmo RBR, apesar de ambos terem a mesma medida de pureza. Os gráficos equivalentes à Tabela 5.16 são mostrados nas Figuras 5.15 e 5.16.

Pode-se perceber na Figura 5.15 que os algoritmos RB e RBR obtiveram resultados bem similares com as funções de *e1* a *i2* ao contrário dos algoritmos Agglo e Bagglo. Nos demais experimentos o Agglo gerou poucos resultados superiores ao Bagglo mas, neste experimento a situação se inverteu. Já no gráfico da Figura 5.16, pode-se perceber

que Agglo e Bagglo conseguem resultados semelhantes apenas usando as funções *slink* e *wslink*, que geraram os agrupamentos de pior qualidade deste experimento.

Figura 5.15: Gráfico dos resultados da execução de algoritmos de *clustering* no córpus LNAI/*WebMiner* (funções *e1* até *i2*).

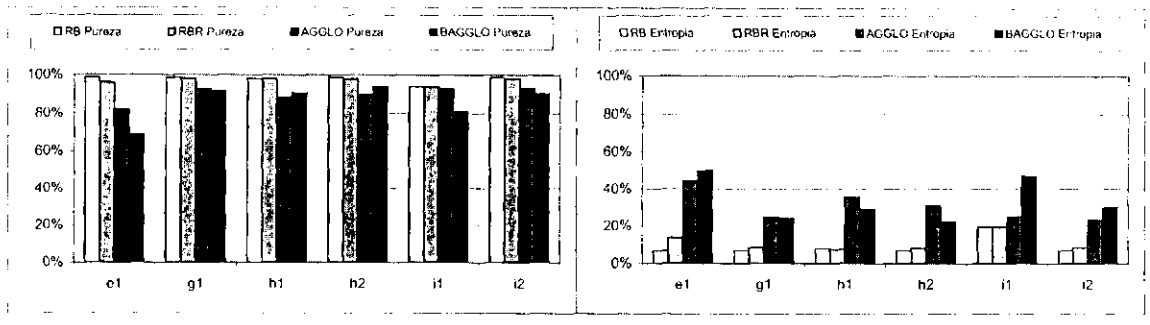
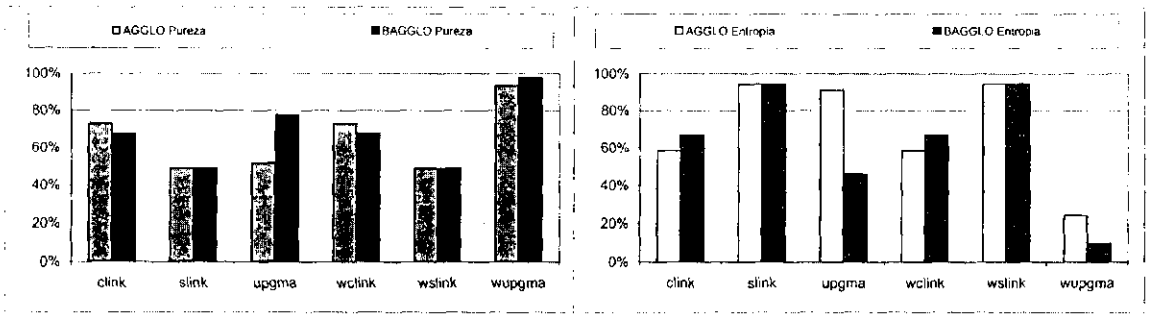


Figura 5.16: Gráfico dos resultados da execução de algoritmos de *clustering* no córpus LNAI/*WebMiner* (funções *clink* até *wupgma*).



Entre os algoritmos aglomerativos, o Agglo superou o Bagglo em 6 das 12 execuções, 2 resultados ficaram iguais (*slink* e *wslink*) e 4 resultados a favor do Bagglo. Este, por sua vez, conseguiu o melhor resultado (função *wupgma*). Houve uma grande variação na pureza dos clusters encontrados pelos algoritmos hierárquicos. Além disso, o tempo computacional foi de 2 a 7 vezes superior ao dos algoritmos particionadores (Tabela 5.17).

Como neste experimento os melhores resultados foram alcançados com os algoritmos RB (função *i2*) e Bagglo (função *wupgma*), optou-se por estas duas técnicas para se testar as demais configurações descritas anteriormente. Tais algoritmos foram aplicados nos dados e os resultados dos experimentos usando os pesos *tf-idf* e binário podem ser vistos na Tabela 5.18.

Tabela 5.17: Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus LNAI/ *WebMiner*.

	RB		RBR		AGGLO		BAGGLO	
Função	**T**	**M**	**T**	**M**	**T**	**M**	**T**	**M**
e1	2,23	11,00	2,29	11,00	7,73	15,00	11,42	15,00
g1	2,16	11,00	2,22	11,00	7,72	15,00	11,47	15,00
h1	2,23	11,00	2,31	11,00	12,46	15,00	16,20	15,00
h2	2,12	11,00	2,17	11,00	14,19	15,00	17,91	15,00
i1	2,19	13,00	2,23	13,00	7,63	15,00	11,46	15,00
i2	2,06	11,00	2,12	11,00	7,65	15,00	11,35	15,00
clink	-	-	-	-	7,55	15,00	11,35	15,00
slink	-	-	-	-	7,63	15,00	11,44	15,00
upgma	-	-	-	-	7,59	15,00	11,30	15,00
wclink	-	-	-	-	7,58	15,00	11,35	15,00
wslink	-	-	-	-	7,62	15,00	11,28	15,00
wupgma	-	-	-	-	7,63	15,00	11,45	15,00

Tabela 5.18: Medida de Pureza média dos Clusters.

Dados de Entrada	**Atributos**	**RB**	**BAGGLO**
Resumo (*tf-idf*)	1020	91,8%	**92,2%**
Texto das Referências (*tf-idf*)	1001	**98,0%**	97,5%
Resumo e Texto das Refs (*tf-idf*)	1033	**98,6%**	97,9%
Bag of References (*tf-idf*)	6636	92,0%	**92,4%**
Resumo + *Bag of References* (*tf-idf*)	7656	93,1%	93,1%
Resumo (binário)	1020	**91,1%**	83,7%
Texto das Referências (binário)	1001	**98,4%**	86,0%
Resumo e Texto das Refs (binário)	1033	**98,6%**	89,7%
Bag of References (binário)	6636	**95,2%**	91,8%
Resumo + *Bag of References* (binário)	7656	**94,9%**	79,6%

Como esperado, o uso das referências bibliográficas juntamente com o resumo melhorou o resultado do *clustering*. Mais do que isso, elas conseguem, individualmente, um resultado melhor, apesar da *bag-of-references* ter atingido um resultado inferior ao do processamento das referências como texto.

Ao analisar os termos mais freqüentes nos clusters, percebeu-se que os nomes dos autores e simpósios são capazes de discriminar as classes dos documentos utilizados no experimento, explicando o melhor resultado do uso to texto das referências em relação ao resumo.

É interessante notar que, com o algoritmo RB, os resultados utilizando peso binário foram de qualidade superior aos de peso *tf-idf*. Não foi possível identificar o que causou essa característica visto que, com o algoritmo BAGGLO, os resultados reproduzem o que é comumente informado na literatura, ou seja, que o peso *tf-idf* apresenta melhores resultados do que o binário.

Na Tabela 5.19 tem-se as matrizes de confusão para duas configurações: *bag-of-references* (binário) e resumo e texto das referências como *bag-of-words* (*tf-idf*) com o algoritmo RB. Percebe-se que o resultado inferior no uso da *bag-of-references* deveu-se, principalmente, à classe RI, que teve 18 erros contra apenas 3 quando foi usado o texto das referências. Ao verificar as referências dos documentos que foram assinalados erroneamente, percebeu-se que a maioria era de freqüência um, ou seja, não se repetiam em nenhum outro documento. Assim, o vetor da *bag-of-references* correspondente a esse artigo não encontrou nenhum outro similar e terminou sendo adicionado ao cluster incorreto.

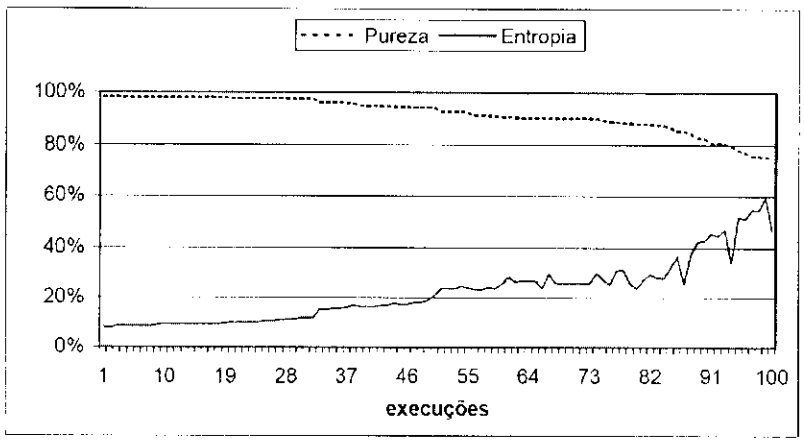
Tabela 5.19: Matrizes de Confusão.

	Bag-of-refs			*Bag-of-words*		
Cluster	**RBC**	**PLI**	**RI**	**RBC**	**PLI**	**RI**
0	4	**117**	*12*	3	**118**	2
1	4	0	**153**	2	0	**168**
2	**267**	1	*6*	**270**	0	1

Para este córpus o *Spherical K-Means* foi executado 100 vezes com a opção *random seed* usando a *bag-of-words* do Resumo com Texto das Referências, com peso *tf-idf*. Os resultados foram ordenados em ordem decrescente de pureza e podem ser vistos na Figura 5.17.

De acordo com a média e o desvio padrão das medidas de pureza e entropia, percebe-se-se um resultado de boa qualidade, apesar de um pouco inferior aos melhores obtidos com o CLUTO (Tabela 5.21). O tempo computacional foi baixo e próximo ao dos algoritmos particionadores.

Figura 5.17: Gráfico de 100 execuções do *Spherical K-Means* no córpus LNAI/*WebMiner*.



No experimento seguinte, foi utilizada uma implementação do *Spherical K-Means*, capaz de trabalhar com duas visões simultaneamente (*Multi-view Spherical K-Means*). Seguindo a metodologia, este algoritmo foi executado 100 vezes com diversas configurações, como na Tabela 5.18, utilizando, ainda, duas visões. Os resultados de pureza média e desvio padrão são mostrados na Tabela 5.20.

Tabela 5.20: Medida de Pureza média e Desvio Padrão dos Clusters usando *o Multi-view Spherical K-Means*.

Visão 1	**Visão 2**	***tf-idf***	**D.P.**	**binário**	**D.P.**
Resumo		0,782	0,182	**0,830**	0,164
Texto das Referências		0,911	0,152	**0,982**	0,110
Bag-of-References		0,488	0,009	**0,594**	0,086
Resumo e Texto das Refs		0,918	0,178	**0,956**	0,158
Resumo · *Bag-of-References*		0,832	0,178	**0,848**	0,179
Resumo	Texto das Refs	0,933	0,137	**0,966**	0,115
Resumo	*Bag-of-References*	0,840	0,189	**0,899**	0,120

Nesta tabela é possível perceber que os resultados com peso binário são melhores do que com peso *tf-idf*, como no experimento usando o algoritmo RB (ver Tabela

5.18). Porém, com o *Multi-view Spherical K-Means* a qualidade dos resultados foi inferior, principalmente com relação à *Bag-of-References*. Pode-se supor que, dada a alta esparsidade desta representação vetorial, o algoritmo *K-Means* não foi capaz de gerar bons centróides. Apesar disso, é importante notar a melhora nos valores de pureza quando são utilizadas duas visões, mesmo sendo uma diferença pequena.

Na Tabela 5.21 tem-se os melhores resultados dos experimentos para este córpus. A pureza da solução encontrada pelo algoritmo RB (*i2*) e o baixo tempo computacional fazem desta técnica a melhor escolha para este córpus. Porém o consumo de memória foi bastante elevado se comparado ao *Spherical K-Means*.

Tabela 5.21: Melhores Entropia e Pureza do córpus LNAI/*WebMiner* usando CLUTO e Entropia e Pureza médias do *Spherical K-Means*.

Algoritmo	Entropia	D.P.	Pureza	D.P.	T	M
RB (*i2*)	0,067	-	0,986	-	2,06	11,00
BAGGLO (*wupgma*)	0,097	-	0,979	-	11,45	15,00
Spherical K-Means	0,220	0,124	0,920	0,062	1,75	2,27

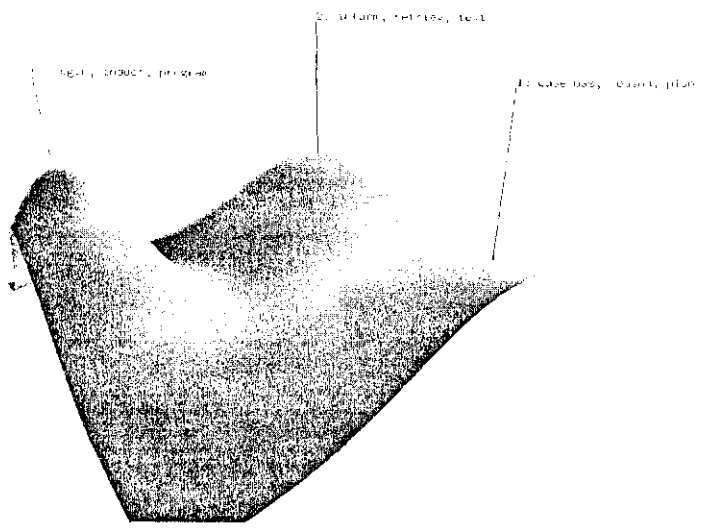
Como a quantidade de documentos neste córpus é pequena (564 documentos), não há razão para não se usar a ferramenta CLUTO. Porém, multiplicando-se a quantidade de documentos, pode ser necessária a troca de algoritmo para algum outro que seja mais eficiente na questão de complexidade de espaço. Neste caso, os algoritmos *Spherical K-Means* ou sua versão *Multi-view* podem ser uma solução.

Em 74 das 100 execuções o algoritmo *Spherical K-Means* conseguiu uma pureza média acima de 90%, sendo que em 32 delas a pureza foi de 98%. Ainda, em apenas 6 execuções a pureza foi abaixo de 80%. Considerando o tempo de processamento e o consumo de memória, este algoritmo pode ser considerado satisfatório para este córpus.

Mapas

O mapa em 3D do córpus LNAI/*WebMiner* (*bag-of-words*) usando RB/*i2*, criado em **4,63 segundos,** é mostrado na Figura 5.18. Analisando-se as montanhas neste mapa, é visível a boa divisão dos documentos que o algoritmo conseguiu encontrar. Mais uma vez, como aconteceu no córpus WebKb, é interessante notar o conjunto de termos mais descritivos nos clusters, explicando a boa divisão dos documentos.

Figura 5.18: Mapa 3D do córpus LNAI/ *WebMiner* (*bag-of-words*).

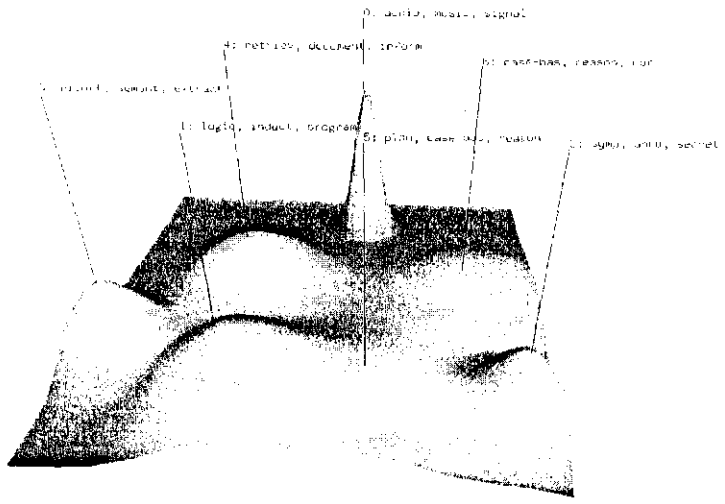


Apesar de ser o menos puro (95,2%), o cluster 0, representando a classe PLI, tem um desvio padrão baixo e uma boa similaridade entre seus documentos (cor vermelha e maior altura). Os outros dois clusters possuem valores próximos de pureza, similaridade interna e desvio padrão. Isso fica evidente no formato de suas montanhas nos gráficos.

Da mesma maneira que ocorreu com o córpus WebKb, o córpus LNAI *WebMiner* permite que um número maior de clusters seja utilizado para encontrar sub-áreas contidas no córpus. Já na Figura 5.19, tem-se o resultado do *clustering* com 7 clusters. Neste novo mapa a pureza média teve uma pequena queda de 98.6% para 96,8%.

Nesta nova configuração a classe PLI se manteve praticamente intacta no cluster 1 (109 documentos); a classe RBC foi particionada entre clusters 5 e 6 (111 e 154 documentos, respectivamente); e a classe RI se dividiu entre os clusters 0, 2, 3 e 4 (17, 35, 35 e 82 documentos, respectivamente). Na Figura 5.19 o cluster 0 (classe RI) se sobressai. Os termos descritivos são uma boa indicação de seu conteúdo. Além disso, sua altura indica que os documentos ali presentes são bastante similares. A montanha formada pela união dos cluster 1 (PLI) e 5 (RBC) possuem um grande volume e baixa altura pois, apesar de uma pureza próxima a 100%, a similaridade interna é baixa. Outro cluster que chama a atenção é o número 2 que, apesar de ter documentos agrupados como RI, este cluster se encontra do lado oposto dos demais documentos da mesma classe e próximo aos documentos de RBC. Os termos mais discriminativos deste grupo não apresentam informação suficiente sobre seu conteúdo. Analisando

Figura 5.19: Mapa 3D do córpus LNAI/*WebMiner* (*bag-of-words*) - 7 clusters.



os documentos encontramos o tópico **Information-theoretic private information retrieval**, relacionado com criptografia, o que explica a inclusão do termo *secret*.

Tabela 5.22: Termos mais discriminativos da *bag-of-references* com 3 clusters.

Cluster	Termos mais discriminativos
0 - PLI	Muggleton_3106, Lavrac_3005, Plotkin_115
1 - RI	Mitchell_4763, Aha_4301, Tzanetakis_5803
2 - RBC	Quinlan_2476, Aamodt_2770, Smyth_3390

Com relação à *bag-of-references*, a ferramenta gCLUTO não conseguiu gerar um mapa com 3 clusters. Aparentemente pelo fato de os valores de similaridade interna serem muito baixos. Por este motivo são apresentadas apenas os termos (referências) mais discriminativas dos clusters (ver Tabela 5.22). Os números nos termos desta tabela equivalem ao ID da referência na tabela que contém todas as referências do córpus. As referências mais discriminativas são mostradas em sua forma original na Tabela 5.23.

É possível perceber que elas são um bom indicativo do conteúdo dos clusters. No cluster 0 tem-se 3 referências sobre PLI. No cluster 2 tem-se 2 referências sobre RBC e uma sobre um classificador (C45), que é usado em aprendizagem de máquina.

Tabela 5.23: Referências dos termos mais discriminativos da *bag-of-references* com 3 clusters.

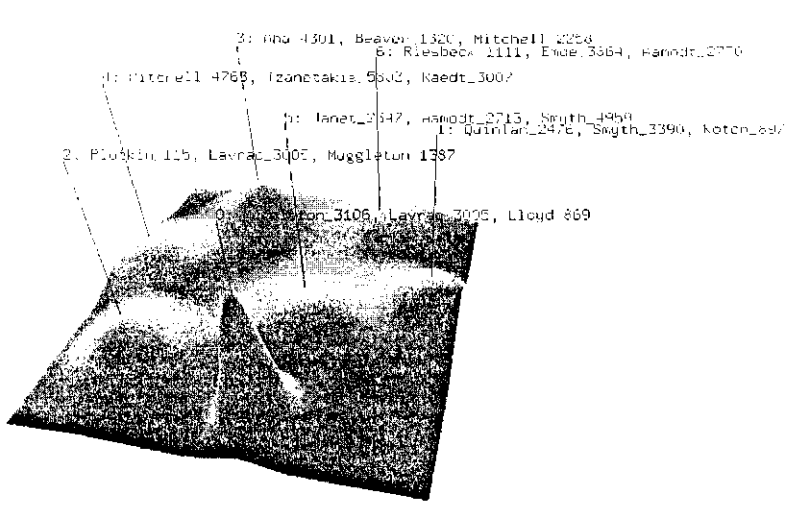
Termos	Referência equivalente
Muggleton_3106	Muggleton, S. & De Raedt, L. Inductive Logic Programming: Theory and methods Journal of Logic Programming 19 (20) (1994).
Lavrac_3005	Lavrac, N & Dzeroski, S. Inductive Logic Programming: Techniques and Applications. Ellis Horwood. 1994.
Plotkin_115	G. D. Plotkin. A Note on Inductive Generalization. Machine Intelligence, 5:153 163, 1970.
Mitchell_4763	T. Mitchell. Machine Learning, McGraw Hill. 1997.
Aha_4301	Aha, D. W. and Breslow, L. Refining conversational case libraries. In Proceedings of ICCBR-97, pp 267-276, Providence RI, USA, 1997.
Tzanetakis_5803	G. Tzanetakis and P. Cook. Marsyas: A framework for audio analysis. Organised Sound, 4(3), 2000.
Quinlan_2476	Quinlan, J. R. C45: Programs for Machine Learning. Morgan Kaufmann, San Mateo, California (1993).
Aamodt_2770	A. Aamodt & E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AICOM Vol.7 Nr. 1, pp.39-59, March 1994.
Smyth_3390	B. Smyth and M. Keane Remembering to forget : A competence-preserving case deletion policy for case-based reasoning systems. International Joint Conference on Artificial Intelligence, 1:377382, 1995.

Já no cluster 1, que agrupou documentos de RI, tem-se uma referência citada em várias áreas da literatura (Mitchell), uma sobre análise de áudio e uma de RBC (Aha). Procurando-se o motivo pelo qual a referência de Aha se encontra entre os termos mais discriminativos do cluster 1, encontramos documentos de RBC que tratam de RI, como **Bruce McLaren and Kevin Ashley. Case Representation, Acquisition, and Retrieval in SIROCCO.**

Como o resultado com 3 clusters foi de boa qualidade decidimos realizar o *clustering* das *bag-of-references* com 7 clusters. A solução encontrada é mostrada no mapa da Figura 5.20. Neste mapa a classe RBC é majoritária nos clusters 1, 5 e 6; a classe PLI nos clusters 0 e 2; e a classe RI nos clusters 3 e 4. Nesta solução, que possui 95,2% de pureza média, o único cluster com pureza abaixo de 94% é o cluster 2 (pureza de 81,7%), que possui 67 documentos de PLI, 3 de RBC e 12 e RI. Esta representação

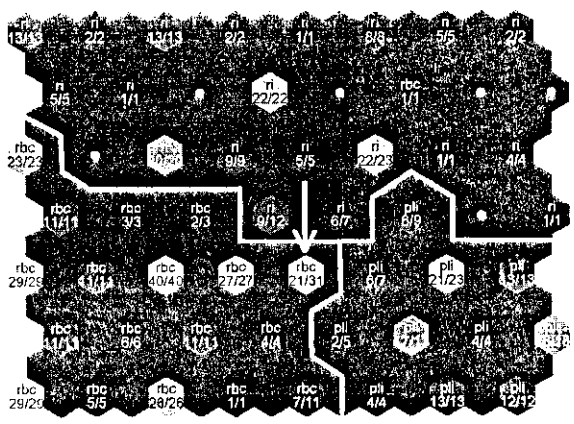
pode ser considerada muito boa, dada a esparsidade dos dados. Se cada documento possui em média 15 referências e a tabela atributo/valor possui 6.636 atributos, tem-se 99.77% de atributos nulos. Este valor é variável, pois alguns documentos de RI são dissertações e outros apresentam menos de 10 referências.

Figura 5.20: Mapa 3D do córpus LNAI/ *WebMiner* (*bag-of-references*) - 7 clusters.



Nos experimentos a seguir são apresentados apenas resultados com a *bag-of-words*. Os algoritmos SOM e GHSOM não foram capazes de lidar com a *bag-of-references* de maneira eficaz, gerando resultados muito ruins.

Figura 5.21: Mapa do córpus LNAI/ WebMiner gerado pela ferramenta Som_Pak.



Na Figura 5.21, tem-se o resultado da aplicação do SOM em um mapa de dimensão 8x7 para distribuir os documentos deste córpus. O processo consumiu **1min 16,5s** e **3,3**

Mb. Como no experimento com o córpus WebKb, o algoritmo SOM gerou um mapa bem delimitado. Os documentos de PLI foram posicionados na extremidade direita do mapa, com um particionamento ligeiramente equilibrado dos documentos entre os neurônios. Os documentos de RBC, posicionados na extremidade esquerda, não ficaram tão bem divididos, visto que 6 neurônios agruparam menos de 10 documentos cada um, 1 neurônio com 40 documentos e 5 com cerca de 30 documentos. Já a classe de RI, que ocupou quase metade do mapa, se caracteriza por ter 6 neurônios sem documentos e 5 com apenas 1 documento, sendo que em um destes o documento é da classe RBC.

No experimento seguinte, também com a *bag-of-words*, o algoritmo GHSOM gerou um mapa semelhante ao do SOM (ver Figura 5.22): classe PLI compactada; classes RBC e RI espalhadas pelo mapa e neurônios vazios na área delimitada por documentos de RI.

Figura 5.22: Mapa do córpus LNAI/*WebMiner* gerado pelo GHSOM.

network, wireless, acoust, signal, music rbc (1) ri (15)		signal, case-bas, reason, network, mobil rbc (1)	design, case, signal, proc, imag rbc (11) ri *(1)*	case, plan, adapt, design, reason rbc (27)	plan, case, adapt, reason, cbr down *rbc (41)*	solv, case, reason, page, cbr rbc (19)	similar, cbr, case, reason, solv down *rbc (56)*
acoust, signal, music, audio, mobil ri (5)			case-bas, case, retriev, reason, user rbc (2)	design, adapt, plan, softwar, case-bas rbc (4)	softwar, plan, reason, case, case-bas rbc (11)	learn, page, design, adapt, cbr rbc (8)	case, learn, similar, cbr, design rbc (14)
proc, page, mobil, imag, inform pli (1) ri (1)		learn, proceed, case, collabor, recommend rbc (3) ri (2)	learn, case, fuzzi, collabor, adapt rbc (8)	softwar, design, case, adapt, proc rbc (7)	similar, proc, learn, plan, case rbc (13)	reason, learn, adapt, case-bas, cbr rbc (9)	design, cbr, solv, induct, case rbc (15)
automat, linguist, audio, network, text ri (1)		automat, model, algorithm, develop, classif ri (4)	adapt, decis, springer, develop, structur rbc (1) pli (1) ri (1)	algorithm, decis, knowledg, reason, acm rbc (1) pli (1)	case, adapt, knowledg, fuzzi, design rbc (5)	case, editor, springer, muggleton, design rbc (6)	artifici, similar, solv, adapt, springer rbc (5)
semant, audio, proceed, language, automat down rbc (1) ri (36)	user, page, network, proceed, data ri (1)	ieee, softwar, case-bas, commun, cryptologi rbc (1) ri (3)		case, rule, annual, nois, note ri (1)	machin, secret, algorithm, theori, editor rbc (1)	ilp, machin, proc, reason, muggleton pli (1)	raedt, reason, ilp, case-bas, program rbc (1) pli (6)
proceed, word, cluster, extract, dictionari ri (18)				vol, volum, editor, lectur, commun ri (2)	ilp, program, induct, logic, theori pli (1)	ilp, induct, learn, page, program pli (2)	page, raedt logic, learn, program rbc (1) pli (9)
databas, proceed, automat, acm, corpora down ri (51)			semant, languag, statist, proceed, algorithm ri (10)	text, ilp, inform, pir, retriev ri (19)		learn, ilp, page, raedt, induct pli (4)	learn, muggleton, induct, raedt, ilp down pli (92)

Uma grande diferença são os neurônios com dezenas de documentos como o situado no canto inferior direito que contém 92 documentos de PLI. Na realidade, essa característica é importante pois tais neurônios acabam necessitando da criação de um novo mapa para a distribuição dos documentos. A execução da ferramenta levou **5,75 segundos** e consumiu **3,9 Mb**.

Todos os mapas gerados para este córpus podem ser considerados boas representações dos documentos. Os mapas criados com o gCLUTO fornecem informações importantes, como o grau de similaridade interna do cluster. A boa divisão do córpus LNAI/*WebMiner* com um pequeno número de clusters indica que é possível usa um maior número de partições a fim de se encontrar sub-áreas. Isto pode ser conferido na Figura 5.19. Apesar de uma pequena redução no valor médio de pureza, causada pela criação de clusters menores que não são 100% puros, o novo mapa representa uma boa separação dos documentos. Pode-se perceber que as classes PLI e RI têm características opostas. Enquanto a primeira é bem compacta (1 cluster), a segunda é bastante abrangente (5 clusters).

A solução encontrada pelos algoritmos da ferramenta CLUTO usando a *bag-of-references* indica que esta representação de documentos fornece informações interessantes sobre o relacionamento entre os artigos. Com apenas 3 clusters, os termos mais discriminativos - no caso, referências - do cluster correspondente à classe RI, não foram capazes de explicar seu conteúdo, por se tratar de documentos com usos diversos de Recuperação de Informação. Porém, ao se incrementar a quantidade de clusters, os termos passaram a indicar melhor os documentos do córpus. Ainda, os algoritmos SOM e GHSOM não foram capazes de lidar com esses dados, tratando apenas da *bag-of-words*.

O mapa gerado pelo algoritmo SOM é de fácil delimitação. Porém, há um desequilíbrio na quantidade de documentos que cada neurônio representa. Existem 5 neurônios com apenas 1 documento e 6 neurônios vazios na região da classe RI, enquanto um único neurônio agrupou 40 documentos de RBC. Já no mapa do GHSOM tem-se a mesma quantidade de neurônios que possuem 1 documento, mas o dobro de neurônios vazios, devido ao fato de um acúmulo ainda maior de documentos em um outros neurônios. Tal acúmulo não é problema para o algoritmo GHSOM, visto que ele é capaz de criar novos mapas para representar esses documentos.

O mapa criado com o gCLUTO teve menor custo computacional mas o maior

consumo de memória. Já o GHSOM necessitou de um pouco mais de tempo, mas consumiu 35% da memória necessária pelo CLUTO. Como o mapa gerado pelo GHSOM agrupou os documentos de forma satisfatória, permitindo uma navegação hierárquica pelos exemplos, consideramos que, para este córpus, esta ferramenta apresentou o melhor mapa.

5.2.4 *Experimento 4:* Datasets *do CLUTO*

Neste experimento foram utilizados 4 *datasets* fornecidos pelo *framework* CLUTO. Estes datasets são disponibilizados em formato de matriz esparsa. Como não se tem acesso aos documentos, serão utilizados apenas os algoritmos do próprio CLUTO, visto que não poderá ser feita uma análise subjetiva dos exemplos. Nas tabelas, a seguir, que contêm os resultados da aplicação das técnicas de *clustering*, é apresentado apenas o valor de pureza média. A entropia é apresentada na tabela final para o melhor resultado em cada *dataset*.

1. *Classic*: é composto por quatro classes que correspondem aos *datasets* CACM, CISI, CRANFIELD e MEDLINE, num total de 7.089 exemplos e 12.009 atributos.

2. FBIS: possui 17 classes numeradas como 202, 187, 189, 100, 108, 142, 118, 221, 161, 111, 240, 3, 11, 12, 119, 95 e 4, num total de 2.463 exemplos e 2.000 atributos.

3. *Reuters*: possui 13 classes: *housing*, *money*, *trade*, *reserves*, *cpi*, *interest*, *gnp*, *retail*, *ipi*, *jobs*, *lei*, *bop* e *wpi*, num total de 1.504 exemplos e 2.886 atributos.

4. WAP: possui 20 classes: *people*, *television*, *health*, *media*, *art*, *film*, *business*, *cable*, *culture*, *music*, *politics*, *sports*, *review*, *technology*, *stage*, *entertainment*, *online*, *industry*, *variety* e *multimedia*, num total de 1.560 exemplos e 8.440 atributos.

Na Tabela 5.24 são apresentados os valores de pureza da execução dos algoritmos particionadores e aglomerativos nos córpus *Classic* e FBIS. Esses algoritmos obtiveram resultados semelhantes no córpus *Classic*, encontrando clusters de pureza próxima a 80%. Este valor pode ser considerado bom. O algoritmo RB, apenas com otimização local, obteve resultado ligeiramente superior ao RBR neste *dataset*. O algoritmo Bagglo também não foi muito superior ao Agglo. Ou seja, os resultados foram bastante equilibrados neste experimento.

Já no córpus FBIS os valores de pureza não chegaram a 70%. Os algoritmos RBR e Bagglo obtiveram, novamente, melhores resultados do que suas versões mais simples (RB e Agglo) sem diferença significativa na maioria dos casos. Porém, é importante notar que, com as funções *e1*, *h2* e *i1*, o algoritmo Agglo encontrou clusters com maior pureza que o algoritmo Bagglo.

Tabela 5.24: Pureza dos *datasets CLASSIC* e FBIS.

	CLASSIC				FBIS			
Função	RB	RBR	AGGLO	BAGGLO	RB	RBR	AGGLO	BAGGLO
e1	0,724	**0,770**	0,734	0,654	0,644	**0,688**	0,648	0,634
g1	0,734	0,719	0,632	**0,734**	**0,651**	0,651	0,62	0,648
h1	**0,785**	0,784	0,747	-	0,683	**0,692**	0,628	0,64
h2	0,718	**0,747**	0,742	-	0,688	**0,694**	0,65	0,629
i1	**0,787**	0,783	0,747	0,523	0,690	**0,691**	0,61	0,602
i2	0,747	0,748	**0,757**	0,698	0,676	**0,690**	0,634	0,658
clink	-	-	0,472	**0,481**	-	-	0,596	**0,672**
slink	-	-	**0,452**	0,451	-	-	0,211	**0,212**
upgma	-	-	0,451	**0,762**	-	-	0,624	**0,691**
wclink	-	-	0,459	**0,497**	-	-	0,596	**0,672**
wslink	-	-	**0,452**	0,451	-	-	0,211	**0,212**
wupgma	-	-	0,756	**0,776**	-	-	0,636	**0,673**

Tabela 5.25: Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus *Classic*.

	RB		RBR		AGGLO		BAGGLO	
Função	T	M	T	M	T	M	T	M
e1	1,60	6,68	1,83	6,68	68,01	483,00	70,85	483,00
g1	1,96	7,32	1,99	7,32	67,86	483,00	76,71	483,00
h1	2,07	7,33	2,29	7,33	14.341,97	198,00	-	-
h2	1,62	6,68	1,84	6,68	14.187,45	198,00	-	-
i1	1,69	7,28	1,84	7,28	63,18	483,00	72,61	483,00
i2	1,50	7,24	1,48	7,24	63,39	483,00	66,49	483,00
clink	-	-	-	-	46,91	483,00	49,38	483,00
slink	-	-	-	-	51,66	483,00	53,71	483,00
upgma	-	-	-	-	52,47	483,00	53,92	483,00
wclink	-	-	-	-	52,76	483,00	57,37	483,00
wslink	-	-	-	-	57,33	483,00	61,87	483,00
wupgma	-	-	-	-	59,53	483,00	62,31	483,00

Algoritmos particionadores possuem custo computacional bem inferior aos aglomerativos hierárquicos, principalmente no *dataset Classic*. Isso fica bem visível na

Tabela 5.25. O tempo consumido pelas funções *h1* e *h2* nos algoritmos aglomerativos é consideravelmente elevado, apesar da boa qualidade dos resultados. Não se tem os valores para o algoritmo BAGGLO no *dataset Classic* porque o processo foi finalizado pelo sistema operacional antes de seu término.

O tempo e a memória consumidos no *dataset* FBIS são mostrados na Tabela 5.26. Mais uma vez, os algoritmos particionadores foram mais eficientes e as funções *h1* e *h2* consumiram um tempo expressivo.

Os melhores resultados com o *dataset Classic* foram RB(*i1*) com pureza de 78,7% e Bagglo(*wupgma*) 77,6%. Como os algoritmos hierárquicos possuem custo computacional elevado, o algoritmo mais indicado para este córpus é o RB com a função de otimização *i1*.

Para o *dataset* FBIS os clusters com melhor qualidade foram encontrados pelo algoritmo RBR com a função *h2*. O algoritmo RB com a função *i1*, resultou em clusters com uma pureza apenas 0,4% menor em um tempo 25% menor (0,86 segundos). Para um *dataset* pequeno e uma máquina robusta, tal quantidade de tempo é insignificante. Assim, é recomendável o algoritmo RBR. Caso contrário, seria interessante avaliar melhor a questão custo computacional/qualidade dos clusters e optar pelo algoritmo RB.

Tabela 5.26: Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus FBIS.

	RB		RBR		AGGLO		BAGGLO	
Função	**T**	**M**	**T**	**M**	**T**	**M**	**T**	**M**
e1	2,41	9,26	2,82	9,26	10,43	61,00	11,37	61,00
g1	2,81	9,56	3,84	9,56	15,24	61,00	17,10	61,00
h1	2,44	9,60	3,28	9,60	455,29	61,00	455,67	61,00
h2	2,80	9,48	2,85	9,48	453,17	61,00	453,45	61,00
i1	2,46	9,83	2,72	9,83	9,22	61,00	9,95	61,00
i2	2,22	9,56	2,40	9,56	9,47	61,00	9,86	61,00
clink	-	-	-	-	9,85	61,00	12,42	61,00
slink	-	-	-	-	6,71	61,00	11,72	61,00
upgma	-	-	-	-	10,14	61,00	9,31	61,00
wclink	-	-	-	-	10,17	61,00	12,88	61,00
wslink	-	-	-	-	9,60	61,00	11,93	61,00
wupgma	-	-	-	-	11,52	61,00	9,50	61,00

A execução dos algoritmos no *dataset Reuters* resultou em clusters de qualidade razoável com pureza entre 60% e 70% na maioria dos casos (ver 5.27). Utilizando a função *slink*, mais uma vez, praticamente todos os documentos foram agrupados em um mesmo cluster. Esta função não gerou bons resultados em nenhum dos experimentos. Já a função *e1*, que gerou resultados razoáveis nos demais experimentos, superou as funções *i* e *h*, que, até então, apresentavam os melhores resultados.

No *dataset* WAP os resultados foram melhores do que no *Reuters*, onde o melhor resultado foi de 67,8% de pureza. Como pode ser visto na Tabela 5.27, o algoritmo Agglo teve dificuldade em encontrar clusters com pureza acima de 60%, ao contrário do Bagglo que, por fazer uso de uma solução do algoritmo RB, obteve três dos melhores resultados entre as funções de *e1* a *i2*.

Tabela 5.27: Pureza dos *datasets* Reuters e WAP.

	Reuters				WAP			
Função	RB	RBR	AGGLO	BAGGLO	RB	RBR	AGGLO	BAGGLO
e1	0,652	**0,678**	0,608	0,656	0,699	**0,715**	0,567	0,692
g1	0,633	**0,633**	0,578	0,616	0,65	0,637	0,554	**0,687**
h1	0,623	0,638	0,603	**0,663**	0,693	**0,713**	0,621	0,69
h2	0,624	**0,668**	0,578	0,664	0,678	**0,69**	0,576	**0,69**
i1	0,614	0,612	0,554	**0,622**	0,615	0,61	0,561	**0,693**
i2	0,630	0,634	0,577	**0,653**	**0,696**	0,696	0,61	0,694
clink	-	-	0,431	**0,515**	-	-	0,576	**0,683**
slink	-	-	0,410	**0,412**	-	-	0,233	**0,292**
upgma	-	-	0,475	**0,616**	-	-	0,54	**0,684**
wclink	-	-	0,418	**0,606**	-	-	0,576	**0,683**
wslink	-	-	0,410	**0,412**	-	-	0,233	**0,292**
wupgma	-	-	0,593	**0,634**	-	-	0,645	**0,685**

Nas Tabelas 5.28 e 5.29 tem-se o tempo computacional e a memória consumidos pelos algoritmos nos córpus *Reuters* e WAP. Com o *Reuters* os algoritmos RB e RBR tiveram tempos bem próximos entre si. Os algoritmos Agglo e Bagglo também tiveram tempos similares. Já no córpus WAP o algoritmo RBR precisou de quase o dobro do tempo do algoritmo RB para realizar o *clustering* nas diversas funções de critério. O mesmo vale para o Bagglo em relação ao Agglo. Mais uma vez, as funções *h1* e *h2* necessitaram de um tempo computacional substancialmente alto.

Tabela 5.28: Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus *Reuters*.

	RB		RBR		AGGLO		BAGGLO	
Função	**T**	**M**	**T**	**M**	**T**	**M**	**T**	**M**
e1	0,48	2,46	0,56	2,46	2,78	23,00	3,46	23,00
g1	0,53	2,48	0,61	2,48	3,24	23,00	3,83	23,00
h1	0,63	2,63	0,73	2,63	107,00	23,00	106,84	23,00
h2	0,51	2,47	0,58	2,47	103,41	23,00	103,56	23,00
i1	0,57	2,61	0,57	2,61	2,93	23,00	3,28	23,00
i2	0,40	2,48	0,42	2,48	2,55	23,00	2,95	23,00
clink	-	-	-	-	2,38	23,00	2,98	23,00
slink	-	-	-	-	2,27	23,00	2,87	23,00
upgma	-	-	-	-	2,38	23,00	2,86	23,00
wclink	-	-	-	-	2,39	23,00	3,05	23,00
wslink	-	-	-	-	2,28	23,00	2,94	23,00
wupgma	-	-	-	-	2,67	23,00	2,93	23,00

Tabela 5.29: Tempo (T) em segundos e Memória (M) em Mb usando CLUTO no córpus WAP.

	RB		RBR		AGGLO		BAGGLO	
Função	**T**	**M**	**T**	**M**	**T**	**M**	**T**	**M**
e1	1,20	5,36	2,60	5,36	4,36	25,00	5,15	25,00
g1	1,56	5,65	2,88	5,65	4,48	25,00	6,01	25,00
h1	1,50	5,65	2,78	5,65	116,84	25,00	116,56	25,00
h2	1,22	5,37	2,00	5,37	116,03	25,00	116,43	25,00
i1	1,42	5,62	2,70	5,62	3,50	25,00	3,72	25,00
i2	1,28	5,62	1,86	5,62	4,09	25,00	4,97	25,00
clink	-	-	-	-	3,80	25,00	4,92	25,00
slink	-	-	-	-	3,69	25,00	4,97	25,00
upgma	-	-	-	-	2,51	25,00	5,22	25,00
wclink	-	-	-	-	2,55	25,00	5,40	25,00
wslink	-	-	-	-	3,98	25,00	5,40	25,00
wupgma	-	-	-	-	2,69	25,00	5,74	25,00

No *dataset Reuters* o algoritmo RBR foi o que obteve os melhores resultados, com a função *e1*. O algoritmo Bagglo encontrou clusters de qualidade semelhante aos do RBR mas, como possui custo computacional elevado, deve ser evitado. Com a função *i2*, o algoritmo RB foi cerca de 28% mais rápido, mas a pureza foi 4,8% menor e a entropia 3,7% maior. Assim, o algoritmo RBR, mesmo com um custo computacional maior, se saiu melhor neste *dataset*.

No *dataset* WAP o único algoritmo que superou 70% de pureza foi o RBR com as funções *e1* e *h1*. Outros resultados mais próximos foram RB (*e1*) com 69,9% e, com algoritmos aglomerativos, Agglo (*wupgma*) com 64,5% e Bagglo (*i2*) com 69,4%. Analisando-se os valores de tempo e memória consumidos pelas configurações citadas, o melhor resultado é atribuído ao RB (*e1*) que necessitou de menos da metade do tempo gasto pelo algoritmo RBR, consumindo a mesma quantidade de memória. Porém, como o processamento levou apenas 2,6 segundos, a escolha é pelo algoritmo RBR. Caso os tempos do RB e RBR fossem em minutos ou horas, a decisão seria pelo RB, apesar de uma pureza 1,6% menor.

Tabela 5.30: Melhores resultados de pureza e entropia médias nos *datasets* do CLUTO.

Dataset	**Algoritmo**	**Entropia**	**Pureza**	**T**	**M**
Classic	RB (*i1*)	0,393	0,787	1,69	7,28
FBIS	RBR (*h2*)	0,324	0,694	2,85	9,48
Reuters	RBR (*e1*)	0,358	0,678	0,56	2,46
WAP	RBR (*e1*)	0,316	0,715	2,60	5,36

Finalizando, os melhores resultados nestes experimentos são mostrados na Tabela 5.30. O algoritmo RBR alcançou os melhores resultados em 3 dos 4 *datasets*. Em dois deles com a função *e1*. O melhor resultado obtido pelo algoritmo RBR já era esperado, visto que ele tenta refinar o resultado da aplicação do algoritmo RB. Por este mesmo motivo o tempo computacional é superior.

Nos 4 *datasets* deste experimento a melhor solução não alcançou 80% de pureza média. Porém, de acordo com os experimentos realizados por Karypis (Karypis, 2002), com valores de k (clusters) diferentes do número de classes do *dataset*, foi possível encontrar melhores soluções (ver Tabela 5.31).

Tabela 5.31: Melhores resultados de pureza e entropia médias nos *datasets* do CLUTO, com valores de k (clusters) diferentes do número de classes.

Dataset	k	**Algoritmo**	**Entropia**	**Pureza**
Classic	5	RB (*e1*)	0,210	0,920
Reuters	20	RBR (*h2*)	0,310	0,700

Para o *dataset Classic*, bastou incrementar em uma unidade o número de clusters para que a pureza média da solução passasse de 78,7% para 92%. Já no caso do *dataset*

Reuters, a diferença só foi considerável ao se aumentar de 13 para 20 clusters. Ainda assim, a pureza melhorou em apenas 2,2%.

5.3 Discussão sobre os Experimentos

Nesta seção são comentados os resultados de todos os experimentos deste capítulo. Os algoritmos que conseguiram as melhores soluções são apresentados na Tabela 5.32, juntamente com seus respectivos valores de entropia e pureza médias, tempo computacional e memória. Nesta tabela, prioriza-se o custo do algoritmo (tempo e memória) dentre os que obtiveram melhor qualidade. A qualidade do resultado é escolhida apenas se o custo não for elevado, como no caso dos experimentos com os *dataset Reuters* e WAP. Levando-se em conta apenas o consumo de memória, o algoritmo *Spherical K-Means* é o mais indicado. Porém, as soluções encontradas por esta técnica são aleatórias e, em geral, de qualidade inferior à dos demais algoritmos.

Tabela 5.32: Melhores resultados de pureza e entropia médias em todos os experimentos.

Córpus/*Dataset*	**Algoritmo**	**Entropia**	**Pureza**	**T**	**M**
20 *Newsgroups*	RBR (*i2*)	0,114	0,946	7,15	45,00
WebKb	RB (*h1*)	0,149	0,904	3,06	33,00
LNAI/ *WebMiner*	RB (*i2*)	0,067	0,986	2,06	11,00
Classic	RB (*i1*)	0,393	0,787	1,69	7,28
FBIS	RBR (*h2*)	0,324	0,694	2,85	9,48
Reuters	RBR (*c1*)	0,358	0,678	0,56	2,46
WAP	RBR (*e1*)	0,316	0,715	2,60	5,36

De acordo com os resultados, os algoritmos particionadores foram a melhor opção em todos os experimentos. Como o algoritmo RBR tenta otimizar uma solução encontrada pelo RB, a tendência é que consuma mais tempo computacional e que encontre uma solução ainda melhor. Porém, pode acontecer de se manter a pureza média da solução gerada pelo RB, mas com uma entropia ligeiramente maior. Ou, no pior caso, na tentativa de uma otimização, o algoritmo acaba por reduzir o tamanho de um determinado cluster, causando um maior valor de entropia neste cluster e uma queda na pureza média da solução. Assim, priorizando o custo, decide-se que o algoritmo RB é a escolha ideal para a ferramenta FIP.

Sobre a função de otimização, a Tabela 5.32 não deixa claro qual seria a melhor escolha. Foram realizados experimentos com 7 *datasets*, sendo que as funções *e1*, *i2* e *h2* encontraram a melhor solução em 2 ocasiões cada, ficando a função *i1* com o melhor resultado em 1 *dataset*. Como a função *i2* foi a melhor no *dataset* LNAI/*WebMiner* e esteve próxima da melhor solução encontrada nos demais experimentos, a decisão é pela função *i2*.

Com relação aos mapas criados para organização e visualização dos córpus/*datasets* utilizados nos experimentos, pode-se dizer que nenhum deles atende as necessidades da FIP. A representação gráfica criada pelo gCLUTO fornece informações importantes sobre cada cluster, como grau de similaridade entre os documentos contidos no grupo e a distância entre os grupos. Porém, não permite acesso aos documentos. O mapa criado pela ferramenta *U-matrix* do Som_Pak provê uma boa representação da distribuição dos dados. Nos experimentos com os córpus WebKb e LNAI/*WebMiner* foi possível delimitar facilmente a região que cada classe ocupou no mapa. Já no experimento com o córpus 20 *Newsgroups* isso não foi possível, pois havia documentos de várias classes em uma mesma região.

Os problemas dessa ferramenta são: criar apenas um mapa (imagem), sem permitir qualquer interação com o usuário; possuir alto custo computacional; não fornecer informações sobre o conteúdo dos clusters como no gCLUTO. Já o mapa que a ferramenta GHSOM fornece é uma melhora em relação ao SOM por permitir uma interação/navegação e, ainda, organizar os documentos hierarquicamente. Porém, peca pela falta de uma melhor apresentação visual do mapa. Assim, seria interessante desenvolver uma ferramenta que criasse uma visualização conciliando as características do GHSOM e do gCLUTO.

Capítulo 6

Conclusões e Trabalhos Futuros

A popularização da *web*, juntamente com a redução dos preços dos dispositivos de armazenamento, facilitou e aumentou consideravelmente a distribuição *online* de documentos. Atualmente, é fácil disponibilizar instantaneamente todo tipo de documento digital para usuários de todo o planeta.

Porém, disponibilidade não significa, necessariamente, facilidade de acesso aos dados. Assim, várias técnicas têm sido implementadas e testadas para realizar uma filtragem mais eficiente dessas informações, facilitando o trabalho do usuário. Ferramentas que ajudam na busca, seleção e extração de informações específicas e relevantes de artigos científicos tornaram-se populares (Bollacker et al., 1998; Lawrence et al., 1999).

O projeto de uma Ferramenta Inteligente de apoio à Pesquisa (FIP), iniciado no LABIC/ICMC-USP, tem como objetivo disponibilizar uma ferramenta que auxilie um estudante, ou pesquisador, em uma determinada área por meio da construção de representação gráfica de uma área ou tópico de pesquisa escolhido. Tal mapa deve permitir a visualização das áreas principais, subáreas, artigos e autores, além de outros aspectos importantes relacionados com o assunto/tópico de pesquisa. A informação de entrada para a criação do mapa é um conjunto de artigos recuperados da *web*, relacionados com o assunto e com o conjunto de palavras-chave escolhidos pelo usuário. Atualmente, três módulos da FIP estão em desenvolvimento. Esses módulos estão relacionados com as etapas de recuperação dos artigos da *web*, análise dos artigos (determinação da simila-

ridade entre artigos) e visualização (construção do mapa). Para construir tal mapa, a similaridade entre os artigos deve ser calculada (baseada em palavras-chave, n-gramas, etc). Além disso, pode-se esperar que artigos similares possuam referências em comum.

Neste trabalho de mestrado foram investigadas técnicas de *clustering* conhecidas da literatura para se decidir por uma delas que fosse capaz de agrupar os documentos da FIP, de maneira eficiente e eficaz, sem a necessidade da intervenção do usuário. Foram realizados testes com 2 córpus e 4 *datasets* comuns da literatura (20 *Newsgroups* e WebKb; *Classic*, FBIS, *Reuters* e WAP), além do córpus construído para uso e teste na própria FIP (LNAI/ *WebMiner*).

No capítulo 2 foi comentado que nenhuma das técnicas disponíveis atualmente é capaz de obter a melhor solução em qualquer conjunto de dados. Pelos resultados apresentados no capítulo 5, pode-se verificar que nenhuma das técnicas testadas obteve o melhor resultado em todos os experimentos, apesar de a escolha ter sido pelo algoritmo particionador RB, implementado no *framework* CLUTO. A função de critério de otimização e o peso atribuído aos atributos também interferem no resultado. Ainda, foi verificado que o número de clusters pode ter grande influência na solução. Uma solução pode ser considerada de baixa qualidade com um número k de clusters e de ótima qualidade com um número $k + 1$ de clusters, ou vice-versa.

Com relação ao córpus de artigos científicos (LNAI/ *WebMiner*) foram realizados experimentos usando duas representações para as referências bibliográficas: uma faz o pré-processamento das mesmas em modo texto, ou seja, do modo comumente usado em *clustering* de documentos gerando uma *bag-of-words* do resumo com as referências; a outra faz um pré-processamento para identificar individualmente cada uma das referências e, sabendo-se qual delas é citada em cada artigo, usa essa informação para se construir uma *bag-of-references*.

De acordo com os resultados, acreditamos ser interessante analisar um mesmo córpus de diferentes pontos de vista. Trabalhando-se com documentos científicos, duas visões apresentam bons resultados: a *bag-of-words* do resumo e texto das referências e a *bag-of-references*. A primeira configuração obteve melhor pureza no agrupamento dos documentos de acordo com as classes pré-definidas para o experimento. Já a segunda, pode informar quando uma determinada referência é citada em uma área distinta, indicando a repercussão de um trabalho. Esta característica é útil, também, quando se

desejar encontram trabalhos similares que citem referências em comum. É importante lembrar que o resultado dessa segunda visão foi melhor utilizando-se o peso binário ao invés do *tf-idf*.

Uma característica importante do *clustering* é que, por ser um algoritmo descritivo, ele organiza os dados de acordo com a métrica definida pelo usuário e tenta encontrar relações de semelhança entre esses dados. Entretanto, para um problema real, em que não se conhece as classes dos documentos *a priori*, não se pode afirmar que os clusters encontrados seriam os melhores. Por esse motivo, após o *clustering* dos documentos o usuário deve avaliar o resultado subjetivamente e, se necessário, modificar os parâmetros do algoritmo de modo a se conseguir um melhor resultado.

É importante lembrar que a alta taxa de acerto nos resultados nos experimentos realizados não será obtida em qualquer córpus. O número de classes era conhecido e os córpus continham documentos em classes bem definidas, na maioria dos casos. Numa situação no mundo real, costuma-se tratar com um número elevado de classes que, geralmente, é desconhecido pelo usuário. Ainda, pode-se ter documentos que pertençam a classes distintas, como no caso do córpus 20 *Newsgroups*, do dataset *Reuters* e mesmo no LNAI/*WebMiner*, no qual os artigos foram rotulados como sendo de uma classe, simplesmente em função do simpósio, mas podem existir artigos que tratam de abordagens híbridas como RBC e PLI ou RBC aplicado a RI, etc.

6.1 Contribuições do Trabalho

Além das análises de diversos algoritmos de *clustering* apresentadas, outras contribuições deste trabalho de Mestrado são apresentadas a seguir.

- Para se criar a *bag-of-references* foi implementada uma abordagem para extrair e estruturar referências bibliográficas de artigos científicos, permitindo a identificação das referências duplicadas.

  A técnica implementada usa um *parser* baseado em regras para extrair as informações das referências, constrói uma chave (*string*) com essas informações para cada referência e as armazena em uma estrutura de dados métrica chamada *Slim-Tree*.

A estrutura *Slim-Tree* permite a minimização da comparação entre referências, considerando apenas as k chaves mais similares a uma dada chave, alcançando 97% de acurácia no processo de identificação;

Até o término deste trabalho foi encontrada apenas uma publicação em que o conceito usado aqui para *bag-of-references* é mencionado (Erosheva et al., 2004). Entretanto, como é um trabalho recente, em classificação, não tínhamos conhecimento dele quando desenvolvemos o mesmo conceito.

- A saída da implementação que cria a *bag-of-references* é compatível com a ferramenta PreText. Existem *scripts* de conversão do formato adotado pelo PreText para *frameworks* como o *Weka* e as implementações do *Discover*. Assim, a *bag-of-references* pode ser investigada por outras técnicas como classificação ou regras de associação, por exemplo;

- Para teste das ferramentas de *clustering*, foram implementados diversos *scripts* para converter os arquivos de saída gerados pela ferramenta PreText para o formato correto para uso nas ferramentas.

Os resultados obtidos neste trabalho contribuíram nas publicações a seguir.

- Melo, V., M. Secato & A. A. Lopes (2003). Extração e identificação automáticas de informações bibliográficas de artigos científicos. In Jornadas Chilenas de Computacion - IV Workshop on Advances and Trends in AI for Problem Solving (ATAI), Chillán - Chile, pp. 1 - 7.

- Melo, V. V. & A. A. Lopes (2004a). Identificação eficiente de referências bibliográficas duplicadas em um corpora de artigos científicos. In *Workshop de Teses e Dissertações em IA (WTDIA)*, São Luís - Maranhão, pp. 71 - 80.

- Melo, V. V. & A. A. Lopes (2004b). Usando as referências bibliográficas no clustering de artigos científicos. In *Jornadas Chilenas de Computacion - Workshop on Artificial Intelligence (WAI)*, Arica - Chile, pp. 1 - 7.

- Melo, V. V. & A. A. Lopes (2005). Efficient identification of duplicate bibliographical references. In *Logic Applied to Technology (LAPTEC)*, Himeiji, Japan, pp. 1 - 8.

- Artigo a ser submetido: Lopes, A. A., R. Minghim and V. V. Melo. *Creating Interactive Document Maps Through Dimensionality Reduction and Visualization Techniques.*

6.2 Sugestões de Trabalhos Futuros

Neste trabalho foram utilizadas, basicamente, técnicas de *clustering* clássicas: aglomerativas hierárquicas, particionadoras e redes neurais. Porém, existe ainda uma grande quantidade de algoritmos que podem ser utilizadas para agrupamento de documentos. Talvez algum deles seja capaz de trabalhar melhor com as referências bibliográficas do que os algoritmos testados neste trabalho.

Uma outra técnica que pode ser melhor estudada é a de redução de dimensão. Como o tempo computacional consumido por estas técnicas é elevado, optamos pelo modo mais simples de redução: cortes de Luhn.

A representação gráfica gerada pela ferramenta gCLUTO apresenta apenas as informações do cluster por completo, ou seja, não se tem informação, por exemplo, sobre um determinado documento. Já o mapa gerado pelo GHSOM permite visualização dos documentos e navegação pela estrutura hierárquica dos mapas. Assim, seria interessante desenvolver uma técnica que pudesse unir as capacidades destas duas ferramentas e fosse capaz de informar, por exemplo, a importância de um documento no cluster, a proximidade entre documentos - não apenas entre clusters, o grau de similaridade entre pares de documentos, a posição dos documentos dentro dos clusters, entre outras informações.

Com relação à *bag-of-references*, utilizamos, neste trabalho, dois tipos de medida: binário, para informar a presença ou não de uma referência em um documento; e *tf-idf*, levando em consideração a quantidade de textos em que um determinado documento é referenciado. Como trabalho futuro, seria interessante utilizar a medida *tf-idf* para a *bag-of-references* do mesmo modo que é utilizada para a *bag-of-words*, sendo que, nesse caso, seria levada em consideração a freqüência com que cada referência é citada no corpo do documento. Para tal, é necessário implementar um *parser* capaz de reconhecer vários tipos de citação e relacioná-los com a referência bibliográfica correspondente.

Referências

Al-Sultan, K. (1995). A tabu search approach to the clustering problem. *Pattern Recognition 9*(28), 1443–1451.

Alahakoon, D., S. K. Halgamuge, & B. B. Srinivasan (1998). Self growing cluster development approach to data mining. *IEEE International Conference on Systems, Man, and Cybernetics 3*, 2901–2906.

Ankerst, M., M. M. Breunig, H.-P. Kriegel, & J. Sander (1999). Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (ICMD)*, Philadephia, Pennsylvania, pp. 49–60.

Babu, G. P. & M. N. Murty (1993). A near-optimal initial seed value selection in k-means algorithm using a genetic algorithm. *Pattern Recognition Letters 14*(10), 763–769.

Berendt, B., A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, & G. Stumme (Eds.) (2004). *Web Mining: From Web to Semantic Web, 1st European Web Mining Forum, EMWF 2003, Cavtat-Dubrovnik, Croatia, September 22, 2003, Revised Selected and Invited Papers*, Volume 3209 of *Lecture Notes in Computer Science*. Springer.

Berkhin, P. (2002). Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA.

Berry, M. W., S. T. Dumais, & G. W. O'Brien (1994). Using linear algebra for

intelligent information retrieval. In *SIAM Review*, Volume 37, pp. 573–595.

Beyer, K., J. Goldstein, R. Ramakrishnan, & U. Shaft (1999). When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science 1540*, 217 235.

Bickel, S. & T. Scheffer (2004). Multi-view clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pp. 19 26.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc.

Bloedorn, E., I. Mani, & T. R. MacMillan (1996). Representational issues in machine learning of user profiles. In *AAAI Spring Symposium on Machine Learning in Information Access, 1996.*, Stanford.

Boley, D., V. Borst, & M. Gini (1999). An unsupervised clustering tool for unstructured data. In *International Joint Conference on Artificial Intelligence IJCAI*, Stockholm, pp. 297 301. ACM Press.

Bollacker, K., S. Lawrence, & C. L. Giles (1998). CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In K. P. Sycara & M. Wooldridge (Eds.), *Proceedings of the 2nd International Conference on Autonomous Agents*, New York, pp. 116 123. ACM Press.

Brasil, C. S. & A. A. Lopes (2004). Mineração de artigos científicos usando aprendizado de máquina. In *Jornadas Chilenas de Computacion - Workshop on Artificial Intelligence (WAI)*, Arica - Chile, pp. 1 7.

Buckley, C. & A. F. Lewit (1985). Optimizations of inverted vector searches. In *Proceedings of the 8th International Conference on Research and Development in Information Retrieval (SIGIR)*, New York, pp. 97 110.

Cheeseman, P. & J. Stutz (1996). Bayesian classification (autoclass): Theory and results. In P. S. U. M. Fayyad, G. Diatetsky-Shapiro & R. U. (Eds.) (Eds.), *Advances in Knowledge Discovery and Data Mining*, Chapter 6, pp. 153 180. AAAI Press The MIT Press.

Cheng, K., J. Lin, & C. Mao (1996). The application of competitive hopfield neural

network to medical image segmentation. *IEEE Transactions on Medical Imaging 15*(4), 560–567.

Cutting, D. R., J. O. Pedersen, D. Karger, & J. W. Tukey (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th ACM SIGIR Annual International Conference on Research and Development in Information Retrieval (ICRDIR)*, pp. 318–329.

Dasgupta, S., M. Littman, & D. McAllester (2001). Pac generalization bounds for co-training. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 375–382. MIT Press.

Day, W. H. E. (1992). Complexity theory: an introduction for practitioners of classification. In P. Arabie & L. Hubert (Eds.), *Clustering and Classification*. River Edge, NJ: World Scientific Publishing Co.

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.

Dempster, A. P., N. M. Laird, & D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society 39*, 1–38.

Dhillon, I. S. & D. S. Modha (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning 42*(1), 143–175.

Dittenbach, M., D. Merkl, & A. Rauber (2000). Using growing hierarchical self-organizing maps for document classification. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, pp. 7–12. D-Facto Publications.

Drineas, Frieze, Kannan, Vempala, & Vinay (1999). Clustering in large graphs and matrices. In *Proceedings of the 10th ACM SIAM Symposium on Discrete Algorithms (SODA)*, Philadelphia, PA, USA, pp. 291–299. Society for Industrial and Applied Mathematics.

Erosheva, E., S. Fienberg, & J. Lafferty (2004). Mixed membership models of scientific publications. In *Proceedings of the National Academy of Sciences*, Volume 101 - Suppl 1, USA, pp. 5220 5227.

Ertoz, L., M. Steinbach, & V. Kumar (2003). Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *SIAM International Conference on Data Mining*, Volume 3, San Francisco, CA.

Ester, M., H.-P. Kriegel, J. Sander, & X. Xu (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, & U. Fayyad (Eds.), *2nd International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, pp. 226 231. AAAI Press.

Faloutsos, C. & K.-I. Lin (1995, 22-25). FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In M. J. Carey & D. A. Schneider (Eds), *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (ICMD)*, San Jose, California, pp. 163 174.

Feldman, R., Y. Aumann, M. Fresko, O. Liphstat, B. Rosenfeld, & Y. Schler (1999). Text mining via information extraction. In J. M. Zytkow & J. Rauch (Eds.), *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery (PKDD-99)*, Volume 1704 of *LNAI*, Prague, Czech Republic, pp. 165 173. Springer.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning 2*, 139 172.

Fodra, R. & A. Lopes (2001). Analise da semantica latente na determinaçao de similaridade entre artigos científicos e visualização de córpus. In *Workshop de Teses e Dissertações em IA (WTDIA)*, São Luís - Maranhão.

Foltz, P. W. & S. T. Dumais (1992, December). Personalized information delivery: An analysis of information-filtering methods. *Communications of the ACM 35*(12), 51 60.

Frakes, W. B. & R. Baeza-Yates (1992). *Information Retrieval Data Structures & Algorithms*. Englewood Cliffs, NJ: Prentice Hall.

Fraley, C. & A. Raftery (1999). Mclust: Software for model-based cluster and discriminant analysis. Technical Report 342, Department of Statistics, University of Washington.

Fritzke, B. (1995). A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), *Advances in Neural Information Processing Systems 7*, pp. 625–632. Cambridge MA: MIT Press.

Ganti, V., J. Gehrke, & R. Ramakrishnan (1999). CACTUS - clustering categorical data using summaries. In *Knowledge Discovery and Data Mining*, pp. 73–83.

Geng, J. & J. Yang (2003). Automatic extraction and integration of bibliographic information on the web. Technical report, Department of Computer Science, Duke University.

Gosh, J. (2003). *Scalable Clustering Methods for Data Mining. Handbook of Data Mining*, Chapter 10, pp. 247–277. Lawrence Ealbaum Assoc.

Grefenstette, G. (1994). *Explorations in automatic thesaurus discovery*. Boston: Kleuwer.

Guha, S., R. Rastogi, & K. Shim (1998). CURE: an efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 73–84.

Guha, S., R. Rastogi, & K. Shim (1999). Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Australia*, pp. 512–521. IEEE Computer Society.

Han, J. & M. Kamber (2000). *Data Mining: Concepts and Techniques*. San Francisco, CA: Morgan Kaufmann.

Han, J., M. Kamber, & A. K. H. Tung, (2001). Spatial clustering methods in data mining: A survey. In H. Miller & J. Han (Eds.), *Geographic Data Mining and*

Knowledge Discovery. Taylor and Francis.

Hartigan, J. A. & M. A. Wong (1979). A K-means clustering algorithm. *Applied Statistics 28*, 100–108.

Haykin, S. (1999). *Neural Networks*. NJ: Prentice Hall.

Hinneburg, A. & D. A. Keim (1998). An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 58–65. AAAI Press.

Honkela, T., S. Kaski, K. Lagus, & T. Kohonen (1996). Newsgroup exploration with WEBSOM method and browsing interface. Technical Report A32. Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland.

Huang, S., M. Ward, & E. Rundensteiner (2003). Exploration of dimensionality reduction for text visualization. Technical Report TR-03-11, Worcester Polytechnic Institute, Computer Science Department.

Hylton, J. A. (1996). Identifying and merging related bibliographic records. Dissertação de Mestrado, MIT.

Jesan, J. P. (2004). The neural approach to pattern recognition. *Ubiquity 5*(7), 2–2.

Karypis, G. (2002). CLUTO a clustering toolkit. Technical Report 02-017, Department of Computer Science, University of Minnesota. Available at http://www.cs.umn.edu/~cluto.

Kaski, S., T. Honkela, K. Lagus, & T. Kohonen (1996). Creating an order in digital libraries with self-organizing maps. In *Proceedings of the 1996 World Congress on Neural Networks (WCNN)*, pp. 814–817. Mahwah, NJ: Lawrence Erlbaum and INNS Press.

Kaufman, L. & P. J. Rousseeuw (1990). Finding groups in data: an introduction to cluster analysis. In *Wiley series in Probability and Mathematical Statistics*.

King, B. (1967). Step-wise clustering procedures. *Journal of the American Statistical Association* (69), 86–101.

Kohonen, T. (1990). Self-organized formation of topologically correct feature maps. In J. W. Shavlik & T. G. Dietterich (Eds.), *Readings in Machine Learning*, pp. 326–336. San Mateo, CA: Kaufmann.

Kohonen, T. (2001). *Self-Organizing Maps* (3rd ed.), Volume 30 of *Information Sciences*. Springer Verlag.

Kohonen, T., J. Hynninen, J. Kangas, & J. Laaksonen (1996). Sompak: the self-organizing map program package. Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo.

Koller, D. & M. Sahami (1997). Hierarchically classifying documents using very few words. In D. H. Fisher (Ed.), *Proceedings of the 14th International Conference on Machine Learning (ICML)*, Nashville, US, pp. 170–178. Morgan Kaufmann Publishers, San Francisco, US.

Kowalski, G. (1997). *Information Retrieval Systems: Theory and Implementation*. Kluwer Academic Publishers.

Kurita, T. (1991). An efficient agglomerative clustering algorithm using a heap. *Pattern Recognition 24*(3), 205–209.

Larsen, B. & C. Aone (1999). Fast and effective text mining using linear-time document clustering. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, pp. 16–22. ACM Press.

Lawrence, S., C. L. Giles, & K. Bollacker (1999). Digital libraries and autonomous citation indexing. *IEEE Computer 32*(6), 67–71.

Levenshtein, V. I. (1966). Binary codes capable of correcting insertions and reversals. *Soviet Physics Doklady 10*(8), 707–710.

Liao, T. W., Z. Zhang, & C. Mount (1998). Similarity measures for retrieval in case based reasoning systems. *Applied Artificial Intelligence 12*(4), 267–288.

Lin, X., D. Soergel, & G. Marchionini (1991). A self-organizing semantic map for information retrieval. In *Proceedings of the 14th ACM SIGIR Annual International Conference on Research and Development in Information Retrieval (ICRDIR)*, pp. 262–269. ACM Press.

Lopes, A. (1995). Raciocínio baseado em casos. Dissertação de Mestrado, ICMC-USP.

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

Macqueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Symposium on Math, Statistics, and Probability*, Berkeley, CA, pp. 281 297. University of California Press.

Martinetz, T. & K. Schulten (1991). Topology representing networks. *Neural Network 7*(3), 507 522.

Matsubara, E. T., C. A. Martins, & M. C. Monard (2003). Pretext: Uma ferramenta para pré-processamento de textos utilizando a abordagem bag-of-words. Technical Report 209, ICMC-USP, São Carlos - SP.

Megiddo, N. & K. J. Supowit (1984). On the complexity of some common geometric location problems. *SIAM Journal of Computing 13*(1), 182 196.

Melo, V., M. Secato, & A. Lopes (2003). Extração e identificação automáticas de informações bibliográficas de artigos científicos. In *Jornadas Chilenas de Computacion - IV Workshop on Advances and Trends in AI for Problem Solving (ATAI)*, Chillán - Chile, pp. 1 7.

Melo, V. V. & A. A. Lopes (2004a). Identificação eficiente de referências bibliográficas duplicadas em um corpora de artigos científicos. In *Workshop de Teses e Dissertaçoes em IA (WTDIA)*, São Luís - Maranhão, pp. 71 80.

Melo, V. V. & A. A. Lopes (2004b). Usando as referências bibliográficas no clustering de artigos científicos. In *Jornadas Chilenas de Computacion - Workshop on Artificial Intelligence (WAI)*, Arica - Chile, pp. 1 7.

Melo, V. V. & A. A. Lopes (2005). Efficient identification of duplicate bibliographical references. In *Logic Applied to Technology (LAPTEC)*, Himeiji, Japan, pp. 1–8.

Milligan, G. W. & L. M. Sokol (1980). A two-stage clustering algorithm with robust recovery characteristics. *Educational and Psychological Measurement* (40), 755–759.

Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill Series in Computer Science.

Monge, A. E. & C. Elkan (1997). An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Research Issues on Data Mining and Knowledge Discovery*, pp. 0–.

Moukas, A. & G. Zacharia (1997). Evolving a multi-agent information filtering solution in amalthaea. In *Proceedings of the 1st International Conference on Autonomous Agents (Agents)*, pp. 394–403. ACM Press.

Murtagh, F. (1984). A survey of recent advances in hierarchical clustering algorithms which uses cluster centers. *Computer Journal* (26), 354–359.

Needleman, S. B. & C. D. Wunsch (1970). A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology 48*, 443–453.

Rauber, A., D. Merkl, & M. Dittenbach (2002, November). The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks 13*(6), 1331–1341.

Ritter, H. & T. Kohonen (1989). Self-organizing semantic maps. *Biological Cybernetics 61*, 241–254.

Rogers, S. K., J. M. Colombi, C. E. Martin, J. C. Gainey, K. H. Fielding, T. J. Burns, D. W. Ruck, M. Kabrisky, & M. Oxley (1995). Neural networks for automatic target recognition. *Neural Network 8*(7-8), 1153–1184.

Salton, G. & M. J. McGill (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc.

Sarafis, I., A. Zalzala, & P. Trinder (2002). A genetic rule-based data clustering toolkit. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pp. 1238 1243.

Schikuta, E. & M. Erhart (1997). The bang-clustering system: Grid-based data analysis. In *Proceedings of the 2nd International Symposium on Advances in Intelligent Data Analysis, Reasoning about Data (IDA)*, pp. 513–524. Springer-Verlag.

Scholtes., J. C. (1993). *Neural Networks in Natural Language Processing and Information Retrieval*. Tese de Doutorado, Amsterdam, Netherlands.

Secato, M. V. & A. A. Lopes (2001). Extração de informações de artigos científicos. In *SHCUSP*.

Sheikholeslami, G., S. Chatterjee, & A. Zhang (1998). WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB)*, pp. 428 439.

Sibson, R. (1973). Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal 16*(1), 30 34.

Smith, T. F. & M. S. Waterman (1981). Identification of common molecular subsequences. *Journal of Molecular Biology 147*, 195–197.

Sànchez-Marrè, M., U. Cortés, I. R-Roda, & M. Poch (1998). L'eixample distance: a new similarity measure for case retrieval. In *Proceedings of the 1st Catalan Conference on Artificial Intelligence (CCIA)*, Volume 15, pp. 246 253. ACIA Bulletin

Sneath, P. H. & R. R. Sokal (1973). *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. San Francisco: W.H. Freeman.

Steinbach, M., G. Karypis, & V. Kumar (2000). A comparison of document clustering techniques. In *KDD Workshop on Text Mining*.

Strehl, A., J. Ghosh, & R. Mooney (2000, July). Impact of similarity measures on

web-page clustering. In *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000), 30-31 July 2000, Austin, Texas, USA*, pp. 58–64. AAAI.

Torgerson, W. S. (1958). *Theory and Methods of Scaling*. New York: Wiley.

Tung, A. K. H., R. T. Ng, L. V. S. Lakshmanan, & J. Han (2001). Constraint-based clustering in large databases. In *Proceedings of the 8th International Conference on Database Theory*, pp. 405–419. Springer-Verlag.

van Rijsbergen, C. J. (1979). *Information Retrieval* (2 ed.). London: Butterworth.

Visa, A. (2001). Technology of text mining. In *Proceedings of the 2nd International Workshop on Machine Learning and Data Mining in Pattern Recognition (MLDM)*, Volume 2123 of *Lecture Notes in Artificial Intelligence*, pp. 1–11. Springer.

Voorhees, E. M. (1986). Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing and Management 22*(6), 465–476.

Wang, W., J. Yang, & R. R. Muntz (1997). STING: A statistical information grid approach to spatial data mining. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, & M. A. Jeusfeld (Eds.), *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB)*, Athens, Greece, pp. 186–195. Morgan Kaufmann.

Ward, J. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association 58*, 236–244.

Wilson, D. R. & T. R. Martinez (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research (JAIR) 6*, 1–34.

Yan, T. W. & H. Garcia-Molina (1994). Index structures for information filtering under the vector space model. In *Proceedings of the 10th International Conference on Data Engineering*, pp. 337–347. IEEE Computer Society.

Yao, K. C., M. Mignotte, C. Collet, P. Galerne, & G. Burel (2000). Unsupervised

segmentation using a self-organizing map and a noise model estimation in sonar imagery. *Pattern Recognition 33*(9), 1575–1584.

Zamir, O. & O. Etzioni (1998). Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pp. 46–54.

Zhang, Q. P., M. Liang, & W. C. Sun (2004). Multi-resolution image data fusion using 2-d discrete wavelet transform and self-organizing neural networks. In *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry (VRCAI)*, New York, NY, USA, pp. 297–301. ACM Press.

Zhao, Y. & G. Karypis (2002). Criterion functions for document clustering: Experiments and analysis. Technical Report 01-40, University of Minnesota, Department of Computer Science, Minneapolis.

Zipf, G. K. (1949). *Human Behavior and the Principle of Least-Effort*. Cambridge, MA: Addison-Wesley.