REDES NEURONAIS ART NA MINERAÇÃO DA WEB

Newton Graça Reis de Carvalho

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Luis Alfredo Vidal de Carvalho, D.Sc.

Prof. Geraldo Bonorino Xexeo D.Sc.

Prof. Rosa Maria Esteves Moreira da Costa, D.Sc.

Prof. Alexandre Gonçalves Evsukoff, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

FEVEREIRO DE 2005

CARVALHO, NEWTON GRAÇA REIS DE

Redes Neuronais ART na mineração da WEB

[Rio de Janeiro] 2005

VII, 112 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2005)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Webmining
2. Datamining
3. Análise de Agrupamentos
4. Inteligência Artificial
5. Redes Neuronais ART

I. COPPE/UFRJ II. Título (série)

Dedico esta menção a todos que de certa forma contribuíram para a conclusão deste trabalho. Os nomes são incontáveis. Infelizmente, por meio destas parcas linhas eu não conseguiria expressar todo o meu carinho.

Agradecimentos

Agradeço primeiramente a D'us, por tantos mimos que me proporcionou, aos meus pais, minha família, amigos e todas as pessoas que de certa forma emanaram bons pensamentos para que eu conseguisse reunir forças suficientes para finalizar este trabalho.

Ao meu orientador Luis Alfredo, que teve muita paciência comigo ao longo desses dois anos.

Ao profº Sérgio Exel, que em muito contribui para a realização deste trabalho. Infelizmente, eu não pude retribuir à altura a atenção que me foi dispensada.

Aos meus amigos José Afonso, Paula, Lú e Luciana, que sempre se mostraram solícitos e disposto a ajudar durantes os difíceis dias de adaptação a esta terra.

Aos meus colegas de república, Matheus, Gustavo e Rodrigo. Sou muito agradecido pela oportunidade de conhece-los e ter convivido com vocês durante esses anos.

Agradeço imensamente à minha tia Rizete e às minhas primas Berta e Tâmara. Vosso carinho foi de fundamental importância para que eu não desistisse durante os terríveis e sombrios dias de tristeza que eu enfrentei assim que cheguei aqui.

Não posso me esquecer do Beto e do Guto. Nós éramos o famoso trio PMG, bons tempos, bons tempos.

Á Ana Paula, pela dedicação dispensada na revisão dos capítulos desta tese.

Enfim, não sei o que acontecerá no futuro, mas levarei sempre vocês em meu coração.

Obrigado, e que D'us vos abençoe.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

REDES NEURONAIS ART NA MINERAÇÃO DA WEB

Newton Graça Reis de Carvalho

Fevereiro/2005

Orientador: Luis Alfredo Vidal de Carvalho

Programa: Engenharia de Sistemas e Computação

Há uma quantidade enorme e continuamente crescente de textos eletrônicos disponíveis na *Web*, aos quais podem conter conhecimento humano ou mesmo acerca de organizações. Um importante esforço de pesquisa diz respeito ao estudo e desenvolvimento de melhores maneiras para acessar este conhecimento. Dentro deste contexto, a Mineração de Dados da Web – por meio da construção de agrupamentos – constitui uma abordagem que permite organizar coleções de documentos por tópicos de forma que ajude os usuários a encontrar as informações necessárias. As Redes Neuronais ART possuem propriedades que as tornam aplicáveis à construção de agrupamentos de documentos hipertexto. Neste trabalho, serão apresentados resultados experimentais obtidos com uma rede *FUZZY ART*.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

WEB MINING USING ADAPTIVE RESONANCE THEORY

Newton Graça Reis de Carvalho

February/2005

Advisor: Luis Alfredo Vidal de Carvalho

Department: System and Computing Engineering

There is a large and continually growing quantity of electronic text available, which contain essential human and organization knowledge. An important research endeavor is to study and develop better ways to access this knowledge. In this context, Web Mining – using Cluster Analysis - constitutes an approach to automatically organize textual document collections by topics to help users find the information they need. Adaptive Resonance Theory (ART) neural networks possess several interesting properties that make them applicable to Cluster Analysis task. This work presents experimental results obtained using an FUZZY-ART Neural Network.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1 - Introdução

1.1 Motivação

Devido ao crescimento contínuo do volume de dados eletrônicos disponíveis na *Web*, técnicas de mineração de dados tornam-se cada vez mais necessárias para gerenciar e organizar essa enorme quantidade de informações. Como a mineração de dados tradicional foi desenvolvida para lidar com dados estruturados, a mineração de dados da *Web* surge como uma variação criada com o propósito de lidar com dados semi-estruturados, constituídos pelos documentos de hipertexto.

Apesar de todo avanço empregado na melhoria dos serviços de busca, e do desenvolvimento de algoritmos de *PageRank* mais eficientes, ainda nos dias atuais, os usuários acessam o conteúdo disponível da *Web* através de duas formas: clicando em *links* ou executando consultas a partir de serviços de busca, utilizando palavras chaves. Este tipo de processo tende a ser insatisfatório.

Neste contexto, a análise de agrupamentos surge como uma forma alternativa e atraente para organizar os documentos presentes na *Web* em uma estrutura lógica, de forma a proporcionar uma acessibilidade intuitiva aos usuários da grande rede.

Em razão da sua aplicabilidade, ela tem sido estudada e empregada de forma intensiva em áreas como: *information retrieval* e mineração de textos.

Ainda no contexto da *Web*, o estado de arte a ser alcançado pela análise de agrupamentos corresponde à geração automática de agrupamentos de documentos organizados de forma hierárquica, semelhante à organização de assuntos adotadas por diretórios como o YAHOO!

CUTTING[32] propõe o uso da análise de agrupamentos para navegar através de coleções de documentos, enquanto ZAMIR *et al* [63] a utiliza para organizar resultados obtidos por meio de uma consulta submetida a um serviço de busca.

Em se tratando do emprego das redes neuronais para este fim, existem aplicações utilizando o tradicional mapa de Kohonen, existindo poucas publicações tratando sobre a aplicabilidade das redes neuronais *ART* para a construção de agrupamentos de documentos [64].

O objetivo deste trabalho é a fazer a construção de agrupamentos de documentos de hipertexto usando uma rede neuronal *FUZZY-ART*.

Para tal, a parte inicial deste trabalho implica num minucioso processo preparação de dados, o que envolve a execução de alguns passos: montagem de um conjunto palavras vazias, construção de um *parser*, implementação de um algoritmo de extração de raízes e a composição de um dicionário.

Entretanto, não são raras as ocasiões em que os documentos sejam representados por uma quantidade excessiva de termos, o que implica num custo computacional elevado. Para otimizar essa quantidade de termos, é freqüente a utilização de heurísticas para seleção de palavras. Estas heurísticas tem como objetivo reduzir aa representação destes arquivos.

A partir do conjunto de termos resultantes os documentos podem, então, ser convertidos para uma forma mais estruturada de dados, neste caso, vetores de freqüências.

Uma vez obtidos os vetores de freqüências em mãos, a rede neuronal FUZZY-ART é executada, resultando numa saída que corresponda aos agrupamentos encontrados.

Por sua vez, interpretar esses agrupamentos representa outro problema à parte, uma vez que analisa-los não representa uma tarefa intuitiva. Por essa razão, neste trabalho, é proposta uma ferramenta que possibilite a visualização dos agrupamentos de documentos encontrados.

1.2 LINHAS GERAIS DA TESE

Para desenvolver este trabalho foi necessária a compreensão sobre as técnicas de análise de agrupamentos e extração de propriedade de hipertextos. Este capítulo apresenta a motivação e a organização deste trabalho.

Por sua vez, o Capitulo 2 abrange um estudo acerca do desenvolvimento da *Internet* ao longo da história, suas ferramentas e tecnologias empregadas. Neste contexto estão incluso os serviços de buscas e a Mineração de Dados da *Web* (*Web Mining*).

No Capítulo 3 é apresentada uma visão geral a respeito das técnicas de análise de agrupamentos existentes, enquanto o Capítulo 4 apresenta um estudo sobre as redes neuronais *ART*, enfocando o processo de aprendizado competitivo, os algoritmos *ART*-1 e *FUZZY-ART*.

O Capítulo 5 apresenta os módulos que descrevem o processo de mineração propriamente dita, e a visualização dos resultados produzidos. No Capítulo 6 seguem as conclusões deste trabalho e sugestões de trabalhos futuros.

Capitulo 2 – A *Internet*, características, ferramentas de recuperação e descoberta de informação.

Neste capítulo serão apresentados dados históricos sobre o desenvolvimento da Internet ao longo do tempo, o documentos de hipertexto, os mecanismos de recuperação de informação e a mineração de dados aplicada no contexto da Web.

2.1 INTRODUÇÃO

A internet é definida como um conjunto de diversas redes de computadores conectadas umas as outras em todo mundo. Estas redes divergem quanto à natureza de seus dados, podendo ser redes universitárias, comerciais, militares e científicas. As redes eletrônicas de computadores proporcionam a seus usuários comunicação a baixo custo e acesso a fontes inesgotáveis de informação. Elas interconectam pessoas para os mais diversos fins e têm contribuído para ampliar e democratizar o acesso à informação, eliminando barreiras como distância, fronteira e fuso horário [1].

Além dos recursos básicos de correio eletrônico e listas de discussão, a Internet proporciona serviços diversificados como: bases de dados especializadas, catálogos de bibliotecas, repositórios de software de domínio público, jornais e revistas eletrônicas. Também é possível ter acesso a recursos de hardware especializados como computadores de alto desempenho e processadores especializados.

Na atualidade, o número estimado de usuários é de 40 milhões em mais de 140 países, considerando que 10 milhões realizam acesso diário. Esta significativa massa acessa

a rede através de computadores e terminais em instituições educacionais, provedores comerciais e outras organizações.

2.2 HISTÓRICO

A Internet surgiu de uma pequena rede experimental de computadores criada em 1969 pela *Advanced Research Projects Agency* (ARPA), agencia ligada ao Departamento de Defesa dos EUA. Seu inicio veio com a idéia de permitir a partilha de recursos computacionais como: bancos de dados, computadores de alto desempenho e dispositivos gráficos, entre os pesquisadores e fornecedores contratados pelo Departamento. Logo passou a ser usada também como meio de cooperação entre os participantes do projeto, possibilitando o uso de correio eletrônico, dentre outros serviços [1].

No ano de 1980, houve uma divisão dessa rede, sendo uma parte designada para uso militar e civil, a *Arpanet*, e a outra direcionada para uso militar, a *Milnet*. A integração existente entre ambas foi chamada de *Defense Advanced Research Projects Agency Internetwork*. Posteriormente, essa nomenclatura foi simplificada para *Internet*.

Em meados de 1989 ocorreu o surgimento da *World Wide Web* (*Web* ou *WWW*), representando uma evolução da própria Internet, que a partir de então passa adotar como padrões o emprego do protocolo *http* (*Hypertext Transfer Protocol*) e a utilização dos documentos hipertexto [2, 3].

Por sua vez, a história da *Internet* no Brasil começa de forma efetiva no ano de 1988, quando a FAPESP (Fundação de Amparo à Pesquisa no Estado de São Paulo), órgão ligado à Secretaria Estadual de Ciência e Tecnologia do estado de São Paulo, buscou o acesso à Rede nos Estados Unidos. A idéia inicial correspondia ao estabelecimento de uma

rede para fins acadêmicos de forma que pesquisadores pudessem compartilhar dados com instituições de outros países [4].

Esta iniciativa fez chegar ao Brasil a *Bitnet* (*Because is Time to Network*), rede que conectava a FAPESP ao FERMILAB, laboratório de Física de Altas Energias de Chicago (EUA). O intercâmbio de informações funcionava por meio de retirada de arquivos e correio eletrônico. O serviço foi inaugurado oficialmente no ano de 1989.

Em 1991, através de uma linha internacional conectada à FAPESP, o acesso à *Internet* foi liberado para instituições educacionais, fundações de pesquisa e órgãos governamentais. O Brasil então passou a participar de fóruns internacionais e trocar arquivos e *softwares* com outros países. No ano seguinte, em 1992, o Ministério da Ciência e Tecnologia inaugurou a RNP (Rede Nacional de Pesquisa) e organizou o acesso à rede no Brasil por meio de um *backbone* (tronco principal da rede). Mesmo nos dias atuais o *backbone* da RNP é o único de alcance nacional no País.

Três anos mais tarde, em 1995, os ministérios das Comunicações (MC) e da Ciência e Tecnologia (MCT) publicaram uma portaria conjunta criando a figura do provedor de acesso privado e liberando a operação comercial da *Internet* no Brasil. Neste mesmo ano, os dois ministérios também afirmaram que para tornar efetiva a participação da Sociedade nas decisões envolvendo a implantação, administração e uso da *Internet*, seria constituído um Comitê Gestor da *Internet*, que contaria com a participação do MC e MCT, de entidades operadoras e gestoras de espinhas dorsais, de representantes de provedores de acesso ou de informações, de representantes de usuários, e da comunidade acadêmica. [4, 5]

Como conseqüência dessas mudanças, o intervalo compreendido entre os anos de 1995 e 1996 registrou um crescimento significativo do número de usuários no país, uma

vez que inúmeros provedores passaram a vender assinaturas para acesso à rede. Este crescimento é ilustrado na tabela 2.1, conforme números obtidos do comitê gestor da *Internet* no Brasil [6].

Mês / Ano	Set / 1990	Ago / 1991	Jul / 1992	Ago / 1993	Dez / 1994	Dez / 1995	Mar / 1996
Total de Usuários	160	560	800	1010	1500	5200	5700

Tabela 2.1: Crescimento do número de usuários, setembro de 1990 a março de 1996.

Apesar destes números consideráveis, no de 1999, o número de usuários já ultrapassava a marca dos 2,5 milhões [4].

Conforme apresentado pelo IBOPE//*Netratings*, empresa formada a partir da *joint-venture* entre o instituto IBOPE e a *Nielsen//Netratings* (líder mundial em medição de audiência pela *Internet*), no Brasil, no período de maio de 2004, a média de acesso da internet foi em torno de 13h e 51 min, superando a média norte-americana, que para o mesmo período foi de 13h e 7 min. No entanto, neste mesmo mês, o Japão apresentou uma média de 16h e 9 min [7].

Estes dados são justificados pelo aumento do número de serviços *on-line* e um aumento na rapidez das conexões, considerando que uma das principais dificuldades até então era o tempo que consumia uma navegação [8].

2.3 DOCUMENTO HIPERTEXTO

Um documento hipertexto contém texto regular, pode ser armazenado, lido ou editado, além de possibilitar conexões que apontem para outros documentos [9].

Na *internet*, estes documentos são utilizados para apresentação da informação de forma estruturada, além de fornecer um meio para interação com os usuários.

A informação é disponibilizada em blocos de texto, chamados de páginas, *web sites* ou sítios. Sua apresentação se dá através de um navegador ou *browser*, aplicação concebida para a interpretação de documentos hipertexto. A referência a outros blocos é chamada de ligações hipertextuais ou *links*.

O conteúdo desses blocos respeita rigorosamente um código, representado por um conjunto de regras chamado *HTML* (*Hypertext Markup Language*). Neste contexto existem códigos abreviados chamados de marcas ou etiquetas (do inglês *tags*), que expressam informações de *layout* (estilos de texto, títulos de documentos, parágrafos, listas, etc) bem como *links*.

```
<HTML>
<HEAD>
<TITLE> Qing Feng Lin's home page</TITLE>
</HEAD>
<BODY>
<P> Hi, this is QingFeng Lin's web page.
    I am a graduate student and a research assistant.
<P> Also, I'm 26 years old.
<P> Here is a pointer to my <A
  HREF="http://www.cs.unl.edu/~sscott"> graduate advisor, Dr. Scott</A>
<P>
</BODY>
</HTML>
```

Figura 2.1: Exemplo de documento hipertexto

A localização de um documento hipertexto é realizada através de um endereço, conhecido como *URL* (*Uniform Resource Locator*). Sua utilização pode ser no computador cliente ou computador servidor, objetivando localizar um certo documento e distingui-lo de outros.

2.4 SERVIÇOS DE BUSCA

Devido à expansão que sofreu ao longo do tempo, a *Internet* se tornou um grande repositório de informação. Este repositório é marcado de maneira notável pelo dinamismo de seu conteúdo, que aumenta e muda a cada dia. Encontrar qualquer tipo de informação nesse gigantesco banco de dados torna-se uma tarefa onerosa [10].

Por este motivo, desde a emergência da *Internet* e da WWW, têm-se buscado maneiras de facilitar e melhorar o processo de aquisição de informações. Neste contexto, os serviços de busca são considerados um dos itens mais bem sucedidos da grande rede, desde o advento de sua criação.

Pela definição presente no glossário ABSOLUTeBUSiNESS[11], os serviços de busca são aplicações computacionais que se destinam a consultar e recuperar informações presentes em bancos de dados, bem como em documentos presentes em diferentes locais de uma rede.

2.4.1 HISTÓRICO DOS SERVIÇOS DE BUSCA

O primeiro serviço de busca foi *Archie*, criado em 1990 por Allan Emtage, um estudante da universidade de McGill em Montreal, num momento em que o método primário para disponibilizar e recuperar arquivos era o protocolo *ftp* (*File Transfer Protocol*). Para compartilhar um arquivo era necessário utilizar um servidor *ftp*, de modo a torná-lo disponível a outros usuários. No entanto, muitos arquivos encontravam-se dispersos em uma infinidade de endereços. A distribuição de um arquivo era feita passando-se a sua descrição, via correio eletrônico, para grupos de discussão. *Archie* mudou esse processo, pois possuía um módulo de coleta que indexava todo arquivo que estivesse presente num

determinado endereço *ftp*. Seu modulo de combinação de expressões regulares (*MATCHER*), provia aos usuários uma forma de consultar sua base de índices.

De forma paralela, foi desenvolvida o *Veronica*, cujo funcionamento era semelhante ao Archie sendo, no entanto destinado a endereços baseados no protocolo *Gopher*.

No que se refere às tarefas de rastrear endereços e indexar arquivos, este mesmo período apresentou o surgimento de uma grande inovação, representada pelo *Word Wide Web Wanderer*, até então o primeiro robô a ser empregado na *Web*. Desenvolvido por Mathew Gray com o propósito de mapear o crescimento da Internet, este software deixou um grande legado, pois os serviços de busca criados a partir de então passaram a utilizar robôs no processo de indexação de arquivos.

Em meados do ano de 1993, três serviços baseados em robôs fizeram os seus debute: *Jumstation*, *World Wide Web Worm*, e o *Repository- Based Software Engineering* (RBSE). Devido à falta de um mecanismo inteligente para interpretar o que realmente estes serviços estavam indexando, surgiram então os diretórios de busca. O pioneiro foi o *EINet Galaxy*, (conhecido mais tarde como *Tradewave Galaxy*). Por sua vez, no mês de abril do ano seguinte, surgiu *YAHOO!*, o representante mais conhecido desta categoria de serviço de busca.

Em dezembro de 1995 foi a vez do *Alta Vista* surgir, adquirindo em pouco tempo o status de serviço de busca mais rápido de seu tempo. Como era executado em um *cluster* de estações *Alpha*, este serviço era capaz de lidar com milhões de requisições por dia. Em 1999, chegou a ser considerado o maior indexador de documentos, devido à evolução dos novos *clusters* surgidos.

O ano de 1998 foi marcado pelo surgimento do *google*, considerado um dos motores de busca mais utilizados na atualidade, existindo em várias línguas e distribuído por vários continentes. Em sua base de dados constam cerca de seis bilhões de itens, majoritariamente páginas, as quais são acessadas cerca de 200 milhões de vezes por dia. Para suportar esse número acentuado de acessos foi criado um *cluster* de cerca de 15000 servidores *Linux* de baixo custo que, num tempo médio próximo do meio segundo, devolve os resultados de uma consulta. Na gênese de sua arquitetura um fator foi essencial: a relação preço/desempenho. O *google* foi também responsável por popularizar o *pagerank*, algoritmo utilizado para ordenar os resultados de uma consulta por relevância [12, 13, 14].

2.4.2 TAXIONOMIA DOS SERVIÇOS DE BUSCA

Segundo WEI[10], os serviços de busca podem ser classificados em três categorias descritas a seguir: diretórios, motores de busca e meta-motores.

Diretórios

Diretórios, ao contrário dos motores de busca, têm o seu conteúdo informado e categorizado de forma não-automatizada, com a finalidade de se obter melhores resultados em consultas. Endereços de *sites*, bem como títulos e descrições, precisam ser submetidos.

Cada diretório apresenta ao usuário um número de categorias no qual um *site* pode ser alocado. Em certos casos, torna-se necessário avaliar em qual categoria um site está sendo colocado. Essa avaliação pode ser feita de forma automática ou via revisão manual, neste caso, feita por uma equipe editorial.

O exemplo mais representativo corresponde ao *Yahoo*. Seu conteúdo é distribuído em pastas e sub-pastas de categorias, todas de fácil navegação.

Motores de busca

A característica fundamental desse tipo de serviço é o emprego de aplicativos denominados robôs ou *spiders*, utilizados para vasculhar endereços da *Internet.*

Motores de busca indexam múltiplos documentos a partir de um *site* e não avaliam a relevância do conteúdo destes documentos. Esse comportamento faz com que motores de busca apresentem uma base de dados com elevado número de registros.

Os exemplos mais representativos dessa categoria são: *Altavista*, *HotBot*, *Excite*, Lycos, Infoseek, *Webcrawler*, *Northern Light*, *Planetsearch*, *GoTo.com*, e o *google.*

Meta -motores

Esta categoria tem como particularidade o seu funcionamento, que está associado a outros motores de busca.

Este funcionamento consiste em submeter uma consulta para cada um dos motores associados, recuperar as listas de endereços geradas e apresentá-las em uma interface visual.

O site *MetaCrawler* é um representante deste tipo de serviço.

2.4.3 ASPECTOS GERAIS DE UM MOTOR DE BUSCA

Nesta subseção, serão descritos os conceitos básicos referentes ao funcionamento de um motor de busca, bem como os diferentes aspectos que permitam traçar uma análise comparativa entre os exemplares existentes.

2.4.3.1 Funcionamento

Um motor de busca possui três componentes principais: um robô, uma base de índices (base de dados), e uma interface de apresentação.

Como visto na subseção anterior, o módulo correspondente ao robô tem como atribuição rastrear endereços da *Internet*. Por sua vez, a base de índices representa um grande banco de dados com o propósito de armazenar referências (URL de documentos) encontradas pelo robô. A interface de apresentação é responsável por submeter uma consulta à base de índices e exibir como resultado uma coleção de endereços de documentos que contemplem os termos empregados na consulta em questão.

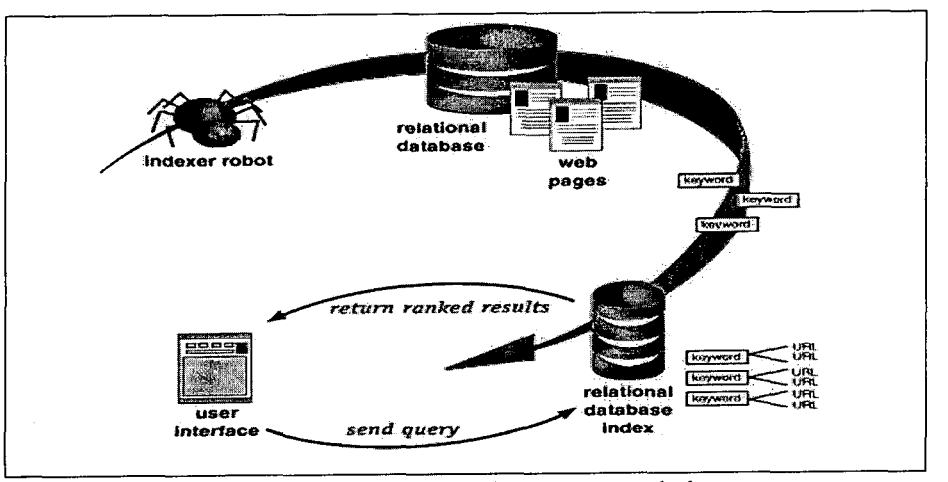

Figura 2.2: Funcionamento de um motor de busca.

2.4.3.2 Diferenças

Há uma variedade de itens ou características que podem ser usados para distinguir os diferentes motores de busca existentes, dentre as quais [10]:

- **Tamanho**

Apesar de todo motor de busca ter como premissa indexar tantos documentos quanto seja possível, ainda não existe nenhum exemplar capaz de cobrir a totalidade da *Web*.

Ainda que muitos avançando tenham sido alcançados, os motores atuais não contemplam esse estado de arte. Isto representa um pequeno déficit de eficácia, pois em muitas ocasiões os usuários estão procurando informações que se encontram numa porção da *Web* ainda não abrangida.

A figura 2.3 mostra o tamanho dos motores de busca, dados estatísticos levantados em maio de 1999, presentes em [15].

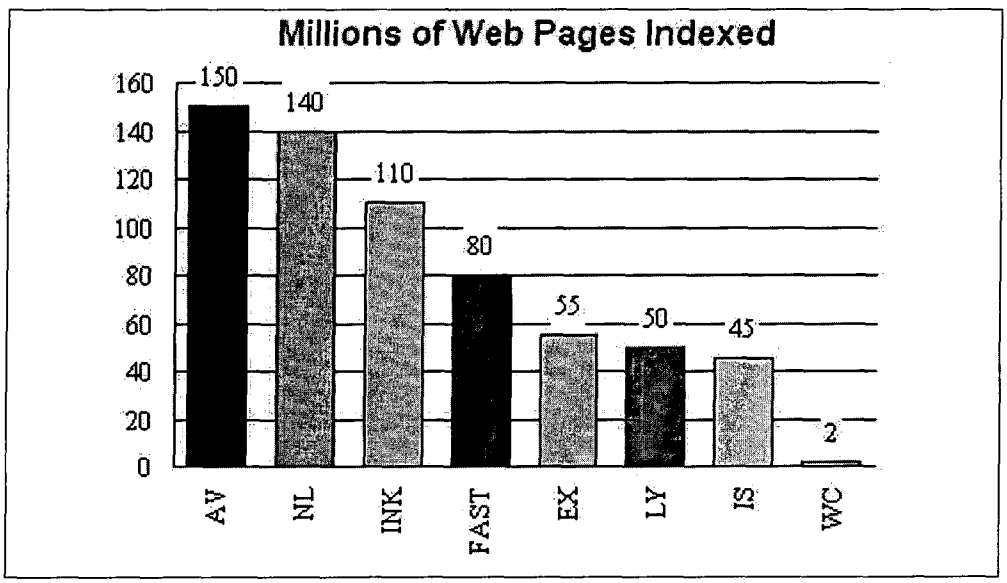

Figura 2.3: Tamanho por milhões de páginas indexadas.

- **Método de indexação**

Existe uma definição em PAL[16], segundo a qual, um índice é um termo extraído a partir de documentos contidos em um dado conjunto. Seu comportamento é semelhante a um ponteiro, pois, para um certo índice, é mantida uma coleção de endereços de documentos

no qual o termo designado ocorre. Esse mapeamento permite facilitar a recuperação de documentos.

Neste contexto, o processo de construção de um índice é chamado de indexação e implica na análise do conteúdo de cada documento a ser mapeado a partir de um termo. Essa análise pode ser executada de forma manual ou automática, tendo como base, em ambos casos, os modelos tradicionais de recuperação de informação: estatístico, texto-completo, e lingüístico [17, 18, 19].

Segundo GIRARDI[17], a indexação baseada no modelo estatístico compreende uma abordagem seletiva, na qual são atribuídos pesos aos termos contidos em um documento, de maneira que estes reflitam uma relação de relevância. Dessa forma, são adotados como índices apenas os termos mais relevantes, que neste caso particular são também nomeados palavras-chaves.

No entanto, como apresentado no manual *Internet and The Web Glossary*[20], a indexação baseada em texto completo transforma em índice toda palavra, relevante ou não, permitindo dessa forma que consultas possam ser executadas a partir de qualquer terminologia.

N modelo lingüístico os índices são obtidos por meio de técnicas de processamento de linguagem natural, como por exemplo: análise morfológica, análise lexical, sintática e semântica[21]. Somente o *Lycos* e o *excite* utilizam indexação baseada neste modelo.

- **Critérios de relevância**

Segundo WEI[10], os algoritmos de *Rank* são algoritmos de ordenação que permitem organizar o resultado de uma consulta, uma vez que este é representado por uma lista de

documentos. Para isso, estes algoritmos dependem de algum critério que torne possível determinar a relevância de um documento.

Neste contexto, os motores de busca podem adotar os seguintes esquemas:

- Localização de palavras-chave: Certos motores de busca atribuem maior relevância a documentos nos quais as palavras chaves ocorram em regiões pré-determinadas, como por exemplo: topo, título ou mesmo a *URL*.
- Freqüência de palavra chave: A relevância de um documento será mensurada pelo número de ocorrências de uma determinada palavra-chave a indexá-lo a partir de uma consulta.
- Popularidade: Neste caso, a relevância de um documento será proporcional ao número de referencias que apontam para este documento. O algoritmo *pagerank*, utilizado pelo *google*, baseia-se neste principio.

2.4.3.3 Problemas

Apesar dos motores de busca terem provido uma melhoria considerável no processo de busca na *Web*, ainda existem questões não resolvidas, dentre as quais WEI[10]:

- Ausência padronização: Cada motor de busca possui o seu próprio robô, seu próprio algoritmo de indexação, bem como um algoritmo de rank em particular.

- Duplicidade de Resultados: Duplicidades nos resultados das consultas ocorrem por conta da natureza dos documentos da *Web*, pois eles contêm *links* apontando para outros documentos. A eliminação de duplicidades presentes na base de índices poderia ser adotada. No entanto, esta solução representa uma tarefa onerosa, e de

certa forma, desnecessária. O objetivo principal de todo motor de busca é prover uma consulta eficiente levando em consideração a velocidade e o tempo de processamento.

- Busca de Multimídia: A grande maioria dos motores de busca trabalha em cima de dados textuais. Não existem motores que consigam lidar com buscas baseadas no conteúdo de arquivos multimídia, o que se tornará uma demanda no futuro próximo.

2.5 MINERAÇÃO DE DADOS DA WEB

Os serviços de busca tradicionais que atuam na *Web* provem um pouco de conforto para os usuários, porém, somados aos problemas citados na subseção 2.4.3.3, estes serviços não provem informação estrutural, categorização, filtragem e interpretação de documentos.

Dessa forma existe um anseio para que a *Web* realmente alcance todo o seu potencial e se torne uma ferramenta mais utilizável, eficaz e compreensível. Nesse contexto a mineração de dados aparece como uma possibilidade óbvia a ser explorada. Em parte pelo seu grande sucesso quando aplicada a bancos de dados tradicionais, e em parte porque a *Web* parece ser uma área fértil em potencial para a aplicação de suas técnicas.

A Mineração de Dados (*Data Mining*) refere-se ao processo não trivial de identificação de informações, previamente desconhecidas e potencialmente úteis a partir de uma grande coleção de dados[22, 23].

No entanto, aplicar esse processo aos dados disponíveis na *Web* não é uma tarefa simples, pois esses dados são muito mais sofisticados e dinâmicos do que os sistemas de armazenamento de bancos de dados tradicionais. Enquanto estes últimos utilizam

estruturas de armazenamento bem definidas e estruturadas, a *Web* não possui qualquer controle sobre a estrutura ou o tipo dos documentos que armazena[24].

Por essa razão, a Mineração de Dados empregada no contexto da web é denominada Mineração de dados da *Web* (*Web Mining*) e corresponde a uma área de pesquisa em particular que visa integrar as tecnologias *Web* e a Mineração de Dados, focalizando o desenvolvimento de novas ferramentas e métodos para análise e descoberta de conhecimento de dados *Web*. Ela também pode ser definida como a descoberta e análise de informações úteis, novas e interessantes da *Web* [25, 26].

A partir das informações descobertas torna-se possível demonstrar características, comportamentos, tendências e padrões de navegação do usuário *Web*, bem como descobrir a hierarquia existente entre os documentos presentes na *Web*[27, 28].

A metodologia utilizada pela Mineração *Web* é composta de quatro etapas, ilustradas na figura 2.4 e descritas a seguir: Recuperação de Informação (*Resource Discovery*), Extração da Informação ou Pré-Processamento (*Information Extraction*), Generalização (*Generalization*) e Análise[24, 29].

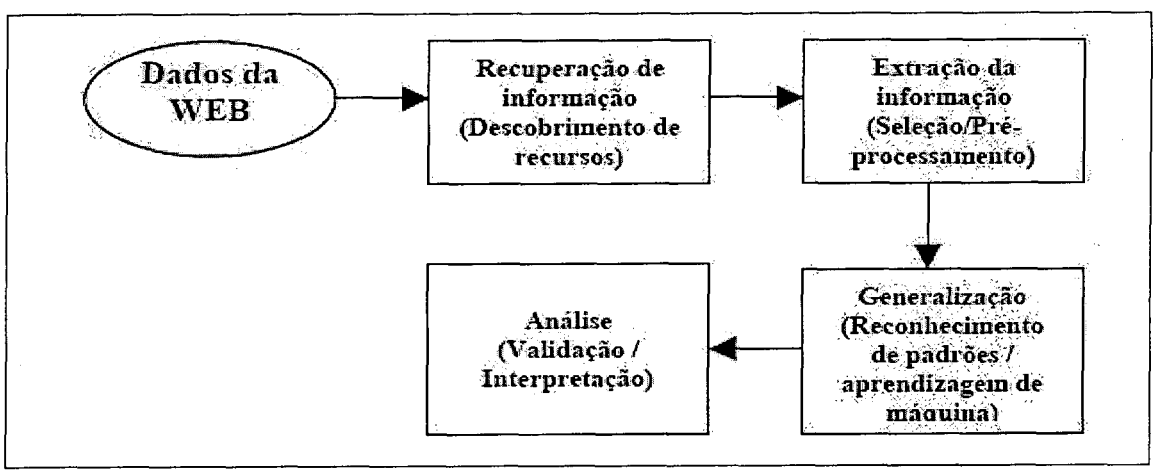

Figura 2.4: Etapas inerentes à Mineração de dados da Web.

- **Recuperação de informação**

A recuperação de informação ou descobrimento de recursos trata da automatização do processo de recuperação de documentos relevantes, que inclui, entre outros, indexação e busca por documentos.

- **Pré-processamento**

Uma vez recuperados os documentos, torna-se necessário convertê-los numa representação adequada que torne possível a aplicação de um algoritmo de mineração de dados de forma efetiva.

O campo disciplinar conhecido como extração de informação presta um grande serviço à mineração da Web, no que diz respeito à fase de extração e pré-processamento da informação. Denomina-se extração de informação à tarefa de identificar fragmentos específicos que constituem o núcleo semântico de um documento em particular e construir modelos de representação da informação a partir dele[16]. Os métodos de extração de informação envolvem o desenvolvimento de *wrappers*, aplicativos baseados em técnicas de *ilp* (*inductive logic programming*) e *nlp* (*natural language processing*), responsáveis por mapear um documento para um modelo de representação do conhecimento.

Com relação ao processo inerente à mineração de documentos hipertextos, essa etapa é freqüente tratada como extração de propriedades de hipertexto (*feature extraction*). A diferença marcante consiste na não utilizada de *wrappers*, e no emprego de heurísticas de seleção de atributos e técnicas de extração de radicais para obter vetores de freqüências[30].

- **Generalização**

Após a construção de um modelo de representação, é executado um algoritmo que possibilite descobrir novo conhecimento a partir das informações existentes, e para tal, este algoritmo deve implementar alguma técnica de mineração de dados.

Muitas técnicas de mineração de dados têm sido usadas para executar atividades comuns de associações, agrupamento, classificação, modelagem, sequenciamento de padrões e previsões de series temporais. Essas técnicas são de caráter genérico e podem ser implementadas através de ferramentas deferentes como Redes Neurais Artificiais, Estatística ou Inteligência Artificial Simbolista.

A tabela 2.2 mostra uma lista das técnicas mais comuns, os algoritmos correspondentes e aplicações típicas de mineração de dados. O tipo de técnica ou função de mineração de dados que estamos tentando executar, junto com a qualidade e quantidade dos dados disponíveis combinam para especificar qual algoritmo deverá ser usado.

Técnica de Mineração	Algoritmo	Aplicação
Associação	Estatística e teoria dos conjuntos	Análise de mercado
Classificação	Arvores de decisão e redes neurais	Mercado alvo, controle de qualidade, controle de qualidade, avaliação de risco.
Analise de Agrupamento	Redes Neurais e estatística	Segmentação de mercado
Modelagem	Regressão linear e não-linear, ajustamento de curvas e redes neurais.	Avaliação dos clientes, modelos de preço, controle de processo.
Previsão de séries temporais	Estatística e redes neurais.	Previsão de vendas, predição de taxa de juros, controle de inventário.
Sequenciamento de padrões	Estatística e teoria dos conjuntos	Analise de mercado ao longo do tempo

Tabela 2.2: Técnicas tradicionais de mineração de dados

No caso particular da *Web*, a grande dificuldade em aplicar as técnicas de mineração refere-se à natureza dos dados empregados, pois os documentos da *Web* não possuem uma marcação semântica associada. Muitos algoritmos de mineração de dados requerem como entrada exemplos positivos e negativos acerca de algum conceito. Dessa forma, seria intuitivo modelar um algoritmo para classificar de forma automática novos documentos como "portais" ou "não-portais" se existisse um conjunto de documentos marcados como exemplos positivos e negativos referentes ao conceito "portal". Dentro desse contexto, a *Web* Semântica provém uma nova forma de construir documentos, permitindo agregar semântica ao seu conteúdo[31].

Dentre as técnicas de mineração tradicionais que podem ser adotadas, a análise de agrupamentos não requer entradas com marcação semântica e por isso tem sido aplicada com sucesso em grandes conjuntos de documentos *HTML* [32]. Neste tipo de técnica, O objetivo é identificar automaticamente grupos de afinidades, avaliando a similaridade entre os documentos, colocando os mais semelhantes no mesmo grupo e os menos semelhantes em grupos diferentes[33, 34].

A análise de Agrupamentos, bem como os seus respectivos algoritmos, será abordada no capítulo 4.

No contexto da *Web*, a técnica de Associação ou Correlação também pode ser empregada. O objetivo desta técnica é encontrar dependências entre os itens presentes em uma transação através da análise de probabilidades condicionais. Em geral, os resultados são apresentados na forma de regras $X \Rightarrow Y$, e expressam a seguinte relação: "Se X está presente numa transação T, então Y tem chances de estar presente também".

Dessa forma, os seguintes exemplos podem ser fornecidos:

- Pessoas que acessam a seção sobre natação também acessam a seção de artes marciais, com probabilidade de 40%.
- Portais que abordam o tema fórmula 1 possuem links apontando para outros endereços que falam da vida de Ayrton Senna, com probabilidade de 35%.

As regras possuem dois graus associados: a confiança e o suporte. O suporte é o número de casos onde a regra foi encontrada (onde X e Y aparecem juntos). A confiança é a probabilidade condicional da regra, ou seja, quais as chances do segundo elemento estar presente. É calculada pela divisão do número de casos onde X e Y aparecem juntos (o suporte absoluto), pelo número de casos onde somente X aparece.

- **Análise**

O último corresponde a analise dos resultados ou saída do processo de mineração. Em alguns casos a saída está numa forma que se torna fácil discernir as informações valiosas dos fatos triviais ou desinteressantes. Em outros casos, contudo, os resultados deverão ser analisados ou visualizados através de outro nível de ferramentas, de forma que os relacionamentos sejam definidos de forma explicita[23].

MOBASHER[35], propõe o sistema WEB-MINER, que utiliza uma linguagem de consulta estruturada para consultar o conhecimento descoberto (na forma de regras de associação e padrões seqüenciais). Outros sistemas utilizam técnicas de OLAP (*On-line Analytical Processing*) [17].

2.5.1 TAXIONOMIA DA MINERAÇÃO DE DADOS DA *WEB*

A mineração de dados da *Web*, de acordo com a parte da *Web* a ser minerada, se divide em três categorias, a seguir descritas [25]: mineração de conteúdo, mineração de estrutura e mineração de uso.

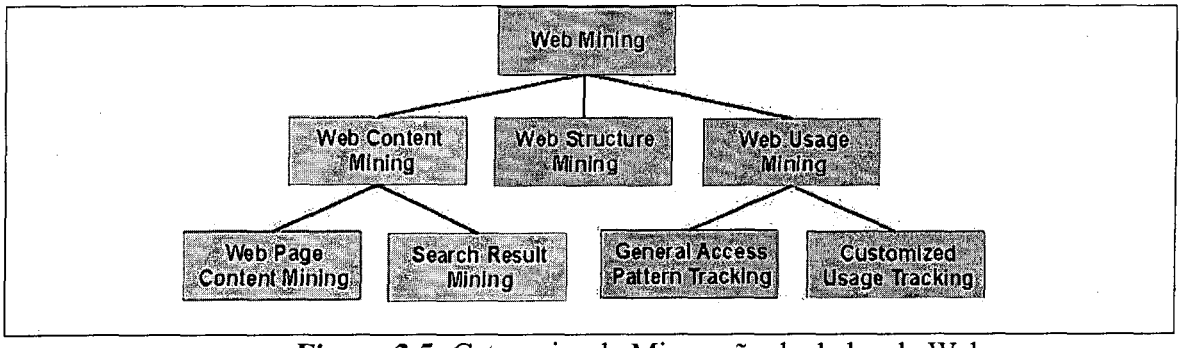

Figura 2.5: Categorias de Mineração de dados da Web

2.5.1.1 Mineração de conteúdo

A mineração de conteúdo trata do descobrimento de informações úteis a partir do conteúdo dos documentos da *Web* [16].

Os dados de textuais da *Web* podem ser de dois tipos: desestruturados, tais como textos comuns, e semiestruturados, tais como documentos *HTML*. No tratamento de dados desestruturados utiliza-se KDT (*Knowledge Discovery in Texts*) ou mineração de dados em textos [36]. Por sua vez, A mineração em hipertexto envolve a mineração de páginas *HTML*, as quais além de texto contém vínculos hipertexto. Um excelente tutorial apresentando esse assunto é descrito em CHAKRABARTI[28].

Segundo MARINHO[24], há uma fina linha separando a mineração de conteúdo e a recuperação de informação na *Web*, pois não há um consenso sobre a relação entre ambas. Há abordagens que tratam a recuperação de informação como uma instancia do processo de mineração de conteúdo [28, 32, 34], enquanto outras associam a mineração de conteúdo com a recuperação inteligente de informação.

Desta forma, existem duas estratégias a serem adotadas pela mineração de conteúdo, ilustradas na figura 2.5: uma realizando a mineração diretamente a do conteúdo dos documentos e a outra incrementando o poder dos serviços de busca. Na primeira estratégia, os documentos pretendidos já foram recuperados e estão prontos para serem minerados. Numa segunda estratégia, a mineração de conteúdo atua como ferramenta de apoio aos serviços de busca, pois ajuda no processo de indexação e categorização de documentos (Agentes Inteligentes), complementando o processo de recuperação de informação. Este segundo caso é freqüentemente descrito como recuperação inteligente de informação [37].

A mineração de conteúdo pode seguir duas abordagens: baseada em agentes ou baseada em bancos de dados. A abordagem baseada em agentes envolve o desenvolvimento de sistemas de inteligência artificial que podem agir de forma autônoma ou semi-autônoma para a descoberta e organização de informações da *Web* de acordo com os interesses de um usuário em particular [26]. Geralmente, esta a abordagem pode ser dividida em três categorias: agentes de busca inteligentes, agentes de filtragem (ou categorização da informação) e agentes de interface [25]. Por usa vez, a abordagem de bancos de dados focaliza-se nas técnicas para transformar os dados semi-estruturados ou desestruturados da Web em modelos de dados estruturados, de forma que uma linguagem de consulta, como, por exemplo, a linguagem SQL, possa ser utilizada como ferramenta de análise.

2.5.1.2 Mineração de Estrutura

Enquanto a mineração de conteúdo tem como objeto de estudo as informações existentes no conteúdo dos documentos da *Web*, a mineração de estrutura apresenta como foco as informações que existem de forma implícita entre estes documentos [24].

As ligações hipertextuais correspondem ao principal objeto de estudo dessa categoria, pois fornecem indicações interessantes acerca de como os documentos se relacionam. Neste contexto, *links* que apontam para um determinado documento podem indicar a importância deste, enquanto *links* apontados por um documento podem indicar, entre outras coisas, a continuação ou o complemento dos tópicos abordados por ele [38].

Esse mesmo comportamento é verificado nas citações bibliográficas, pois quando um artigo é citado com grande freqüência significa que este se trata de um artigo de maior autoridade perante outros que abordem o mesmo tema [24].

Alguns algoritmos foram propostos para a modelagem da topologia da Web tais como o *HITS* (*Hyperlinked Induced Topic Search*) [39] e o *PageRank* [13]. Esses modelos são aplicados principalmente para calcular a qualidade ou relevância das páginas da Web. Uma das regras utilizadas é que quanto mais páginas estiverem apontando para uma determinada página, mais relevantes ela será. Várias medidas são tomadas para garantir que as páginas que apontam tenham credibilidade. Alguns exemplos são o sistema *Clever* [40] e o próprio *google* [13]. Algumas outras aplicações destes modelos são a categorização de páginas Web e a descoberta de micro-comunidades na Web [41].

2.5.1.3 Mineração de Uso

A mineração de uso da *Web* focaliza-se em técnicas que possam prever o comportamento do usuário enquanto ele interage com a *Web* [39]. Enquanto a mineração de conteúdo e a mineração de estrutura utilizam os dados reais ou primários (conteúdo de documentos e *links*), a mineração de uso lida com os dados secundários provenientes da interação do usuário com a *Web*. Os dados de uso incluem dados provenientes de *logs* de servidores *Web*, *logs* de servidores *proxy*, *logs* de *browsers*, perfis de usuário, cookies, seções ou

transações de usuários, pasta favoritos, consultas do usuário, clique de mouse e qualquer outro dado gerado pela interação do usuário com a *Web*.

O processo de mineração de uso pode ser classificado segundo duas abordagens [42]. A primeira mapeia os dados de uso do servidor *Web* em tabelas relacionais antes de um algoritmo de mineração ser aplicado. A segunda utiliza os dados de *logs* diretamente utilizando técnicas especiais de pré-processamento. Assim como no processo tradicional de mineração, a limpeza e o pré-processamento dos dados, aqui, são etapas cruciais, pois a qualidade dos dados vai determinar a eficiência dos algoritmos de mineração [43].

As aplicações da mineração de uso podem ser divididas em duas categorias principais: aprendizado de perfil de usuário (modelagem em interfaces adaptativas ou personalização) e aprendizado de padrões de navegação de usuário. A mineração de uso tem especial apreço no comércio eletrônico, interesse justificado pela necessidade de aprender acerca do comportamento dos clientes, perfis de compra, preferências e padrões de navegação. Alguns *sites* populares de comércio eletrônico já utilizam estas técnicas não só para a adaptação do site de acordo com o perfil do usuário, mas como para fazer recomendações de produtos de acordo com compras anteriores, ou baseadas na similaridade entre perfis de usuários.

Capítulo 3 - Análise de Agrupamentos

3.1 INTRODUÇÃO

A análise de agrupamento (AA) engloba uma variedade de técnicas e algoritmos cujo objetivo é classificar, com respeito a algum critério predeterminado, uma amostra de entidades (indivíduos ou objetos) em grupos mutuamente exclusivos baseado nas similaridades dentre as entidades. Os grupos de objetos resultantes devem então exibir alta homogeneidade interna (dentro do grupo) e alta heterogeneidade externa (entre os grupos). Logo, se um processo de classificação obter êxito, os objetos dentro de um grupo estarão todos juntos quando representados geometricamente, e diferentes grupos estarão separados.

Essa atividade pode ser observada, por exemplo, numa criança brincando com blocos coloridos de diferentes formas, cores e tamanhos. É comum ela separá-los em pilhas segundo uma de suas características, como a cor, por exemplo. Ela está praticando uma análise de agrupamentos (AA). Usar mais de uma característica para formar pilhas torna-se uma atividade mais trabalhosa, exigindo conceitos mais sofisticados de semelhança e procedimentos mais "científicos" para empilhá-las.

Desse modo, o problema que a AA pretende resolver é: "dada uma amostra de n objetos (ou indivíduos), cada um mensurado segundo p variáveis, procurar um esquema de classificação que agrupe os objetos em g grupos. Devem ser determinados também o número e as características desses grupos".

Análise de agrupamentos tem sido referida também como Q-Análise, tipologia, análise de classificação e taxionomia numérica. Essa variedade de nomes se deve em parte

ao uso de métodos de agrupamentos em diferentes áreas do conhecimento como a psicologia, biologia, sociologia, economia, engenharia e negócios. Embora os nomes defiram através das áreas afins, todos os métodos têm uma dimensão em comum: a classificação de dados de acordo com relacionamentos "naturais" entre eles.

Nesse capítulo serão apresentados alguns conceitos básicos e técnicas utilizadas em AA. O próximo tópico situa a AA em um campo mais amplo de estudo chamado de Análise de dados Multivariados, a seguir serão brevemente discutidas algumas questões sobre o tratamento dos dados e finalmente serão apresentadas as principais técnicas de agrupamento e com avaliar os resultados.

3.2 ANÁLISE DE DADOS MULTIVARIADOS

Análise de agrupamento é nome atribuído ao conjunto de técnicas multivariadas cujo propósito primário é agrupar objetos baseados nas características que eles possuem. Técnicas analíticas multivariadas estão sendo amplamente utilizadas na indústria, governo e centros de pesquisa.

Analise de dados multivariados (ou simplesmente análise multivariada) refere – se a todos os métodos estatísticos que simultaneamente analisam múltiplas medidas de cada individuo ou objeto sob investigação. Qualquer análise simultânea de mais de duas variáveis pode ser vagamente considerada Análise Multivariada. Muitas técnicas multivariadas são extensões da análise univariada e bivariada. Por exemplo, Regressão Simples (com uma variável de predição) é estendida no caso multivariado para incluir varias variáveis de predição. Da mesma forma, a única variável dependente achada na Análise de Variância é estendida para incluir múltiplas variáveis dependentes na Análise Multivariada de Variância. Em muitas instâncias, técnicas multivariadas são meios de

executar uma única análise ao invés de múltiplas análises usando técnicas univariadas. Outras técnicas multivariadas, tais como Análise de Fator para identificar a estrutura principal de um conjunto de variáveis ou Análise de Discriminante para diferenciar entre grupos baseado num conjunto de variáveis.

Para ser considerada verdadeiramente multivariada, todas as variáveis devem ser aleatórias e inter – relacionadas de tal forma que seus diferentes efeitos não possam ser significantemente interpretáveis separadamente. O propósito da análise multivariada é medir, explicar e predizer o grau de relacionamento entre as variedades (combinação ponderada de variáveis). Logo, a característica de multivariedade passar por múltiplas variedades, não apenas no número de variáveis ou observações. Análise de multivariada irá incluir ambos, técnicas multivariáveis e verdadeiramente técnicas multivariadas, porque conhecer técnicas multivariáveis é o primeiro passo essencial em entender análise multivariada.

Análise multivariada é sempre um expansível conjunto de técnicas para análise de dados. Dentre as técnicas mais estabelecidas estão:

- Regressão múltipla e correlação múltipla
- Analise múltipla de descriminante.
- Componentes principais e análise de fator comum.
- Análise multivariada da variância e co–variância.
- Correlação canônica
- Análise de agrupamentos
- Escala multidimensional
- Análise Conjunta

Dentre as técnicas emergentes também são incluídas:

- Análise de correspondência.
- Modelos de probabilidade linear.
- Modelagem simultânea / estrutural de equação.

3.3 TRATAMENTO DE DADOS

Em qualquer aplicação, os objetivos da AA não podem ser separados da seleção de variáveis usadas para caracterizar os objetos a serem agrupados. Os resultados possíveis estão diretamente ligados pela seleção das variáveis. Os grupos derivados devem apenas refletir a estrutura hierárquica dos dados definida pelas variáveis.

3.3.1 SELEÇÃO DE VARIÁVEIS

O resultado de uma AA deve ser um conjunto de grupos que podem ser consistentemente descritos através de suas características, atributos e outras propriedades. De maneira conjunta, esses descritores são as variáveis do problema. Assim, um dos fatores que mais influencia o resultado de uma análise de agrupamentos é, indiscutivelmente a escolha das variáveis.

Variáveis que assumem o mesmo valor para todos os objetos não são suficientemente descriminantes de maneira que sua inclusão pouco contribuiria para a determinação da estrutura do agrupamento. Por outro lado, a inclusão de variáveis com grande poder de discriminação, porém, irrelevantes ao problema, pode mascarar os grupos e levar a resultados equivocados. Além disso, é desejável que os objetos sejam comparáveis segundo o significado de cada uma delas. Por exemplo, o tamanho da frota de ônibus das

cidades brasileiras informa mais sobre as populações dessas cidades do que suas condições de transporte público.

Freqüentemente, o número de variáveis medidas é grande, dificultando a análise a análise. Deve-se então procurar diminuir o seu número de forma que sua seleção contemple tanto a sua relevância como o sua capacidade de discriminação face ao problema em estudo.

Existem dois tipos básicos de dados: não-métricos (qualitativos) e métricos (quantitativos).

Dados não-métricos são atributos, características ou propriedades categóricas que identificam ou descrevem um objeto. Eles descrevem diferenças nos tipos e qualidades indicando a presença ou ausência de uma característica ou propriedade. Muitas propriedades são discretas tendo uma característica particular enquanto todas as outras características são excluídas. Por exemplo, se alguém é do sexo masculino, este não pode ser do sexo feminino. Não existe uma quantidade de gênero, apenas o estado de ser masculino ou feminino.

Ao contrário, mensurações de dados métricos são feitas tal que os dados possam ser identificados como diferenças em quantidade ou grau. Variáveis medidas metricamente refletem quantidade relativa ou distância. Medidas métricas são apropriadas onde se pode fazer declarações como a quantidade ou magnitude, tal como o nível de satisfação ou compromisso para um emprego.

3.3.2 ESCALA DE VARIÁVEIS

Um aspecto importante a ser considerado é a homogeneidade entre variáveis. Ao se agrupar observações é necessário combinar todas as variáveis em único índice de similaridade, de

forma que a contribuição de cada variável depende tanto de sua escala de mensuração como daquelas demais variáveis. Há casos em que a variação de uma unidade em uma variável expressa em toneladas é menos significativa que a variação de uma unidade medida em kg em outra variável. Visando reduzir o efeito de escalas diferentes surgiram várias propostas de relativização das variáveis. Algumas mais comuns são:

Considere as observações originais $x_1,...,x_n$.

A transformação mais comum é aquela definida por:

$$z_i = \frac{x_i - X}{s}, i = 1,..,n$$

onde X e s denotam respectivamente, a média e o desvio padrão das observações.

Esta transformação, conhecida como norma-Z, faz com que os dados transformados tenham média zero e variância unitária. A desvantagem desta padronização é reduzir todas as variáveis ao mesmo grau de agrupabilidade. Outra forma de se transformar variáveis é tornar-se os desvios em relação ao menor valor e normaliza-los pela amplitude, ou seja,

$$z_i = \frac{x_i - x_{(1)}}{x_{(n)} - x_{(1)}}, i = 1,...,n$$

onde $x_{(i)}$ e $x_{(n)}$ denotam o mínimo e o máximo da amostra, respectivamente. Esta transformação reduz os dados para o intervalo [0,1].

Tornando-se a média como fator normalizador pode-se definir

$$z_i = \frac{x_i}{X}, i = 1,...,n$$

A despeito da variedade de propostas, recomenda-se que a escala das variáveis seja definida através de transformações sugeridas pelo bom senso e pela área de conhecimento da aplicação.

3.4 MEDIDAS DE DISTÂNCIA E SIMILARIDADE

Um conceito fundamental na utilização das técnicas de AA é a escolha de uma de um critério que meça a distância entre dois objetos, ou que quantifique o quanto eles são parecidos. Essa medida será chamada de coeficiente de semelhança. Cabe observar que tecnicamente pode-se dividir em duas categorias: medidas de similaridade e medidas de dissimilaridade. Na primeira, quanto maior o valor observado mais parecidos são os objetos. Já na segunda, quanto maior o valor observado menos parecidos (ou mais dissimilares) serão os objetos. Coeficientes de correlação é um exemplo de medida de similaridade, enquanto que a distância euclidiana é uma medida de dissimilaridade. A maioria dos algoritmos de AA estão programados para operarem com o conceito de distância (dissimilaridade).

Existem várias formas para mensurar o quanto objetos são parecidos, mas três métodos em especial dominam o cenário de aplicações AA: *medidas de correlação*, *medidas de distância* e *medidas de associação*. Cada método representa uma perspectiva de similaridade, dependendo dos objetivos e do tipo de dados. As medidas de *correlação* e de *distância* requerem dados métricos enquanto as medidas de associação necessitam de dados *não-métricos*.

3.4.1 MEDIDAS DE CORRELAÇÃO

Medidas de correlação representam similaridade pela correspondência dos padrões através de suas características (variáveis). Uma medida de correlação não olha para a magnitude dos valores dos dados, mas sim os padrões desses valores. Devido a isso, medidas de correlação são raramente usadas na maioria das aplicações AA.

3.4.2 MEDIDAS DE DISTÂNCIA

Medidas de distância representam similaridade como a proximidade entre as observações através das variáveis. São as medidas de similaridade mais usadas.

Uma das medidas de distância mais comumente utilizadas para indicar a proximidade entre dois objetos A e B, dado o vetor $\boldsymbol{x}$ de coordenadas reais, $(x_1,..,x_n)$, é a distância euclidiana (DE), expressa por:

$$d(A,B) = \sqrt{\sum_i (x_i(A) - x_i(B))^2}$$

Por sua vez, existem variações da distância euclidianas, criadas com o propósito de calcular medidas específicas. Uma primeira variação da DE, muito usada em AA, é a distância euclidiana média, onde a soma das diferenças ao quadrado é dividida pelo número de coordenadas envolvidas, n:

$$d(A,B) = \sqrt{\frac{\sum_i (x_i(A) - x_i(B))^2}{n}}$$

Esta última expressão é apenas um re-escalonamento da distância euclidiana original, possuindo as mesmas propriedades, portanto produz os mesmos resultados se submetidos às mesmas técnicas de AA.

Entretanto, esta última medida possui duas propriedades interessantes:

1. Ela pode ser usada na ausência de dados para algumas coordenadas (*missing values*)
2. Permite acumular evidências empíricas sobre níveis semelhança.

Outro tipo de DE é a distância euclidiana quadrada, ou absoluta, onde o valor da distância é a soma dos quadrados da diferença sem tomar a raiz quadrada:

$$d(A,B) = \sum_i (x_i(A) - x_i(B))^2$$

A distância euclidiana quadrada tem a vantagem de não tomar a raiz quadrada, o que aumenta sensivelmente a velocidade computacional, e é a medida de distância recomendada para os métodos de agrupamento *Centróide* e *Ward* que serão descritos mais adiante.

Varias distâncias não baseadas na distancia euclidiana também estão disponíveis. Uma das alternativas mais usadas envolve a troca dos quadrados da diferença pela soma das diferenças absolutas das variáveis. Este procedimento é chamado de função de distância absoluta ou "*city-block*":

$$d(A,B) = \sum_i | x_i(A) - x_i(B) |$$

A abordagem *city-block* para calcular distâncias pode ser apropriada sob certas circunstâncias, mas causa vários problemas. Um deles é a suposição de que as variáveis não estão correlacionadas entre si; se elas estiverem correlacionadas, os grupos não são válidos.

Um problema enfrentado por todas as medidas de distância que usam dados não padronizados é a inconsistência entre as soluções de grupos quando a escala de variáveis é mudada.

Uma variação da distância euclidiana que incorpora diretamente um procedimento de padronização (normalização) é distância de *Mahalanobis* dada por:

$$(x\ (A) - x\ (B))' S^{-1} (x(A) - x(B)),$$

onde S é a matriz de covariâncias e $x(A)$ e $x(B)$ são os respectivos vetores de medidas dos objetos A e B.

Nessa abordagem de *Mahalanobis*, os dados são normalizados escalando respostas em termos de desvios padrão, e ajustes são feitos para intercorrelação entre as variáveis.

Na maioria das situações, diferentes medidas de distância podem levar a diferentes soluções de grupos. Logo, é aconselhável usar várias medidas e comparar os resultados com padrões teóricos ou conhecidos.

3.4.3 MEDIDAS DE ASSOCIAÇÃO

Medidas de associação de similaridade são usadas para comparar objetos cujas características são medidas apenas em termos não-métricos (medidas nominais ou ordinais). Como por exemplo, entrevistados poderiam responder sim ou não para um número de perguntas. Uma medida de associação poderia avaliar o grau de concordância entre cada para de entrevistados. A forma mais simples de medida de associação poderia ser a percentagem das vezes que houve concordância (ambos os entrevistados disseram não ou disseram sim para a questão) no conjunto de questões. Extensões deste simples coeficiente de comparação têm sido desenvolvidas para acomodar variáveis nominais multicategóricas e até medidas ordinais.

3.5 FORMANDO AGRUPAMENTOS

A formação de agrupamentos baseia-se em duas idéias básicas: coesão interna dos objetos e isolamento externo entre os grupos. Todos os algoritmos de agrupamento tentam maximizar as diferenças entre grupos relativas à variação dentro dos grupos, como mostrada na figura 3.1.

Os algoritmos mais comumente usados para formar agrupamentos podem ser classificados em duas categorias gerais: (1) técnicas hierárquicas e (2) técnicas não-hierárquicas ou métodos de participação.

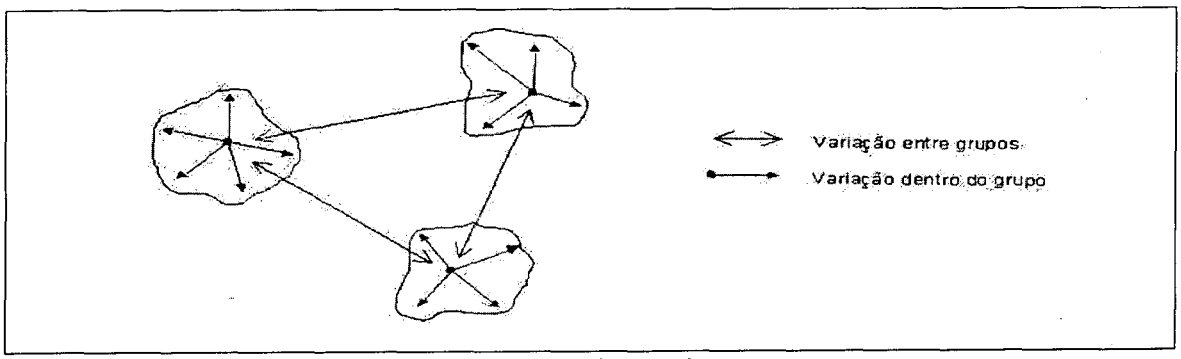

Figura 3.1: Coesão interna dos objetos e isolamento externo entre os grupos.

3.5.1 TÉCNICAS HIERÁRQUICAS

O que caracteriza estes processos é que a reunião de dois grupos numa certa etapa produz um dos grupos da etapa superior, caracterizando o processo hierárquico e permitindo a construção de um *dendograma* (diagrama em forma de árvore). As técnicas hierárquicas podem ser subdivididas em dois tipos: *aglomerativas* e *divisivas*.

Nos métodos aglomerativos, cada objeto inicia formando um grupo com ele mesmo. Nas etapas subseqüentes, os dois grupos (ou indivíduos) mais próximos são

combinados para formar um novo grupo, assim, através de fusões sucessivas dos *n* objetos, vão sendo obtidos n-1, n-2, grupos, até reunir todos os objetos num único grupo.

Quando o processo de agrupamento procede na direção oposta dos métodos aglomerativos, ele é referido como sendo um método *divisivo*. Em se tratando de métodos divisivos, a construção de agrupamentos começa com um grande grupo contendo todas as observações (objetos). Nas etapas subseqüentes, os objetos que são mais dissimilares sãos separados para formar grupos menores. Este processo continua até que cada objeto é grupo em si mesmo.

Na figura 3.2 é apresentado um exemplo de um dendograma. Métodos aglomerativos moveriam da esquerda para a direita, e métodos divisivos moveriam da direita para a esquerda no dendograma.

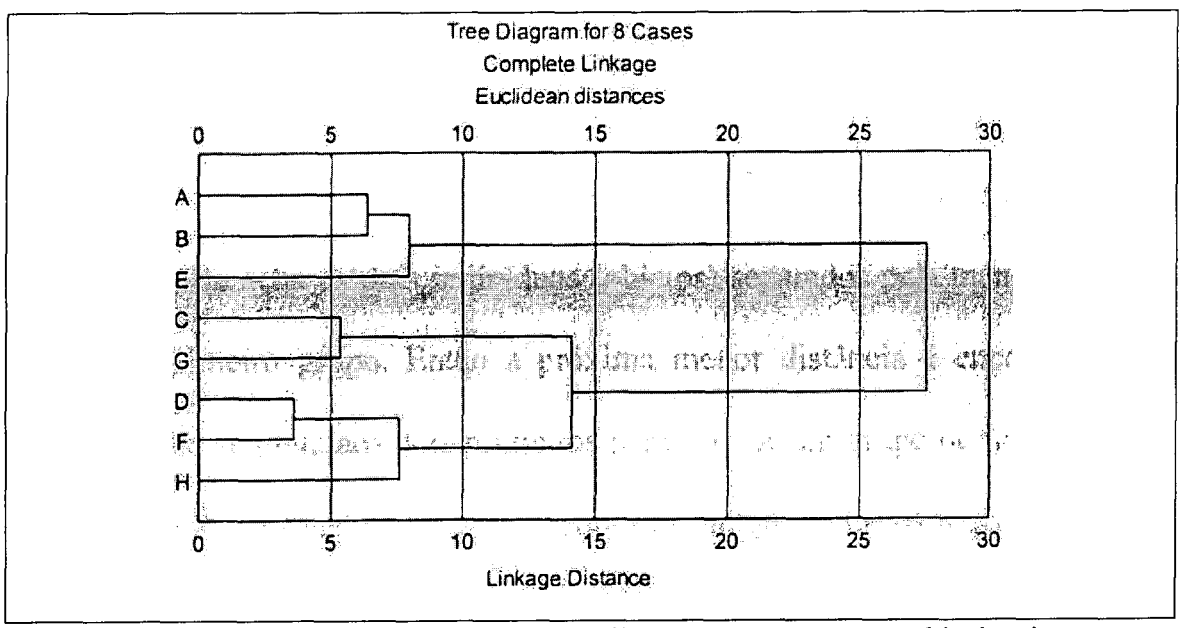

Figura 3.2: Exemplo de um dendograma ilustrando agrupamentos hierárquicos.

O que caracteriza os algoritmos de produzir agrupamentos é o critério usado para definir a distância entre grupos. Como os métodos aglomerativos são mais comumente

usados que os divisivos e estes agem quase ao inverso dos métodos aglomerativos, serão apresentadas a seguir as principais técnicas aglomerativas para achar agrupamentos:

- **Método do Centróide (MC)**

Este processo é o mais direto deles, pois substituí cada fusão de objetos num único ponto representado pelas coordenadas de seu centro que é a média dos valores dos objetos que estão no grupo. A distância entre grupos é definida pela distância entre os seus centros. Em cada etapa procura-se fundir grupos que tenham menor distância entre si. Este método pode produzir resultados desordenados e confusos. Isto acontece devido às inversões, isto é, exemplos onde a distância entre os centros de um par pode ser menor que a distância entre outro par unido numa combinação anterior. A vantagem deste método é que ele é menos afetado por dados não-representativos (*outliers*) que outros métodos hierárquicos.

- **Método da Ligação Simples (MLS) ou do vizinho mais próximo.**

Este método define como critério de semelhança, similaridades ou dissimilaridades (parecença), entre dois grupos aquele dado pelos dois membros mais parecidos. Ele acha os dois indivíduos (objetos) separados pela menor distancia e os coloca no primeiro grupo. Então a próxima menor distancia é encontrada e um terceiro individuo se junta aos dois primeiros para formar um grupo ou um novo grupo de dois indivíduos é formado. O processo continua ate que todos os indivíduos estejam em um grupo.

Assim, dados os conjunto de objetos X e Y, a distância entre eles será definida como:

$$d(X,Y) = \min\{d(i,j) : i \in X, j \in Y\}$$

A distancia entre quaisquer dois grupos é a menor distância de qualquer ponto em um grupo para qualquer outro ponto no outro. Dois grupos são unidos em qualquer

etapa simplesmente pela menor ou mais forte ligação entre eles. Porém, problemas ocorrem quando os grupos não estão bem delineados. Nesses casos, a ligação simples forma longos encadeamentos, e eventualmente todos os indivíduos são colocados em uma cadeia. Indivíduos nos extremos da cadeia podem ser muito dissimilares.

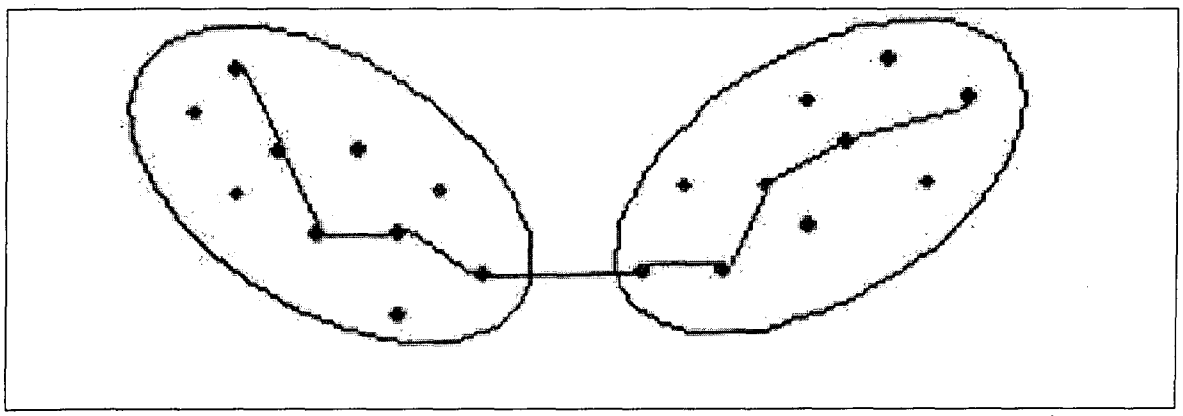

Figura 3.3: Exemplo de Ligação Simples unindo pontos de grupos distintos.

- **Método da Ligação Completa (MLC) ou do Vizinho Mais Longe**

É parecido com o método anterior exceto que o critério de semelhança é baseado na distancia máxima:

$$d(X,Y) = \max\{d(i,j) : i \in X, j \in Y\}$$

A distância máxima entre indivíduos em cada grupo representa a menor esfera (diâmetro mínimo) que pode fechar todos os objetos nos grupos. Este método é chamado de ligação completa porque todos os objetos no grupo são ligados a alguma distancia máxima ou similaridade mínima. Pode-se dizer que a similaridade intragrupo é igual ao diâmetro do grupo. Esta técnica elimina o problema de encadeamento identificado na ligação simples. Convém ressaltar que a fusão ainda é feita com os grupos mais parecidos, menor distância.

A figura 3.4 mostra como a distancias, mais curta (ligação simples) e a mais longa (ligação completa), representam similaridade entre grupos. Ambas as medidas refletem apenas um aspecto dos dados. O uso da menor distância reflete apenas um único par de objetos (os mais próximos), enquanto que a ligação completa reflete novamente um único par, neste caso os dois os dois mais extremos. Isto é então útil para visualizar as medidas refletindo a similaridade do par mais similar ou do par menos similar de objetos. A figura mostra também a distancia utilizada pelo método do centróide que é dada pela distancia entre os centros dos grupos.

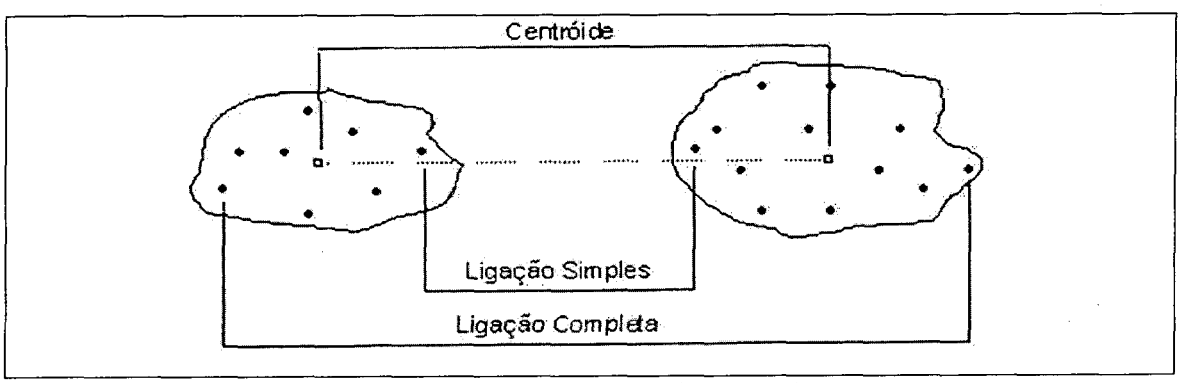

Figura 3.4: Diferentes formas para medir distancia entre os grupos.

- **Método das Médias das distancias (MMD)**

O método das Medias das distancias começa da mesma forma que a ligação simples e completa, mas o critério de agrupamento é a distancia média de todos os indivíduos em um grupo para todos os indivíduos em outro. Tais técnicas não dependem de valores externos, como na ligação simples ou ligação completa, e o particionamento é baseado em todos os membros do grupo ao invés de um único par de membros externos. Uma das formas de computar distancia é simplesmente tomarmos:

$$\frac{1}{n_I n_J} \sum \sum d_{ij} ,$$

onde d_{ij} é a distancia entre o objeto i pertence ao primeiro grupo e o objeto j pertencente ao segundo grupo, a soma é sobre todos os possíveis pares de variáveis entre os dois grupos e n_I e n_J são as quantidades de objetos nos dois grupos.

A abordagem de ligação média tende a combinar grupos com pequenas variâncias.

- **Método de Ward**

Ward [44] propôs um método para construir agrupamentos baseado na perda de informação resultante do agrupamento de indivíduos, medido pela soma total de divergências quadradas de toda observação a partir da média do agrupamento à qual pertence. A regra apóia-se no aumento da soma dos erros quadrados induzida da combinação de todo possível par de agrupamentos. Este valor, que será denotado por ESS (*Error sum of Squares*), é usado como função objetivo [45].

O processo de agrupamento começa com n grupos onde cada elemento forma um grupo. O primeiro grupo é formado selecionando dois destes grupos que irão produzir o menor valor aumento no valor da função objetivo. Os n-1 grupos restantes são então reexaminados para determinar os próximos dois grupos que irão se unir enquanto minimiza o aumento da função objetivo. Os *n* grupos iniciais são então sistematicamente reduzidos de n para n-1, n-2 até 1 grupo formando o procedimento de agrupamento hierárquico.

O ESS é computado como:

$$ESS = \sum_{j=1}^{k}\left(\sum_{i=1}^{nj} X_{ij}^2 - \frac{1}{n_j}\left(\sum_{i=1}^{nj} X_{ij}\right)^2\right),$$

onde X_{ij} denota o valor do *i*-ésimo individuo no *j*-ésimo agrupamento, *k* é o número total de agrupamentos a cada estágio e n_j é o número de indivíduos no *j*-ésimo

agrupamento. A cada etapa no procedimento de agrupamento, a soma dos quadrados dos grupos é minimizada sobre todas as partições possíveis de obter combinando dois grupos do estágio anterior. Este procedimento tende a combinar grupos com pequeno número de observações.

3.5.2 MÉTODOS DE PARTIÇÃO OU TÉCNICAS NÃO–HIERÁRQUICAS

Estes métodos procuram diretamente uma partição dos n objetos, de modo que satisfaçam às duas premissas básicas de coesão interna e isolamento dos grupos. O uso dos métodos de partição pressupõe também o conhecimento dos número $\underline{k}$ de partições desejadas.

Ao contrário dos métodos hierárquicos, os procedimentos não-hierárquicos não envolvem a construção de um dendograma. Em vez disso, eles designam objetos para os grupos uma vez que o número de grupos a serem formados é especificado. Logo, uma solução com sei grupos não é apenas a combinação de dois grupos da solução anterior de sete grupos, mas é baseado apenas em encontrar a melhor solução com seis grupos.

De uma forma geral, o processo funciona da seguinte forma: o primeiro passo é selecionar um centro (uma semente) do grupo como o centro inicial do grupo, e todos os objetos (indivíduos) dentro de uma pré-especificada distância são incluídos no grupo resultante. O critério mais usado é o da soma de quadrados residual, inspirado em Análise de Variância. Então, outra semente do grupo é escolhida e as atribuições continuam até que todos os objetos sejam atribuídos a algum grupo. Por sua vez, os objetos podem ser realocados se eles estiverem mais próximos de outro agrupamento do que daqueles que lhes foram originalmente atribuídos. Procedimentos não - hierárquicos de agrupamento são freqüentemente referidos como métodos das *K-médias* (*K-means*) e talvez seja um dos mais

usados em AA quando se tem muitos objetos. Tipicamente usam uma das três abordagens a seguir para atribuir observações individuais a um dos grupos:

- **Seqüencial:** seleciona uma semente de grupo e inclui todos os objetos dentro de uma distância pré-especificada. Quando todos os objetos dentro da distância são incluídos, uma segunda semente do grupo é selecionada, e todos os objetos dentro da distância pré-especificadas são incluídos. Então uma terceira semente é selecionada, e o processo continua como antes. Quando um objeto é agrupado com uma semente, ele não é mais considerado para as sementes subseqüentes.

- **Paralelo:** seleciona várias sementes de grupos simultaneamente no começo e atribui objetos dentro da distância da semente mais próxima. Conforme o processo sofre evolução, as distâncias podem ser ajustadas para incluir menos ou mais objetos nos grupos. Também, em alguns métodos, objetos se mantém não agrupados se eles estão fora da distância limiar pré – especificada de qualquer semente de grupo.

- **Otimizado:** similar aos outros dois exceto que ele permite a realocação de objetos. Se, no curso da atribuição de objetos, um objeto fica mais próximo de outro grupo que não é o grupo ao qual ele foi atribuído originalmente, então um procedimento de otimização irá trocar o objeto para o grupo mais similar (mais próximo).

O maior problema enfrentado por todos os procedimentos de agrupamento não – hierárquicos é como selecionar a sementes dos grupos. Por exemplo, com a opção de seleção seqüencial, o resultado inicial e provavelmente o resultado final, dependem da

ordem das observações no conjunto de dados, o embaralhamento na ordem dos dados provavelmente irá afetar os resultados. Especificando as sementes de grupos iniciais como no procedimento paralelo pode produzir este problema. Mas mesmo que a seleção das sementes dos grupos seja feita aleatoriamente, irá produzir diferentes resultados para cada conjunto de pontos aleatórios de sementes. Logo, deve–se ter cuidado no impacto do processo de seleção de sementes nos resultados finais.

3.5.3 Escolha da Técnica

Uma outra questão muito importante em AA é a escolha da técnica a ser utilizada. Algoritmos que produzem árvores são difíceis de analisar na presença de muitos objetos, enquanto que processos de partições não são reveladores das similaridades ou dissimilaridades (parecença) entre eles. Possivelmente a grande variedade de técnicas se deve às diferentes concepções de agrupamento. Um conceito intuitivo é o chamado agrupamento natural, que é formado por regiões quase vazias. A figura 3.5 ilustra alguns agrupamentos naturais cujas estruturas são óbvias, mas nem por isso seriam facilmente identificados pela maioria das técnicas. Isso se deve ao fato dos métodos de análise de agrupamentos fazerem suposições implícitas sobre o tipo de estrutura presente nos dados. Por exemplo, o agrupamento da figura 3.5(a) seria identificado pela maioria das técnicas porque os grupos são esféricos e estão bastante separados. O método do vizinho mais próximo (MLS) aplicado aos dados da figura 3.5(d) produziria um resultado equivocado porque o par de objetos mais próximos poderia consistir de componentes pertencentes a grupos diferentes. Este exemplo ilustra o fato que o MLS é incapaz de delinear grupos pouco separados. Por outro lado, essa técnica tem a propriedade de "encadeamento", que é

a tendência de produzir grupos do tipo serpentina, e possivelmente seria o único a reproduzir o agrupamento da figura 3.5(c).

Como cada método de análise de agrupamento impõe um certo grau de estrutura nos dados, e para assegurar que o resultado obtido não é um artefato da técnica utilizada, recomenda–se aplicar diferentes critérios de agrupamento e aceitar a estrutura resultante da maior parte deles.

Outra maneira de verifica a estabilidade do agrupamento consiste em particionar, ao acaso, o conjunto de observações em dois subconjuntos e aplicar o mesmo critério em cada um deles. Se o agrupamento for estável, a alocação dos objetos nas sub-amostras será semelhante aquela na amostra integral.

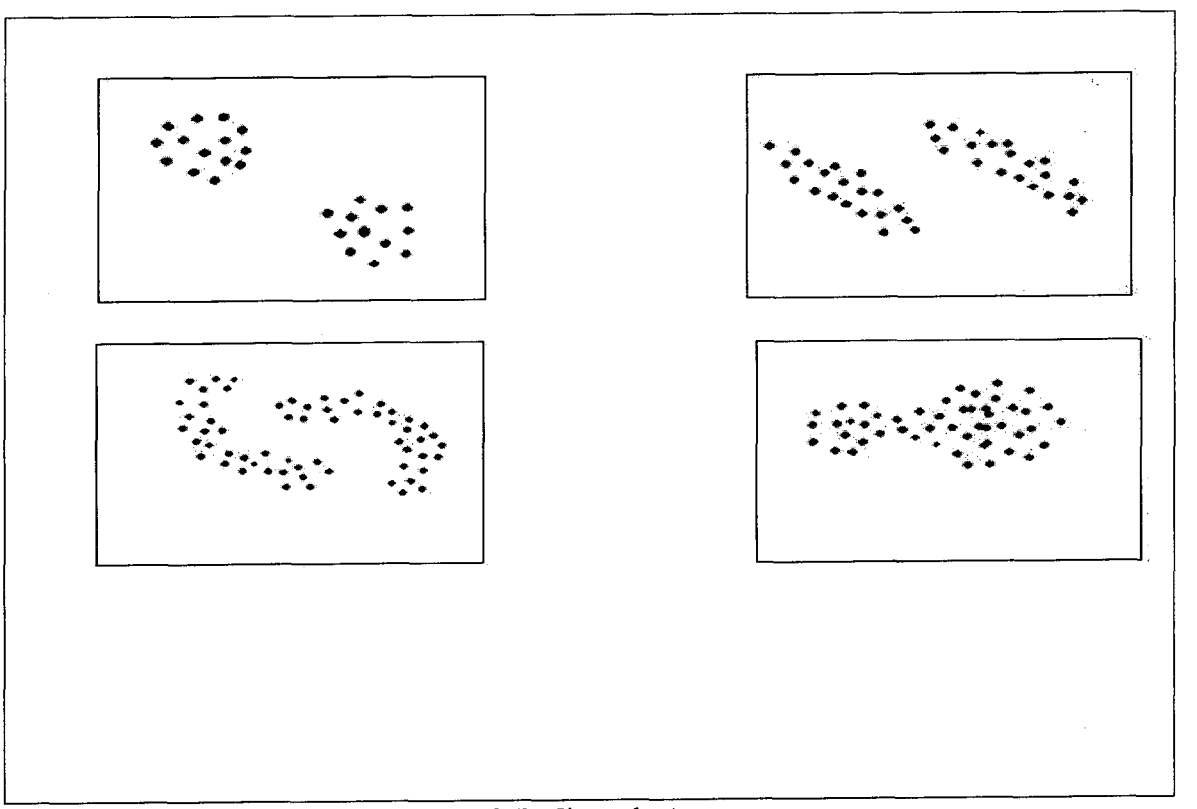

Figura 3.5: Tipos de Agrupamentos.

Capítulo 4 - Redes Neuronais *ART*

4.1 INTRODUÇÃO

As redes *ART* (*Adaptive Resonance Theory*) são apropriadas para aplicações de reconhecimento de padrões, bem como para a análise de agrupamentos Elas foram projetadas para que o usuário possa controlar o grau de similaridade entre padrões que são colocados em um mesmo grupo [46].

A Teoria da Ressonância Adaptativa, inventada por Stephen Grossberg em 1976, é o principio que fundamenta todas as redes dessa família[47]. O primeiro modelo, chamado de *ART*-1, era meramente um classificador de vetores, que recebendo como parâmetro de entrada um vetor, o classificava dentro de um número de categorias estocadas pré-existentes. O vetor era classificado dentro da categoria na qual houvesse maior semelhança [48].

Logo após a apresentação de *ART*-1 para trabalhar com vetores de números binários, no ano de 1987 foi apresentada uma arquitetura mais complexa chamada de *ART*-2 que trabalha com valores contínuos [49]. Posteriormente *ART*-2 foi reapresentada por ANDERSON *et al.* [50].

Para fundamentar o surgimento da Teoria da Ressonância Adaptativa, será ilustrado na subseção 4.2 o aprendizado competitivo. Na subseção 4.3 será examinado o modelo *ART*-1, enquanto que uma versão simplificada do modelo *FUZZY-ART* será apresentada na subseção 4.4.

4.2 APRENDIZADO COMPETITIVO

Em uma rede neuronal que utilize o aprendizado competitivo, os neurônios da camada de saída competem entre si para classificar um padrão e, por conseguinte obter o status de ativo (função de saída igual a um). Por apresentarem este comportamento, os neurônios da camada de saída também são chamados de unidades *winner-take-all* e ao contrario do aprendizado hebbiano, no qual vários neurônios da camada de saída se apresentam ativos ao mesmo tempo, no aprendizado competitivo apenas um neurônio da camada de saída estará ativo a cada vez.

O objetivo prático das redes neuronais que utilizam esse aprendizado é categorizar ou construir agrupamentos a partir um conjunto de dados de entrada.

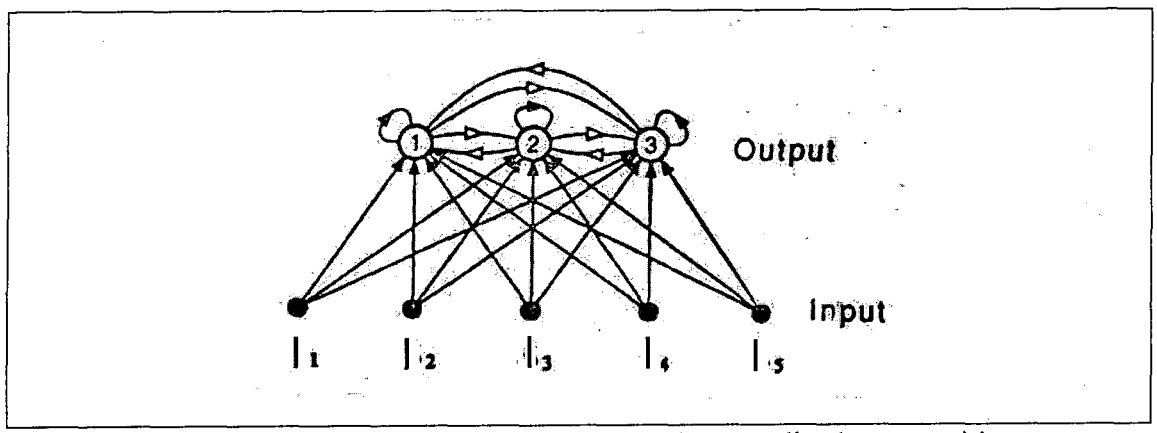

Figura 4.1: Arquitetura de rede neuronal usando aprendizado competitivo.

No modelo apresentado na figura 4.1, existe uma única camada de saída composta de unidades nomeadas O_i, e cada uma destas se apresenta intensamente conectada a todas as unidades da camada de entrada, nomeadas I_j, via conexões excitatórias w_{ij}. Por definição, a unidade vencedora é aquela que apresenta o maior *net*

dentre todas as outras unidades de saída. O *net*, para um vetor de entrada corrente, é calculado da seguinte forma:

$$\text{(I) - } net_i = \sum_j w_{ij} I_j = w_i . I$$

Baseada em (I), a inequação (II) $w_i^* I \geq w_i I$ (aplicada a todo i) define a unidade vencedora i^* com $O_i^* = 1$ (Função de saída para O_i^* igual a um). Se os pesos referentes a cada unidades estão normalizados, de tal forma que para todo i $|w_i| = 1$ então a inequação(I) anteriormente apresentada é equivalente a :

$$\text{(III) } |w_i^* - I| \leq |w_i - I|$$

Por sua vez, a inequação (III) demonstra que a unidade vencedora é sempre aquela cujo vetor de pesos normalizados **w** está mais próximo do padrão de entrada **I**. Em se tratando de uma simulação computacional, a maneira como a estrutura *winner-take-all* é implementada pode ser abstraída. Neste caso, ela é reduzida meramente à busca de um *net* de valor máximo.

Após encontrar um vencedor, é preciso aplicar uma regra de aprendizado. Neste contexto, um neurônio ou unidade de saída aprende deslocando seu respectivo vetor de pesos (sinapse) para próximo do padrão de entrada classificado. Segundo a regra de aprendizado padrão, as seguintes mudanças são aplicadas ao vetor de pesos referentes à unidade vencedora:

$$\underline{\Delta w_{ij}^* = \eta(I_j - w_{ij}^*)}$$

Com relação às outras unidades, as configurações de suas sinapses são mantidas inalteradas.

Para ilustrar a essência do aprendizado competitivo é necessária a utilização de uma analogia geométrica, como apresentada na figura 4.2.

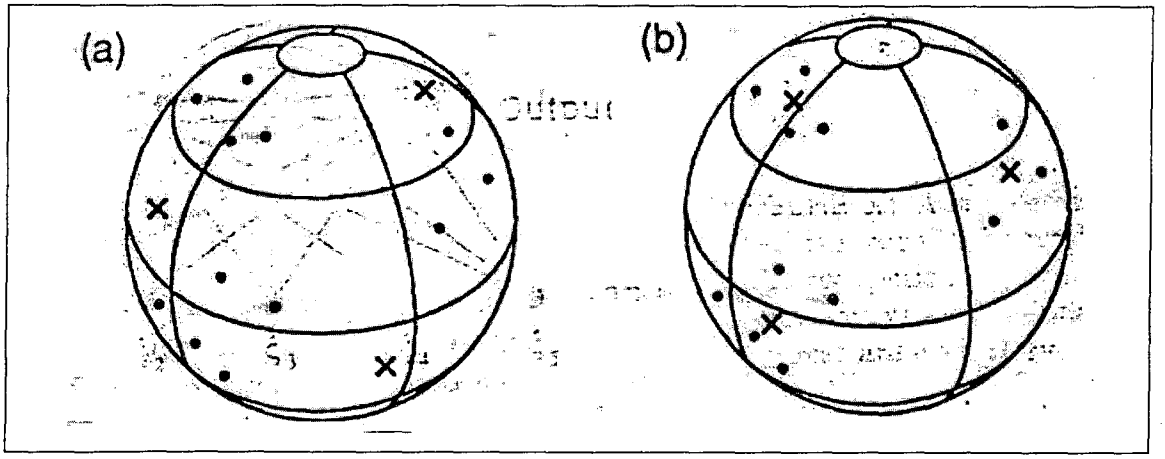

Figura 4.2: Interpretação geométrica do processo de aprendizado competitivo

Neste contexto, cada padrão de entrada é interpretado como um ponto na superfície de uma unidade esférica de dimensão N. A figura 4.2.a indica um estado inicial no qual fica visível a presença de três agrupamentos naturais, constituídos de pontos a representar padrões de entrada. Nesta figura também estão inclusas, representadas através de cruzes, as sinapses referentes às unidades de saídas. A figura 4.2.b mostra um estado final alcançado, onde cada unidade de saída descobriu um agrupamento deslocando suas sinapses para o centro gravitacional dos agrupamentos prévios existentes em 4.2.a.

4.3 TEORIA DA RESSONÂNCIA ADAPTATIVA

Não há garantias de estabilidade das categorias formadas pelo aprendizado competitivo, apresentado na seção 4.2. A apresentação continua de uma seqüência finita e fixa de padrões faz com que as sinapses continuem mudando. Para prevenir este comportamento, uma primeira abordagem consiste em reduzir a taxa de aprendizado η gradualmente a um

valor tendendo a zero, congelando o aprendizado de categorias. Neste caso a rede perde a sua plasticidade, ou seja, a capacidade de reagir à apresentação de novos padrões.

Uma segunda abordagem para prover a estabilidade consiste em decidir sobre o número de unidades de saídas a serem utilizadas. Embora evite a formação de *unidades mortas*, a adoção de um número fixo de unidades de saída pode implicar no esgotamento de unidades disponíveis para classificar um novo padrão. De outra forma, prover um fornecimento inesgotável de unidades de saída pode levar a uma granularização acentuada, onde praticamente não existem agrupamentos.

É neste contexto que se encontra o dilema da *estabilidade-plasticiade*, ou seja, a modelagem de sistemas que aprendam de forma continua sem, no entanto, perder o conhecimento adquirido [51]

Conforme apresentado por HERTZ *et al.* [52], a solução para este dilema consiste em manter um estoque de unidades de saída (finito ou infinito), mas só usá-las quando necessário.

O comportamento das redes *ART*-1 e *ART*-2, desenvolvidas por CARPENTER & GROSSBERG [47, 49], contempla a solução descrita no parágrafo acima. Ambas suprimem o dilema da estabilidade-plasticidade aceitando e adaptando o protótipo de uma categoria (vetor representativo de um agrupamento) apenas quando este é suficientemente similar a um padrão de entrada apresentado.

Neste contexto, o suficiente grau de similaridade é também denominada ressonância. Desta forma, quando um padrão de entrada não é suficiente similar a nenhum protótipo existente então uma nova categoria é formada, combinando o padrão de entrada com um protótipo ainda não utilizado. Quando não restam protótipos para recorrer, então nenhuma resposta é gerada.

Por sua vez, a medida de similaridade depende do parâmetro de vigilância ρ, que obedece ao intervalo $0 < \rho \leq 1$. Se ρ assume um valor muito alto a similaridade se tornará bastante restritiva fazendo com que um grande número de categorias de alta granularidade se formem. Por categorias de alta granularidade deve se entender as categorias pequenas, ou seja, aquelas que acomodam poucos padrões, chegando no pior caso a acomodar um padrão apenas. De outra forma, se ρ assume valores menores a tendência é que ocorra formação de agrupamentos de baixa granularidade. Ou seja, serão formados menos agrupamentos, visto que estes irão acomodar um número maior de padrões, se comparados ao primeiro caso.

4.4 O ALGORITMO *ART*-1

O algoritmo referente à rede *ART*-1 é ilustrativo por excelência. Por ser um algoritmo de características genéricas, sua aplicação pode ser adaptada a outros modelos, dentre os quais o modelo *FUZZY-ART*, descrito na próxima seção.

Neste contexto, os vetores de entrada são representados pelo termo I, enquanto os vetores de protótipos são representados por w_i, ambos constituídos por N posições preenchidas com valores binários. O algoritmo começa com $w_i = 1$, para todo i.

Uma vez que um padrão de entrada é apresentado, a execução do algoritmo segue os seguintes passos:

1. Habilita todas as unidades da camada de saída.
2. Encontra o vencedor i^* entre as unidades de saída habilitadas (Aborta a execução se não existir nenhuma unidade habilitada). O vencedor é definido como aquele que possui o maior valor para T_i, conhecido

como *função de escolha*. Seu calculo é feito da seguinte maneira:

$T_i = \frac{w_i . I}{\alpha + \sum_j w_{ji}}$, onde w_{ji} é o j-ésimo componente de w_i, enquanto α representa um pequeno valor a ser utilizado como critério de desempate, proporcionando a seleção do w_i com os bits mais completos em relação a I. Nesse caso, a tendência é que uma categoria virgem ou pura sempre vença por esse critério adotado.

3. Testa a combinação entre w_i e I computando-se a taxa: $r = \frac{w_i^* . I}{\sum_j I_j}$, onde I_j representa o j-ésimo componente de I. Essa taxa representa a fração de bits de w_i que estão em I e é chamada de *taxa de ressonância*. Se $r \geq \rho$, onde p é o parâmetro de vigilância, então ocorre ressonância, nesse caso vá diretamente para o passo 4. Do contrário, se $r < \rho$, o protótipo w_i^* é rejeitado, a unidade i^* é desabilitada e retorna para o passo 2.
4. Ajusta o vetor vencedor w_i^* removendo dele todo bit que não esteja em I. Esta operação nada mais do que a execução de um *AND* lógico.

A finalização deste algoritmo pode ocorrer de três formas distintas. Se um vetor protótipo, que seja compatível com I, é encontrado então é feito o ajuste presente no passo quatro de maneira que categoria em questão classifique I. Se não existem vetores protótipos disponíveis (habilitados), então um novo vetor protótipo no estado puro é selecionado e então o ajuste do passo quatro é executado, fazendo com que o protótipo fique exatamente igual a I. Esse caso corresponde à criação de uma nova categoria para classificar

especificamente o padrão I. Mas caso não exista mais nenhum protótipo adaptável bem como nenhum protótipo puro presente em uma reserva, então o sistema não gera nenhuma resposta, o que poderia significar que o algoritmo é simplesmente abortado, visto que todas as unidades da camada de saída estão desabilitadas.

A seqüência de passos demonstra como *ART* lida com o dilema da *plasticidade-estabilidade*. Enquanto houverem unidades habilitadas na camada de saída, o algoritmo permanece estável. Este também apresenta estabilidade, pois uma análise mais detalhada mostra que a mudanças dos pesos cessa após um número finito de apresentações de um número fixo de vetores ou padrões de entrada. A estabilidade advém do fato da regra de adaptação, presente no passo quatro, permitir que um vetor protótipo apenas perca bits, nunca ganhe. Dessa forma um dado protótipo nunca volta para a um estado ou configuração anterior.

É possível notar que o laço iterativo existente do passo três para o passo dois constitui uma busca através do conjunto de vetores protótipos disponíveis. Levando em consideração o máximo valor assumido pelo critério *T*, o que se busca seqüencialmente é o protótipo mais próximo, o seguinte mais próximo e assim por diante, até que se encontre um protótipo que satisfaça o critério $r \geq \rho$ Esse comportamento serve para ilustrar as diferenças sutis que existem entre esse dois critérios..

O primeiro mensura a fração de bits de um protótipo w_i que estejam contidos em I, enquanto que o critério *r* representa a fração de bits de I contidos em w_i. Essa busca é relativamente lenta quando comparada a outros algoritmos tradicionais, mas ela só ocorre antes da estabilidade ser alcançada para um certo conjunto de entrada.

CARPENTER & GROSSBERG [47] desenvolveram o modelo arquitetural ART-1 utilizando módulos funcionais baseados em suposições biológicas aceitáveis. A seleção de um vencedor, a camada de entrada, a mudança de pesos e o mecanismo de habilitar/desabilitar são descritos por circuitos governados por equações diferenciais.

HERTZ *et at.* [52] apresenta uma versão rudimentar desse circuito, mostrado na figura 4.2. Nele existem duas camadas: a camada de entrada com suas unidades V_j e a camada de saída com as unidades O_i (função de saída). As duas camadas se apresentam ligadas em ambas direções através de w_j (conexões bottom-up) e w_i (conexões top-down). As conexões bottom-up são cópias das conexões top-down atendendo à seguinte normalização:

$$w_i = \frac{w_j}{\alpha + \sum_j w_{ij}}, \text{ onde } w_{ij} \text{ representa o i-ésimo elemento de } w_j.$$

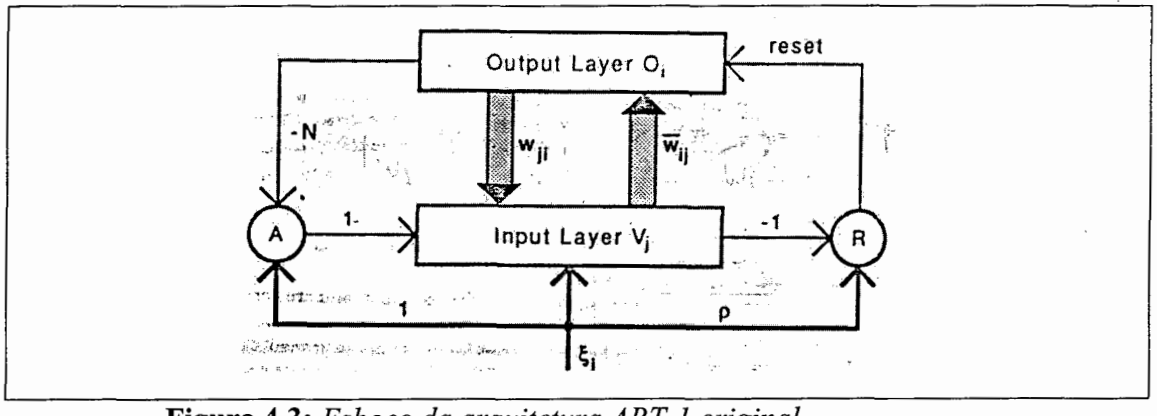

Figura 4.3: *Esboço da arquitetura ART-1 original.*

A camada de saída consiste de uma estrutura winner-take-all, onde apenas a unidade com o maior net entre as unidades habilitadas é que apresenta sua função de saída com valor unitário($O_i = 1$), representando assim uma unidade vencedora. Se o sinal de

"reset" em R é ligado enquanto uma unidade vencedora está ativa, essa unidade é removida de competições futuras. Todas unidades podem ser reabilitadas por um sinal que não é mostrado aqui.

As unidades da camada de entrada podem ser designadas de duas formas:

1. $V_j = I_j$, se todas as unidades de saída apresentarem $O_i = 0$
2. $V_j = I_j \cap \sum_j w_{ij}.O_i$, do contrário. Sendo w_{ij} o i-ésimo elemento de w_j e "$\cap$" um operador AND binário.

Em ambos os casos, a atribuição de valores é executada unidade auxiliar de processamento A. Já o sinal de desabilitar ou "reset" é gerado pela unidade processamento R.

A atualização dos pesos da conexão top-down é feita de acordo com a regra $\Delta w_{ij} = \eta O_i (V_j - w_{ij})$, enquanto que o protótipo correspondente ao vencedor i^* se torna exatamente igual ao input V_j após a ressonância.

4.5 O ALGORITMO *FUZZY-ART*

FUZZY-ART é um algoritmo para formação de agrupamentos que ao contrario do modelo *ART*-1, opera sobre vetores de valores analógicos [53].

Todas a suas aplicações costumam resultar na formação de muitos agrupamentos num grande espaço dimensional e poderiam ser beneficiadas por uma implementação paralela, visando atender aplicações de tempo real. Muitas propriedades do algoritmo *FUZZY-ART* favorecem a sua implementação em hardware, visto que é necessária pouca área de circuito destinada a cada sinapse.

Entretanto, como proposto em CARPENTER *et al* [54], o modelo original *FUZZY-ART* requer sinapses bidirecionais, transporte de peso, duplicação de pesos, detalhes que tornam problemática a sua implementação paralela.

BLUME & ESENER [55] apresentam uma implementação do algoritmo *FUZZY-ART* utilizando uma arquitetura que elimina a necessidade dos requerimentos citados. Essa implementação implica na adoção de uma seqüência diferente de operações para o algoritmo original, porém a classificação de um conjunto de vetores de entrada permaneceria a mesma.

Conforme CARPENTER *et al* [54], o modelo *FUZZY-ART* agrupa vetores levando em consideração os critérios: *match* e *choice*, que na seção 4.3, referente ao algoritmo ART-1, correspondiam à taxa de ressonância e ao critério de escolha de uma unidade na camada de saída. Ambos foram modificados através da substituição do operador AND binário pelo seu equivalente adotado na teoria dos conjuntos difusos.

Dado um vetor de entrada I e um protótipo w_j, a taxa de ressonância é calculada como:

$$r = \frac{|I \wedge w_j|}{|I|}, \text{ onde } w_j \text{ é um vetor de dados analógicos}$$

associado ao agrupamento j. Por sua vez, o símbolo "$\wedge$"representa o operador AND fuzzy, de maneira que $(p \wedge q) \equiv \min(p,q)$, e a norma $|...|$ é definida como $|p| \equiv \sum_i |p_i|$.

Por sua vez, o critério ou função de escolha é definido por:

$$T = \frac{|I \wedge w_j|}{\alpha + |w_j|}, \text{ onde } \alpha \text{ é uma constante de pequeno valor a}$$

ser utilizada como critério de desempate entre duas unidades da camada de saída. Cada

vetor de entrada apresentado é atribuído à categoria que maximize contanto que a condição $r \geq \rho$ seja satisfeita, sendo ρ o parâmetro de vigilância presente no intervalo $0 \leq \rho \leq 1$.

CARPENTER *et al.* [54] propõe a busca por uma categoria j que maximize o critério de escolha T e cuja taxa de ressonância seja maior que ρ. Se uma categoria j é escolhida e sua taxa de ressonância não exceder ρ, então esta é prontamente desabilitada (*reset*) e a busca prossegue até que se encontre uma categoria que gere a ressonância. A rede neuronal descrita por CARPENTER *et al* [54, 56] é também um modelo arquitetural baseado em três camadas, ilustrados na figura 4.3. O cálculo da função de escolha é executado camada F2. Por sua vez, o cálculo da taxa de ressonância é feito na camada F1 juntamente com o nodo *reset*. Por serem necessários em ambos cálculos, cópias dos pesos, w_{ji}, são armazenadas e atualizadas de maneira independente pelas camadas F1 e F2. O custo de tempo que existe entre a apresentação de um vetor de entrada e a sua atribuição a uma categoria é variável, dependendo de quantos ciclos de busca sejam necessários.

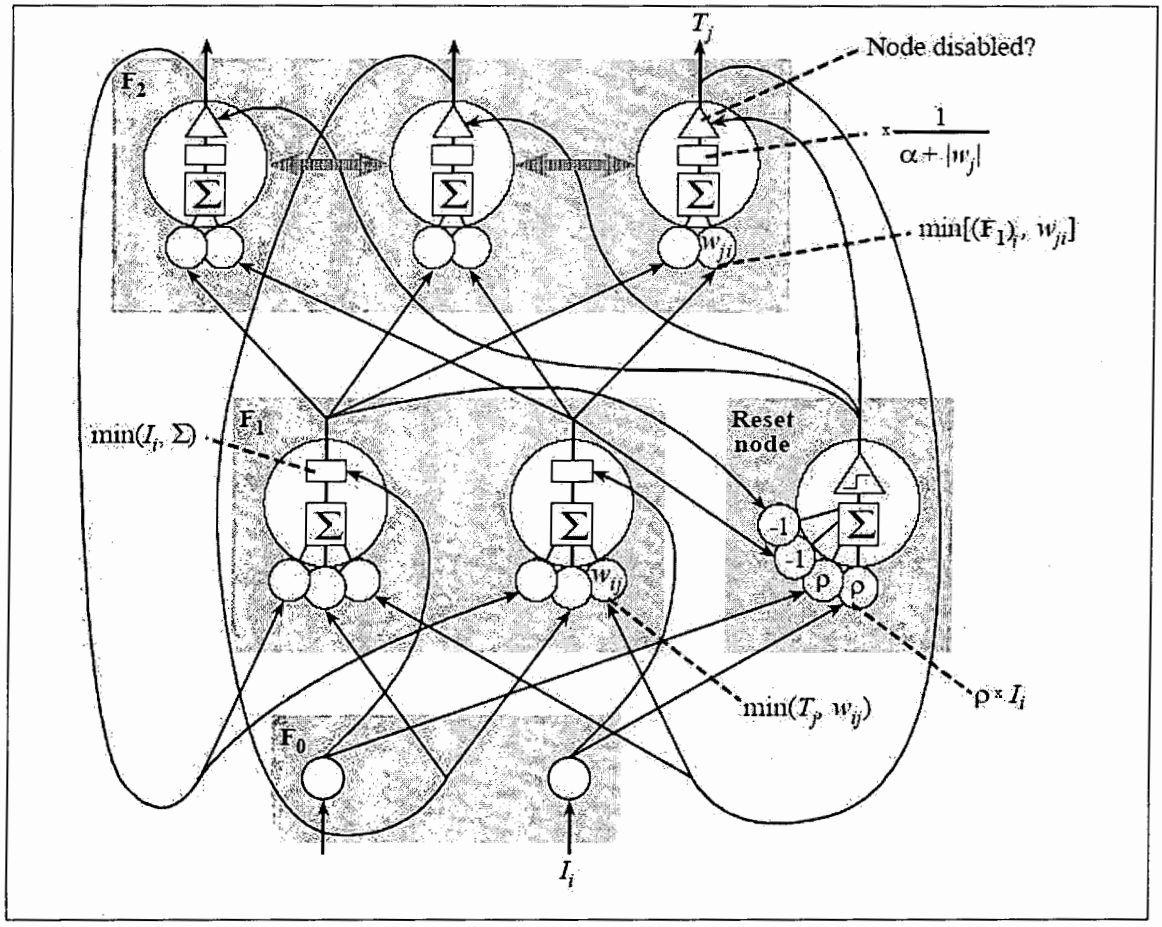

Figura 4.4: Arquitetura *ART* originalmente proposta.

Uma abordagem alternativa é proposta em BLUME & ESENER [55], onde, para cada protótipo referente a uma categoria ou unidade de saída habilitada, a função de escolha e a taxa de ressonância são calculadas ao mesmo tempo. Neste caso, são comparadas pela função de escolha apenas as categorias que geram ressonância. A figura 4.4 apresenta a arquitetura baseada nessa ordem de operações. A função de escolha e a taxa de ressonância são computadas na mesma camada e dessa forma não é necessário que exista uma redundância de pesos. Ou seja, apenas um conjunto de pesos é armazenado e atualizado. Como as duas operações são feitas em paralelo, a execução do algoritmo, em cada unidade de saída, leva um tempo constante para qualquer vetor de entrada que seja apresentado á rede.

Além do tempo constante, outra mudança que a paralelização impõe é a eliminação do nodo de controle de sinais de desativação (*reset*). A eliminação desse componente facilita uma implementação em hardware que se utilize técnicas de *pipelining*.

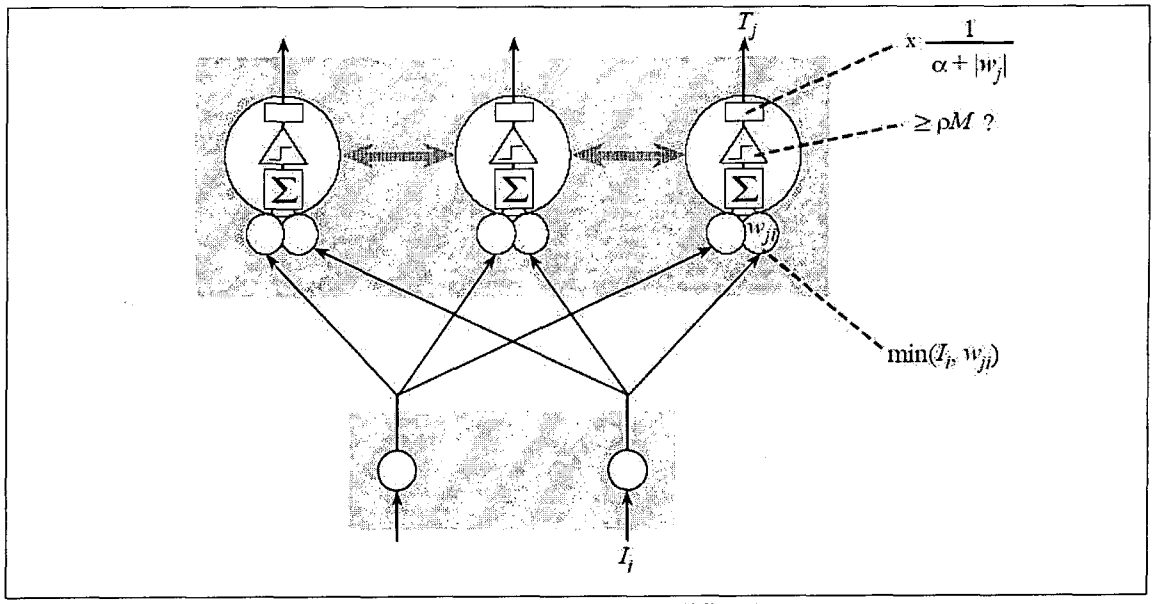

Figura 4.5: Arquitetura *ART* simplificada.

Na arquitetura simplificada, presente na figura 4.5, toda unidade de processamento deve computar a taxa de ressonância. Felizmente, esse processamento não implica na necessidade de um esforço computacional maior. A relação $r \geq \rho$ é equivalente a $|I \wedge w_j| \geq \rho|I|$. A quantidade presente no lado esquerdo da equação é necessária para calcular T_j (função de escolha) em qualquer caso, e sendo assim não há necessidade de nenhum cálculo adicional.

A regra de aprendizado, ou atualização dos pesos, é dada por:

$$w_{Ji}^{novo} = w_{Ji}^{velho} \wedge I_i$$

cabendo ressaltar que, da mesma forma como ocorre no algoritmo ART-1, apenas os pesos referentes à categoria que classificou o vetor de entrada I são atualizadas.

A versão simplificada vista nesta seção preserva do modelo *FUZZY-ART* original as operações simples, enquanto remove os controles de sinais e a replicação de pesos. Segundo BLUME & ESENER [55], a classificação de vetores de entrada se mantém inalterada em comparação ao modelo arquitetural original.

Capítulo 5 - Minerando a *Web* com ART

5.1 INTRODUÇÃO

Na subseção 5.2 deste capítulo será descrito o modelo de mineração adotado. Este modelo é fundamentado no processo de extração de propriedades de hipertexto, que permite converter os documentos hipertexto em vetores de freqüência, representação adotada como padrão pelos algoritmos de formação de agrupamentos. Na subseção 5.3 serão descritos os experimentos executados. Os resultados serão expressos através de *a*grupamentos de documentos obtidos a partir de uma rede neuronal *FUZZY-ART*. Estes agrupamentos serão visualizados através de uma ferramenta gráfica.

5.2 EXTRAÇÃO DE PROPRIEDADES DE HIPERTEXTO

Segundo AAS & EIKVIL [30] o primeiro passo para realizar mineração de documentos consiste na transformação de documentos, conjunto de caracteres, em uma representação apropriada para algoritmos de aprendizagem e classificação. Nesse contexto, a tarefa de extração de propriedades de hipertexto compreende os seguintes fundamentos: extração de *tags, e*liminação de palavras vazias e extração de raízes, conforme serão melhor descritos nas seções seguintes.

Também serão examinados detalhadamente neste capitulo o processo de formação de um dicionário de raízes, bem como a construção dos vetores de freqüência.

5.2.1 EXTRAÇÃO DE *TAGS*

A extração de *Tags* corresponde a uma tarefa de análise sintática responsável por separar *tags* das palavras contidas em um documento. Entretanto, para esta separação é necessário que ao abrir o documento cada símbolo seja lido e verificado:

- Símbolo iniciado e finalizado pelos caracteres "<" e ">", significa que este representa de uma *tag HTML* (Como por exemplo, o símbolo <BODY>).
- Símbolo iniciado e finalizado pelos caracteres "&" e ";" indica uma palavra reservada da linguagem *HTML* (Como por exemplo " ").

Caso não atenda a nenhuma das duas possibilidades citadas acima, significa que o símbolo em questão é uma palavra.

5.2.2 ELIMINAÇÃO DE PALAVRAS VAZIAS

Com a remoção das *tags*, obtém-se apenas as palavras. No entanto, nem todas as palavras podem ser aproveitadas, considerando que algumas delas podem ser classificadas como vazias, por não agregar valor algum. Desta forma, palavras que não apresentam informações relevantes, são eliminadas. Dentro desta categoria podem ser incluídos: os pronomes, as conjunções, as preposições e os artigos.

Cada idioma possui suas regras gramaticais específicas, tornando-se possível a identificação das palavras vazias que se enquadram nos exemplos anteriormente citados. O problema é que alguns nomes próprios podem ser considerados como palavras vazias, pela característica de não acrescentarem informação relevante, com base na hipótese de que quanto mais vezes uma palavra ocorrer através de todos os documentos de uma coleção, mais irrelevante ela se torna [30].

5.2.3 EXTRAÇÃO DE RAÍZES

O processo de extração de raízes consiste no agrupamento de formas variantes de uma palavra em uma representação comum: a raiz, ou radical. Neste contexto, as palavras oriundas da língua inglesa: *presentation*, *presented*, *presenting* são resumidas ao radical *present*. Este tipo de procedimento é empregado com freqüência em sistemas de recuperação de informação (IR), pois estes se baseiam na hipótese de que uma consulta com a palavra *presenting* implica no interesse por documentos contendo as palavras *presentation* e *presented* [57].

Em sistemas de descoberta de informação a objetivo de se empregar esse processo é promover a redução da dimensionalidade dos vetores [58].

Segundo NASCIMENTO & CUNHA [59], existem três técnicas para extração de raízes:

- **N-Gramas:** Essa técnica não produz uma raiz, entretanto, computa o quanto duas palavras são similares.

- **Remoção de Afixos:** Este tipo de técnica se baseia no comportamento da estrutura sintática de palavras para um determinado idioma. O exemplo mais representativo desta técnica é o algoritmo de *PORTER* [60]. Existem versões deste algoritmo para diferentes idiomas, sendo a sua aplicação para a língua portuguesa relatada em ORENGO [57]. Os erros associados ao processo de extração raízes podem ser divididos em 2 grupos:

- *Overstemming*: Ocorre quando a cadeia (*string*) removida não era um sufixo, mas parte de um radical. Isso pode resultar num agrupamento de palavras semanticamente distintas.
- *Understemming*: Ocorre quando um sufixo não é removido, levando à criação de agrupamentos ambíguos, que poderiam ser grupamentos mais genéricos.

- **Variedade de Sucessores:** Neste tipo de técnica reside o algoritmo **PICO&PLATEAU**, que se baseia na construção de uma árvore de *tries* [59].

5.2.4 FORMAÇÃO DE UM DICIONÁRIO

A extração de propriedades de hipertexto (extração de *tags*, eliminação de palavras vazias e extração de raízes) aplicada sistematicamente sobre cada documento de uma coleção, produz como resultado um dicionário, que corresponde a coleção de todas as raízes distintas recuperadas.

5.2.4.1 Modelo de representação de documentos

Uma vez definido um dicionário, torna-se necessário adotar um modelo de representação de documentos. O modelo mais usual corresponde ao *modelo de espaço vetorial* [19]. Neste modelo, documentos são representados por vetores de raízes. Posto dessa forma, uma coleção de documentos é representada por uma matriz *raiz-por-documento* **"A"**, onde cada linha corresponde à ocorrência de uma raiz em um certo documento,

$$A = (a_{ik}),$$

onde a_{ik} é o número de ocorrências da raiz *i* em um documento *k*.

Pelo fato de nem todas as raízess ocorrerem em cada documento, usualmente a tendência é que a matriz "**A**" se torne esparsa. O número de linhas, *M*, da matriz corresponde à quantidade total de raízes presentes no dicionário e este número pode ser muito elevado.

Um dos problemas cruciais referentes à mineração de documentos reside justamente na alta dimensionalidade do dicionário obtido, isso torna necessário à adoção de medidas para reduzir a dimensionalidade do espaço de busca.

5.2.4.2 Redução de dimensionalidade

Cada raíz presente no dicionário representa uma dimensão no espaço de atributos de maneira que, quanto maior o número de raízes maior será o comprimento dos vetores. É necessário que o dicionário obtido apresente um tamanho aceitável, de maneira que não comprometa o processo de formação de agrupamentos. Uma vez que esta condição não é atendida, torna-se necessário aplicar métodos de seleção de atributos para preservar no dicionário apenas as raízes relevantes.

Vários métodos foram avaliados em YANG & PEDERSEN [61], dentre os quais o *limiar de freqüência de documento* (DF). Neste caso, freqüência de documento corresponde ao número de documentos no qual uma raiz ocorre. Sendo assim, inicialmente este método contabiliza a freqüência de documentos para cada raiz presente no dicionário. Uma vez computado número de ocorrências é necessário que se estipule um limiar de separação, expresso em porcentagem e variando num intervalo de 0 a 100. Dessa forma, será descartada da composição de um dicionário toda e qualquer raiz cuja freqüência de ocorrência esteja abaixo do limiar adotado. Para exemplificar seu funcionamento, dado uma

coleção composta por 100 documentos, a aplicação de um limiar 10% implicará na eliminação de raízes que apareçam em menos 10 documentos.

Segundo AAS & EIKVIL [30], DF se baseia na suposição de que termos raros não representam um ganho de informação ou não influenciam no desempenho global de sistemas que possuem como objetivo categorizar dados ou encontrar agrupamentos.

Apesar de sua simplicidade, JOACHIMS [62] advoga que, dependendo do limiar estipulado como parâmetro, o método produz uma eliminação de raízes muito agressiva. Essa preocupação se justifica pelo fato de que em algumas abordagens é mais interessante preservar as raízes raras.

Um segundo método, chamado *limiar de freqüência relativa* (RF), é proposto para atuar como uma ferramenta auxiliar à heurística DF e se restringe ao escopo referente a um documento. O objetivo deste método é preserva em um documento apenas o seu núcleo semântico.

Para um documento, existe a raíz que ocorre o maior número de vezes bem como a raiz que ocorre o menor número de vezes. Esses números de ocorrências representam respectivamente: a maior freqüência interna e a menor freqüência interna.

A freqüência relativa é calculada levando em consideração a maior e a menor freqüências encontradas bem como um terceiro parâmetro, chamado limiar de freqüência relativa, que varia num intervalo de zero a um. O objetivo é remover as raízes que tenham baixo valor semântico para o domínio de conhecimento no qual um documento se encontra contextualizado, evitando que raízes de baixo valor semântico sejam porventura consideradas pela análise de freqüência de documento. Para cada raíz contida em um certo documento, o seu valor semântico é mensurado por meio da comparação de sua freqüência (seu número de ocorrências) em relação à freqüência relativa.

O valor da freqüência relativa, para um dado documento, é calculado da seguinte forma:

$f_{rel} = (f_{\max} - f_{\min})\alpha + f_{\min}$, onde:

- $f_{\max}$: Representa o número de vezes que a raíz mais freqüente aparece no dado documento.
- $f_{\max}$: Representa o número de vezes que a raíz menos freqüente ocorre no dado documento.
- α : Limiar de freqüência relativa, cujo *range* varia de [0 – 1], informado como parâmetro, sendo o mesmo para todos os documentos de um conjunto usado como experimento.

O funcionamento deste método é bastante simples, uma vez calculada a freqüência relativa é feita a seguinte comparação:

Para cada raíz analisada em um documento:

Se $N^{o}ocorrências \langle f_{rel}$ **Então**.

Excluí raíz do documento

Senão

Preserve a raíz analisada no documento

Fim Se

Figura 5.2: Lógica da heurística de freqüência relativa

A virtude desse método é a sua simplicidade de aplicação, além de proporcionar uma considerável melhoria no processo seleção por DF. Sua desvantagem é a

forte dependência em relação ao limiar informado como parâmetro, visto que este assume o mesmo valor para todos os documentos de uma coleção.

5.3 FORMAÇÃO DE VETORES

Nas subseções anteriores, foram examinados os fundamentos necessários à transformação de documentos hipertexto em uma representação adequada que possibilite a aplicação de algoritmos de mineração de dados. Todavia, torna-se necessário descrever em termos práticos os pormenores envolvidos no processo de formação de vetores.

Após a eliminação de *tags* e palavras vazias, bem como a extração de raízes, um documento hipertexto passa pela transformação ilustrada na figura 5.3:

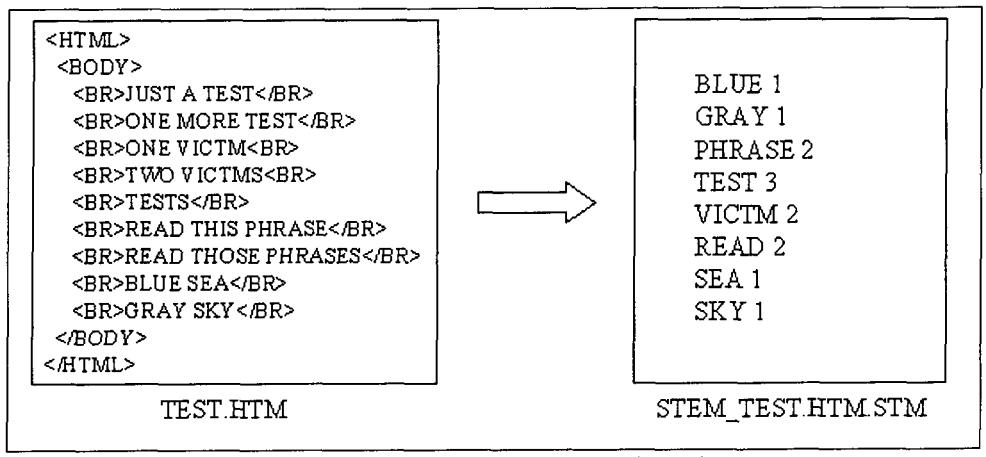

Figura 5.3: Transformação de um arquivo hipertexto.

O documento nomeado TESTE.HTML origina um arquivo simplificado, contendo apenas as raízes juntamente com suas respectivas freqüências de ocorrência. Experimentalmente esse tipo de arquivo é classificado como arquivo "raíz – freqüência interna" (SF) e sua nomenclatura obedece ao seguinte critério:

$$STEMS\_ + Nome\ Hipertexto + .STM$$

Dessa forma, o processo ilustrado no paragráfo acima deve ser estendido a todos os documentos hipertexto contidos em um dado conjunto, originando assim uma coleção de arquivos SF, como ilustrado na figura 5.4. A ilustração do aplicativo responsável pela obtenção dos arquivos SF encontra-se na seção A1 do apêndice.

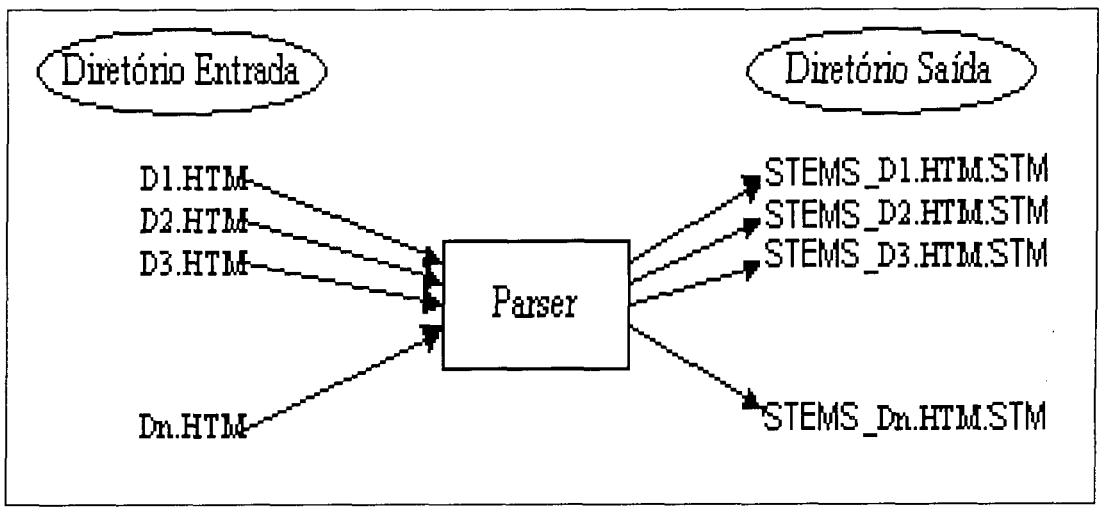

Figura 5.4: Obtenção de uma coleção de arquivos SF.

O dicionário de raízes será montado a partir da coleção de arquivos SF obtidos. Para tal, torna-se necessário combinar a aplicação da heurística RF juntamente com a heurística DF. Como visto anteriormente, RF permite que, em cada arquivo SF, somente as raízes semanticamente importantes sejam preservadas em seu conteúdo e por conseguinte participem do processo de construção do dicionário. As raízes não relevantes são prontamente excluídas. A aplicação da heurística de frequência relativa RF é ilustrada na figura 5.5. A ferramenta que executa esta tarefa encontra-se ilustrada na seção A2 do apêndice.

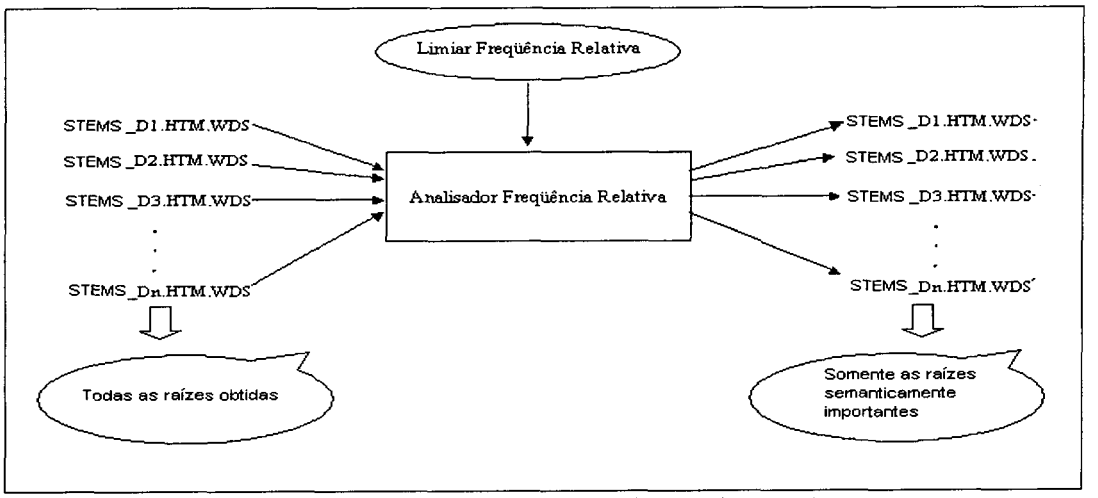

Figura 5.5: Análise de freqüência relativa (RF)

Uma vez que todos os arquivos SF contenham apenas raízes relevantes, a heurística DF é aplicada para construir o dicionário de raízes. Ao contrário da versão original descrita na subseção 5.2.4.2, são aplicados dois pontos de corte. Desse modo, toda raiz cuja freqüência de ocorrência em documentos esteja dentro do intervalo, representado pelos limiares de freqüência mínima e máxima, será selecionada para compor o dicionário. A aplicaçao da heurística DF é ilustrada na figura 5.6. A interface da aplicação que executa esta tarefa encontra-se na seção A3 do apêndice.

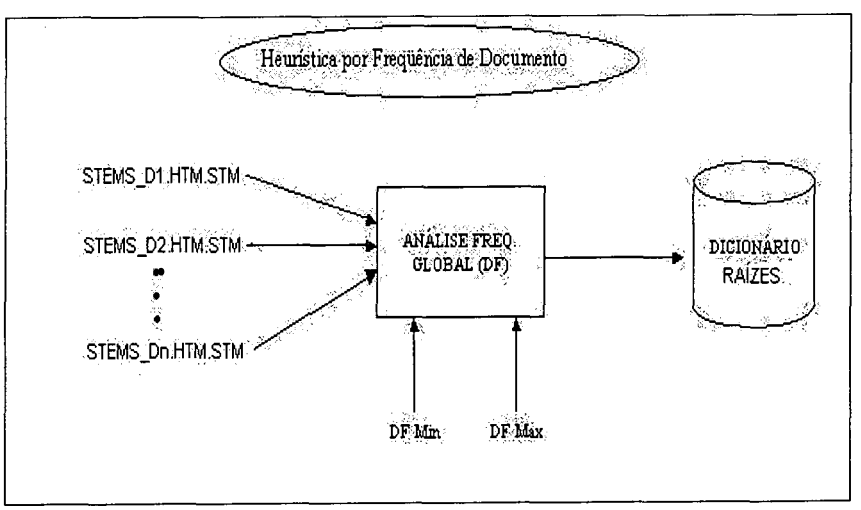

Figura 5.6: Análise de Freqüência de documento

Uma vez montado o dicionário de raízes, os vetores de freqüências podem ser obtidos. Para tal, o algoritmo ilustrado na figura 5.7 deve ser aplicado sobre cada arquivo SF presente em uma coleção.

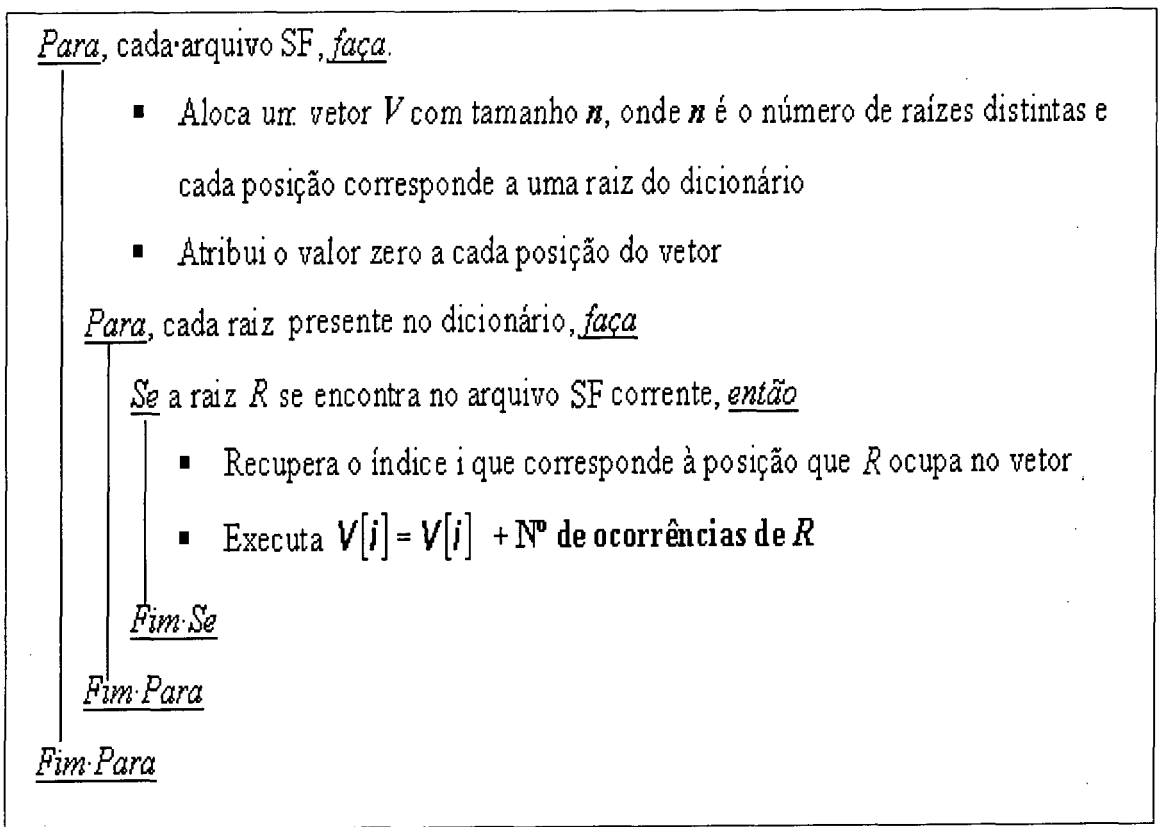

Figura 5.7: Algoritmo para construção dos vetores de freqüência.

A implementação do algoritmo responsável pela construção dos vetores de frequência, tendo como parâmetros uma coleção de arquivos SF e o dicionário de raízes, encontra-se na seção A4 do apêndice.

O processo de obtenção de um vetor a partir de um documento nomeado *Dm* é exibido na figura 5.8.

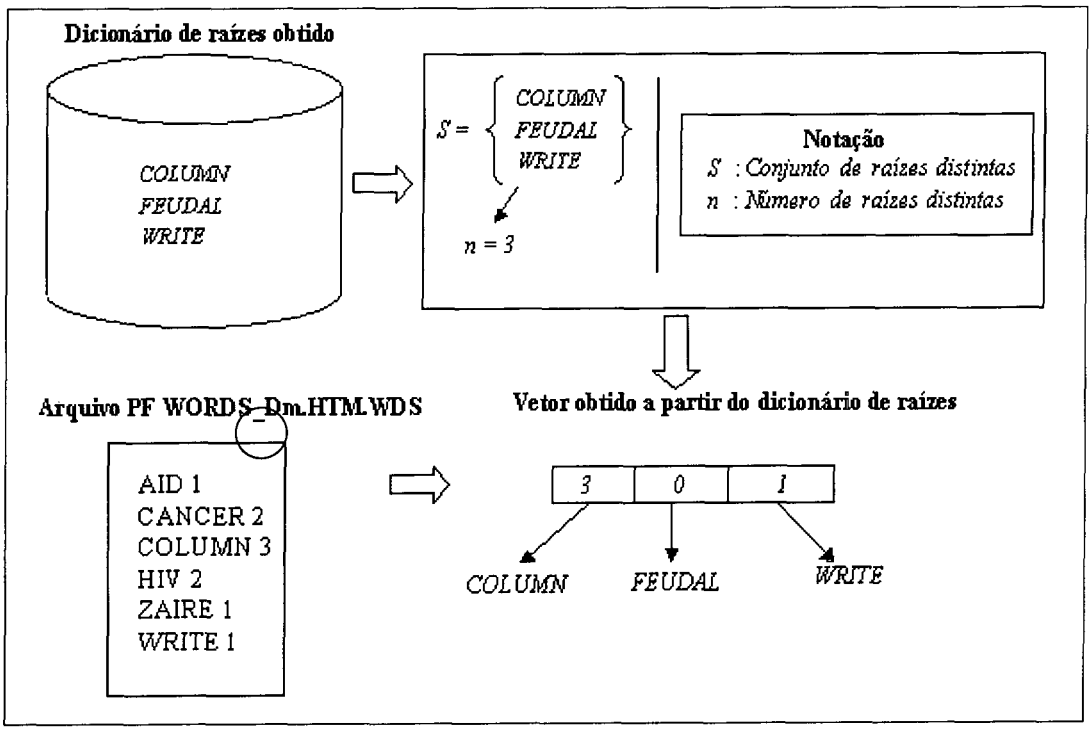

Figura 5.8: Obtenção de um vetor de freqüência para o documento *Dm*.

Cada vetor obtido é armazenado num arquivo bruto, cujo conteúdo representa uma matriz, onde cada linha corresponde a um vetor de freqüência referente a um documento hipertexto. Por sua vez, as colunas R0, R1, R2, R3, R4 e R5 representam as freqüências atribuídas às raízes presentes no dicionário.

Arquivo	Freqüências					
	R0	R1	R2	R3	R4	R5
Aids1.htm	36	1	1	0	0	0
Aids2.htm	16	0	1	0	0	0
Datamining1.htm	0	0	0	0	0	20
Datamining2.htm	0	0	0	0	0	4
Datamining3.htm	0	0	0	0	0	12
Marxism1.htm	0	0	0	5	3	0
Marxism2.htm	0	0	0	12	18	0

Tabela 5.1: Matriz de freqüências obtida.

Entretanto, para executar uma rede *FUZZY-ART* torna-se necessário fazer a normalização da matriz de freqüências presente na figura 5.10. Este processo visa modificar a escala de valores para o intervalo [0, 1]. A implementação do algoritmo responsável pela normalização da matriz de frequências encontra-se na seção A4 do apêndice.

5.4 RESULTADOS

Uma vez obtida a matriz normalizada, a rede *FUZZY-ART* é então executada. A rede neuronal em questão foi implementada em *JAVA* e corresponde a uma aplicação *console*, desprovida de qualquer interface gráfica. A código fonte desta aplicação encontra-se no seção A5 do apêndice.

Os resultados gerados pela rede são armazenados num arquivo texto. Cada linha presente no conteúdo deste arquivo corresponde à relação de cada documento para o respectivo agrupamento que o classificou. Esta relação encontra-se presente na figura 5.9.

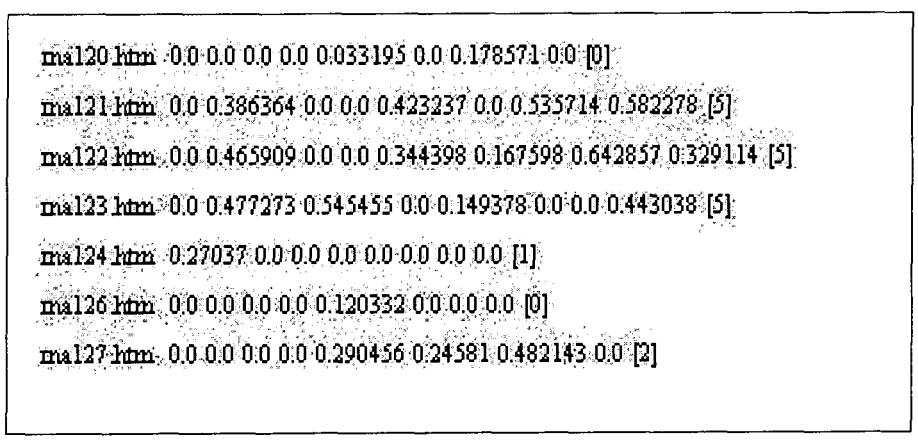

Figura 5.9: Resultados gerados pela rede *FUZZY-ART*.

Para proporcionar uma visão mais clara acerca dos resultados obtidos em uma simulação, foi desenvolvida uma aplicação gráfica que permite visualizar os agrupamentos

encontrados, bem como navegar através dos documentos. Esta aplicação é baseada na *api* gráfica *JAVA / SWING* e a sua interface encontra-se na seção A6 do apêndice.

Nesta ferramenta, cada agrupamento *é* representado por um retângulo que assume a configuração de uma régua e os documentos classificados são representados como pontos desta régua. A cor de cada agrupamento é gerada de forma randômica. Por sua vez, a cor dos documentos é representada por um *dégradé* construído a partir da cor definida para seu agrupamento é proporcional a distancia em relação ao centróide.

5.4.1 CONJUNTO DE DADOS SIMULADOS

Os experimentos foram executados sobre documentos hipertextos recuperados a partir do mecanismo de busca *google.* Estes documentos encontram-se escritos na língua inglesa e foram separados em quatro bases textuais, cada qual contendo cem documentos. Essa separação teve como critério o domínio de conhecimento nos quais os documentos se encontravam:

1. AIDS: Contempla cem documentos que tem como tema *Aids.*
2. CANCER: Contempla cem documentos que tem como tema Câncer
3. RNA: Contempla cem documentos que tem como tema Redes Neuronais Artificiais.
4. MARX: Contempla cem documentos que tem como tema Karl Marx.

Nas subseções 5.4.1.1 a 5.4.1.4 são apresentados os resultado obtidos para os experimentos aplicados sobre cada uma das quatro bases em particular. Por sua vez, na subseção 5.3.1.5, são apresentados os resultados obtidos para os experimentos feitos sobre uma base textual que agrega os quatro grupos de documentos citados.

Em cada experimento foi adotado um conjunto distinto de valores para os parâmetros α, DFMin e DFMax. De forma impirica foi observado que esses valores proporcionavam a obtenção de dicionários compostos pelas raízes mais relevantes e discriminantes possíveis. Os parâmetros são ilustrados na tabela 5.2.

Base de dados	α	DFMin	DFMax	Total de Raízes
AIDS	0.1	4%	90%	151
CANCER	0.1	2%	99%	232
RNA	0.1	5%	93%	453
MARX	0.1	3%	99%	256
ACMR	0.01	22%	30%	31

Tabela 5.2: Parâmetros adotados para os experimentos.

5.4.1.1 Base de dados AIDS

A execução da rede *FUZZY-ART*, resultou em 9 agrupamentos, ilustrados na figura 5.10.

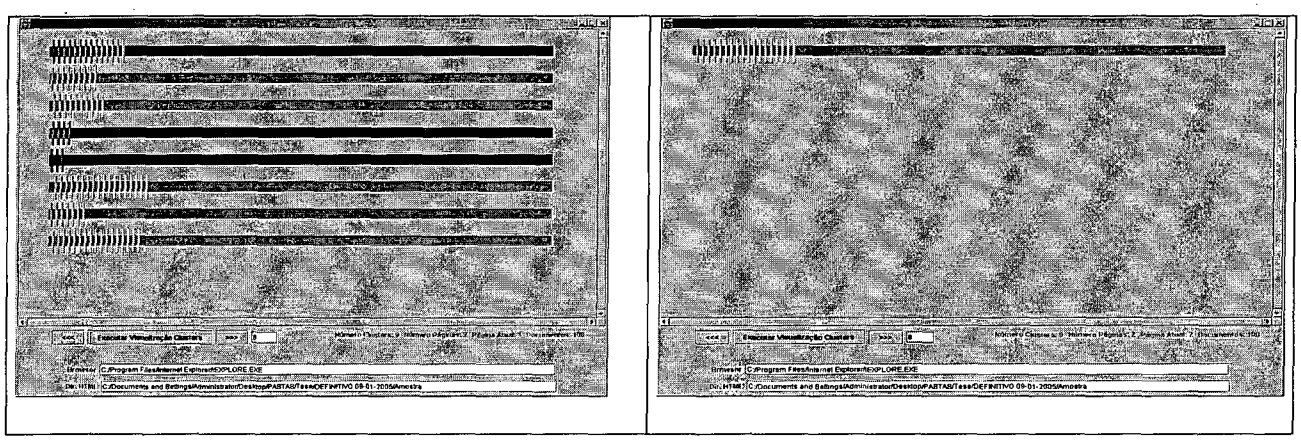

Figura 5.10: Visualização dos Agrupamentos obtidos para base AIDS.

Agrupamento	Documentos	Raízes Comuns
1	14	PEOPL
2	9	BRAZIL
3	10	EPIDEM
4	4	CONFER – ONLIN
5	3	CHILDREN
6	18	VIRUS
7	7	HEALTH
8	17	NATION
9	18	INFECT

Tabela 5.3: Descrição dos agrupamentos obtidos para base AIDS.

No contexto deste experimento, os agrupamentos N°2, N°4 e N°5 demonstram características muito interessantes.

O agrupamento N°2 agregou nove documentos que tem como similaridade a raiz BRAZIL. Neste caso BRAZIL acaba por ser um rótulo perfeito para este agrupamento, uma vez que todos os seus documentos abordam o Brasil em seus respectivos títulos. Esses documentos tratam sobre o programa de combate à aids do governo brasileiro e à iniciativa da quebra de patentes.

Com relação ao agrupamento N°4, este foi constituído por quatro documentos que tem como similaridades as raízes CONFER e ONLIN. São documentos que se referem a conferências internacionais para discutir sobre políticas de combate a propagação da doença.

O agrupamento Nº5 englobou três documentos que tem em comum a raiz CHILDREN, pois tratam sobre a relação existente entre a *aids* e as crianças. São documentos que abordam em seus títulos a situação de crianças órfãs na África, o preconceito enfrentado nas escolas por crianças soro-positivas e a contaminação por leite-materno.

Por sua vez, os agrupamentos restantes são formados por documentos cujas similaridades apenas se baseiam nas raízes comuns, ilustradas na tabela 5.3. Essas raízes, entretanto, não refletem exatamente os temas abordados pelos documentos em questão.

5.4.1.2 Base de dados CANCER

A execução da rede *FUZZY-ART*, proporcionou a obtenção de 9 agrupamentos, ilustrados na figura 5.11.

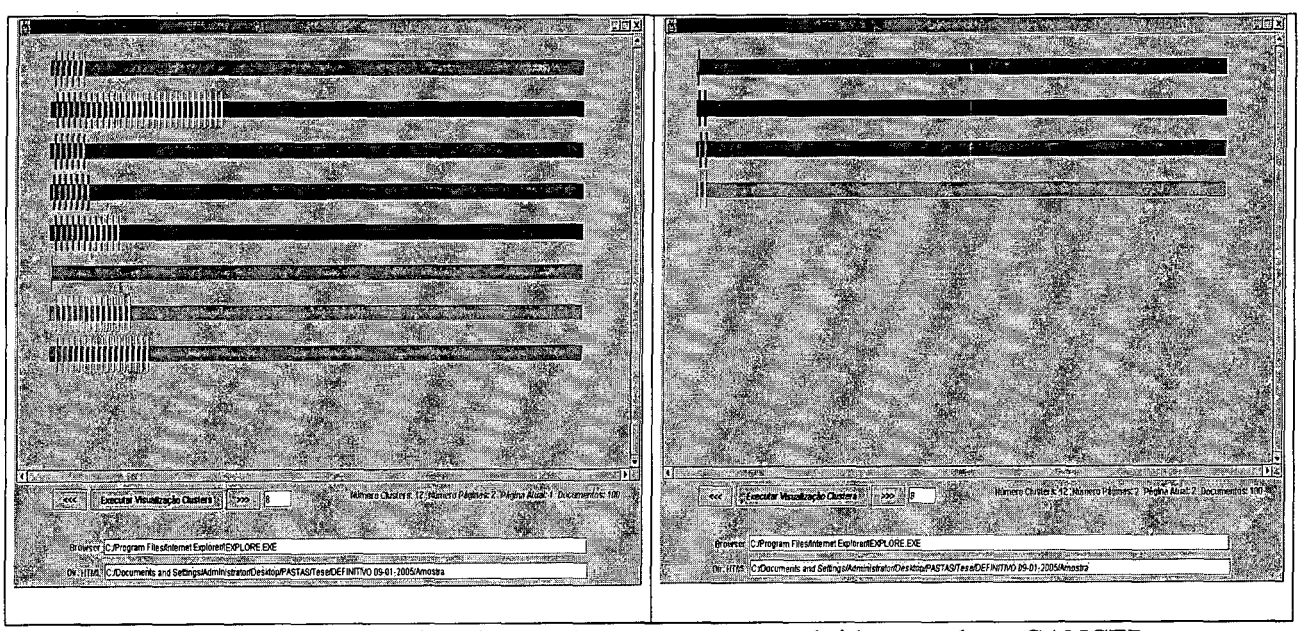

Figura 5.11: Visualização dos agrupamentos obtidos para base CANCER.

Agrupamento	Documentos	Raízes Comuns
1	6	CELL
2	30	TREATMENT
3	6	BONE
4	7	BREAST
5	12	CANCER
6	1	BLOOD – COLON – NODE – TEST
7	14	TYPE
8	17	RISK
9	1	AGENC – INFO – INFORM – PAGE
10	2	NATION – OVARIAN
11	2	PATIENT
12	2	FATIGUE

Tabela 5.4: Descrição dos agrupamentos obtidos para base CANCER.

Para este experimento destacam-se os agrupamentos N°1, N°3 N°4, N°5, N°8 e, N°10.

O agrupamento N°1 contemplou seis documentos que tem como principal similaridade a raiz CELL, uma vez que estes abordam o desenvolvimento de células cancerígenas e o crescimento desordenado de células brancas.

Por sua vez, o agrupamento N°3 agregou seis documentos, sendo que estes possuem como principal similaridade a raiz BONE, pois todos tem como foco temático o câncer de ossos.

Todavia, o agrupamento N°4 foi formado por sete documentos que tratam sobre o câncer de mama e por essa razão eles apresentam como similaridade a raiz BREAST.

Já o agrupamento N°5 agregou os documentos que abordam conceitos introdutórios acerca do câncer, e por esta razão eles têm como similaridade a raiz CANCER..

Enquanto o agrupamento N°8 contemplou dezessete documentos que tratam sobre fatores de risco inerentes ao câncer – justificando a similaridade representada pela raiz RISK – o agrupamento N°10 foi formado por dois documentos, patrocinados pela ONG norte-americana *National Ovarium Cancer Coalition*, que tratavam sobre o câncer de ovários.

5.4.1.3 Base de dados RNA

Para este experimento foram obtidos 6 agrupamentos, ilustrados na figura 5.12.

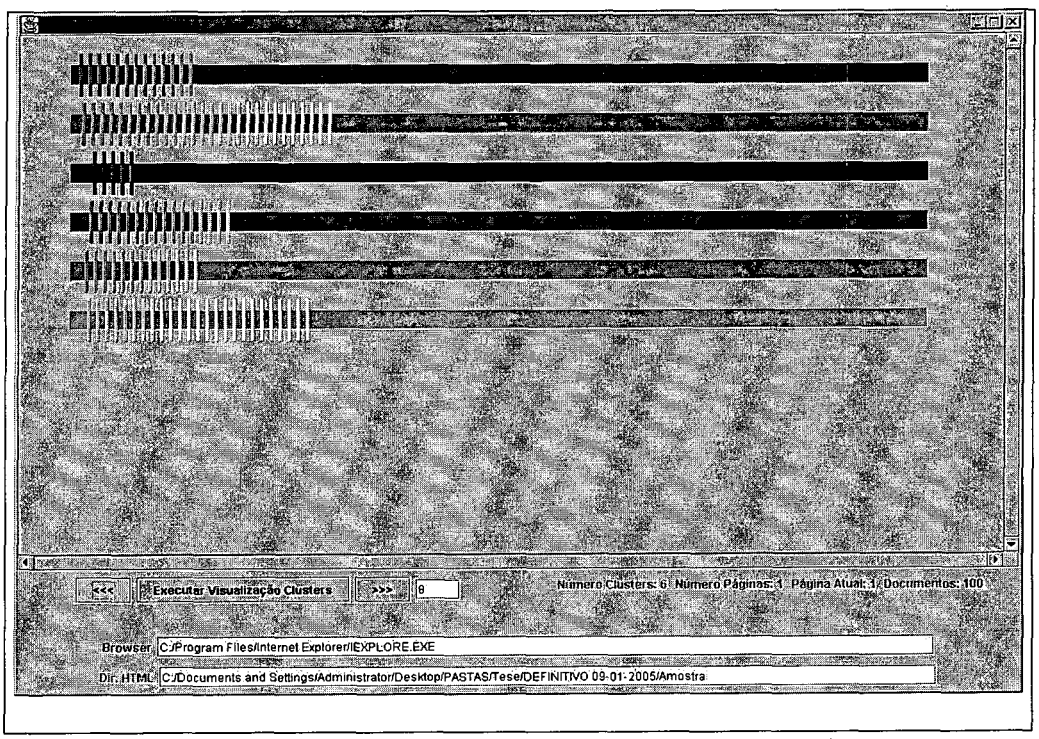

Figura 5.12: Visualização dos agrupamentos obtidos para base RNA

Agrupamento	Documentos	Raízes Comuns
1	13	DATA
2	28	NEURON
3	5	ANALOG
4	16	VECTOR
5	13	SYSTEM
6	25	NUMBER

Tabela 5.5: Descrição dos agrupamentos obtidos para base RNA.

Dentre os agrupamentos obtidos, apenas os agrupamentos Nº2 e Nº3 merecem destaque.Enquanto os documentos agregados no agrupamento Nº2 tratavam de conceitos introdutórios acerca do tema redes neuronais artificiais, os documentos presentes no agrupamento Nº3 tratavam sobre a implementação de redes neuronais em *hardware*.

5.4.1.4 Base de dados MARX

Para este experimento foram obtidos 6 agrupamentos, ilustrados na figura 5.13.

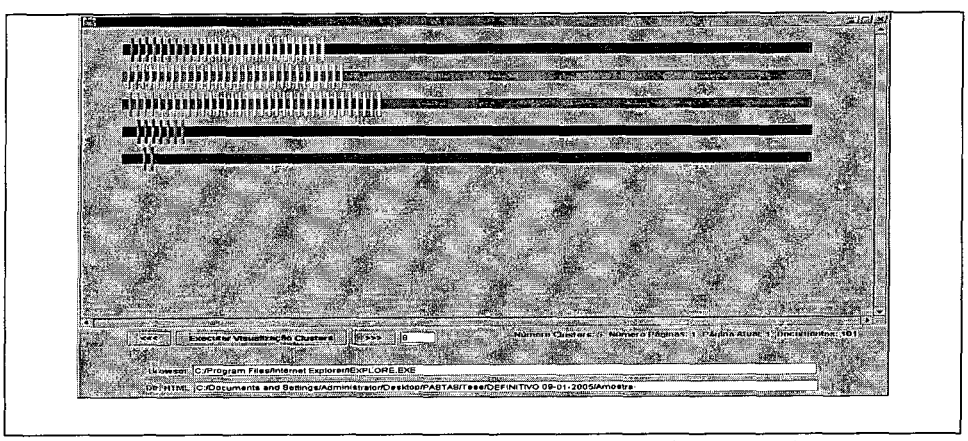

Figura 5.13: Visualização dos agrupamentos obtidos para base MARX.

Agrupamento	Documentos	Raízes Comuns
1	27	SOCIAL
2	30	MARX
3	35	MARXISM
4	7	EUROPE – WORLD
5	2	CLASS – MANIFESTO – SOCIETI – WORK-WORKER

Tabela 5.6: Descrição dos agrupamentos obtidos para base MARX.

O agrupamento Nº3 representa o único bom agrupamento obtido neste experimento, pois, os seus documentos indicam de forma explicita, através de seus respectivos títulos, que tratam sobre o tema marxismo.

Por sua vez, nos agrupamentos restantes, os documentos apresentam apenas similaridade quanto às raízes.

5.4.1.5 Base de dados ACMR

Como visto na subseção 5.4.1, esta base textual agrega os quatro grupos de documentos empregados nas subseções 5.4.1.1, 5.4.1.2, 5.4.1.3 e 5.4.1.4. No total, está base contempla 400 documentos. Apesar do tamanho do universo a ser considerado, está base textual agrega a vantagem de possuir uma classificação prévia, uma vez que existem quatro domínios de conhecimento envolvidos.

Devido a esta peculiaridade, foram executados dois experimentos para esta base textual.

Num primeiro experimento, sem o auxilio das heurísticas – respectivamente RF e DF - foi criado de forma manual um dicionário de raízes contendo os seguintes itens:

AIDS, CANCER, MARX e NEURAL. A finalidade deste experimento foi averiguar a capacidade de separação da rede *FUZZY-ART*, e obteve êxito uma vez que a classificação refletiu uma precisão de 100%, pois foram encontrados exatamente quatro agrupamentos.

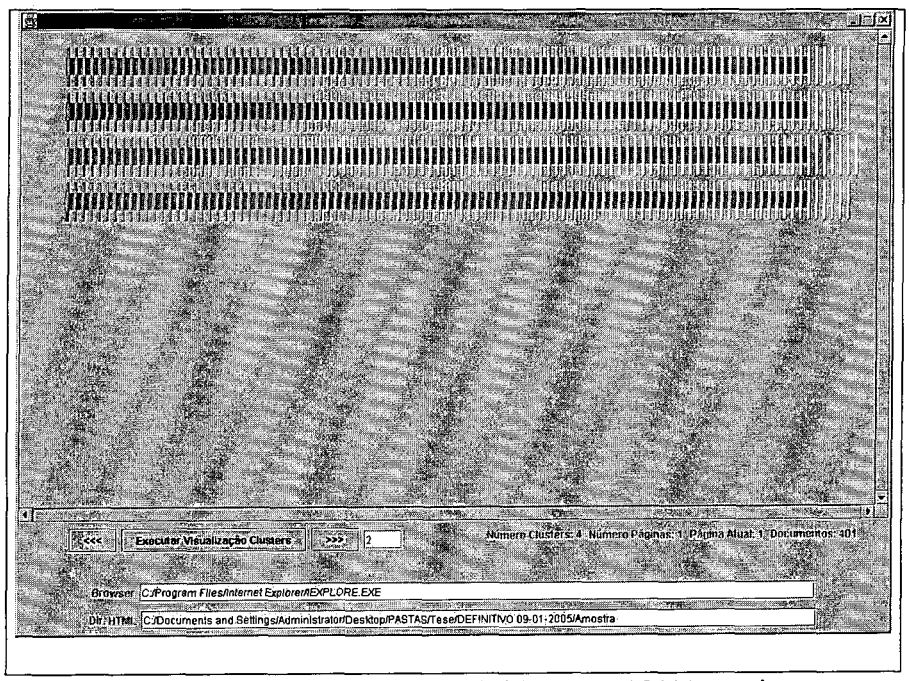

Figura 5.14: Agrupamentos obtidos com 100% precisão.

Por sua vez, para executar um segundo experimento, foram utilizados os parâmetros: α = 0.01, DFMin = 22% e DFMax = 30%. Esse valores encontram-se na tabela 5.2 e permitiram obter as raízes pertinentes a cada tema (Aids, Câncer, Karl Marx e Redes Neuronais Artificiais).

Após a execução da rede FUZZY-ART foram obtidos 8 agrupamentos, ilustrados na figura 5.15.

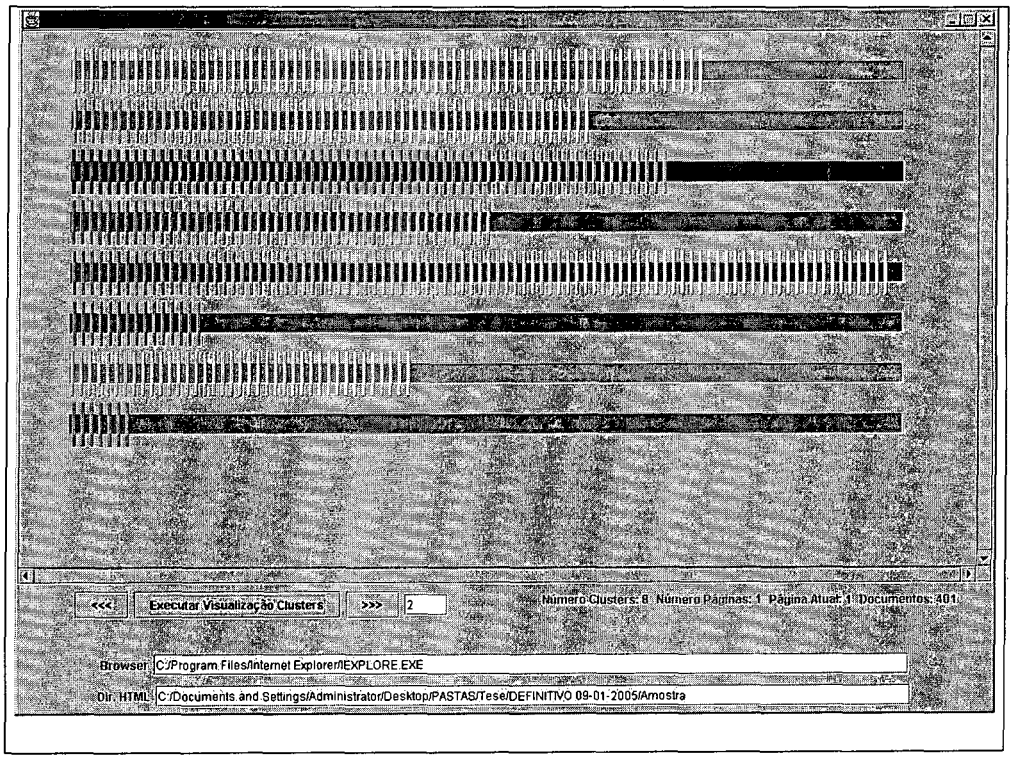

Figura 5.15: Agrupamentos do segundo experimento.

Agrupamento	Documentos	Raízes Comuns
1	72	AIDS
2	59	IMPORT
3	68	CANCER
4	48	MARX – MARXISM
5	93	GENER
6	15	CLASS – HISTORY – SOCIAL – SOCIETI – STATE
7	39	NEURAL – NETWORK
8	7	BASE – DATA – PROCESS – RESULT – SYSTEM

Tabela 5.7: Descrição dos agrupamentos obtidos para o segundo experimento.

Embora este experimento não tenha obtido a mesma precisão do anterior, este revelou alguns agrupamentos bastante interessantes. Os agrupamentos N°1, N°3, N°4 e N°7 agregam somente documentos que tratam respectivamente sobre os temas aids, câncer, Karl-Marx e Redes Neuronais Artificiais.

Por sua vez, os agrupamentos N°6 e N°8, pela característica inerente de seus documentos, revelam ser especialidades dos agrupamentos N°4 e N°7.

Isso pode ser percebido uma vez que os documentos presentes no agrupamento N°6 abordam temas como: A relação entre proletários e comunistas ou a sociedade humana e a economia global.

A mesma característica é observada no agrupamento N°8, pois seus documentos têm como foco temas como mineração de dados, processamento de linguagem natural, aprendizagem de máquina ao invés de tratar de forma explicita apenas sobre redes neurais.

Ainda com relação a este experimentos, os agrupamentos N°2 e N°5 não se mostraram satisfatórios, uma vez que seus documentos têm como similaridades as raízes IMPORT e GENER. Estas raízes são originárias das palavras *important* e *general* e para o contexto referente aos agrupamentos N°2 e N°5 são pouco informativas, embora ocorram, dentro dos respectivos documentos, com uma freqüência considerável.

Em todos os experimentos executados nesta seção verificou-se que os agrupamentos são formados por documentos possuem no mínimo a uma raiz comum. Também foi observado que esta raiz comum aparece em todos os documentos com uma freqüência bastante elevada.

Este fato demonstra que os agrupamentos obtidos são coerentes, pois existe uma similaridade plausível entre os documentos agregados.

Torna-se necessário ressaltar que, em muitos casos, a raiz comum observada - ou o conjunto de raízes - não expressa necessariamente o núcleo temático sob o ponto de vista de um usuário comum. Isto fica patente no agrupamento identificado pela raiz INFECT, presente no experimento da subseção 5.3.1.1. Neste agrupamento o documento intitulado "HIV / AIDS: A GAY MYTH" registra um elevado número de ocorrências das palavras INFECTION, INFECTIONS e INFECTED. Para um usuário comum, "infecção" não seria o contexto ideal para identificar este documento.

Entretanto, para o universo particular aos experimentos executados neste trabalho, isso, no entanto é coerente, pois existe uma distinção entre o núcleo semântico obtido pela análise de freqüência relativa e o núcleo temático visualizado por um usuário comum.

A análise de freqüência relativa, dependendo do parâmetro adotado α produz um núcleo semântico muito reduzido, o que é indesejável.

Isto leva a concluir que a rede *FUZZY-ART* produz bons resultados, mas, no entanto, existe a necessidade de melhorar o processo de obtenção dos vetores de freqüência.

Uma proposta interessante seria buscar técnicas mais eficientes para obter o núcleo semântico dos documentos.

Capitulo 6 - Conclusões

6.1 Conclusão

Este trabalho propôs a utilização de uma rede neuronal *FUZZY-ART* para formar agrupamentos de documentos hipertexto, uma vez que, no contexto da web, estes representam uma alternativa interessante aos tradicionais algoritmos de *PageRank.*

Agrupamentos já são empregados para visualizar resultados de consultas web, porém são obtidos por meio de algoritmos estatísticos e outras variedades de redes neuronais [64]. Nos dias atuais ainda existem poucas publicações sobre a aplicação das redes neuronais *ART* para este fim.

Ao contrário do que ocorre na mineração de dados tradicional, na qual os dados são estruturados, este trabalho envolveu um expressivo esforço na etapa que corresponde à extração dos atributos dos documentos hipertexto.

Esta etapa teve como finalidade de transformar documentos hipertexto em vetores de freqüência, representação adequada para qualquer algoritmo de mineração de dados.

Para visualizar os resultado empregou-se uma ferramenta gráfica na qual cada agrupamento foi representado por uma figura que lembrava uma régua, onde, por sua vez, os documentos eram representados como marcas dessa régua.

Para executar os experimentos foram utilizadas quatro bases textuais que retratavam assuntos bastante distintos: Aids, Redes Neuronais, Câncer e Karl Marx.

Foram feitas simulações sobre cada base em individual e o sobre o conjunto total, na qual foram misturados documentos sobre os quatro assuntos descritos acima.

Os agrupamentos obtidos em simulações executadas para conjuntos temáticos individuais mostraram-se coerentes. Por sua vez, a primeira simulação envolvendo o conjunto global proporcionou quatro agrupamentos bem definidos, o que expressou uma precisão de 100% na tarefa de encontrar os agrupamentos.

Todavia, neste caso não foram empregadas as heurísticas de freqüência relativa (RF) e Limiar de freqüência de documento (DF). As raízes que compuseram o dicionário ACMR foram escolhidas sem a utilização de qualquer heurística.

Isto demonstrou que a qualidade dos agrupamentos obtidos é proporcional ao conjunto de palavras selecionadas que ajudarão a compor o dicionário de raízes.

O segundo experimento executado para esta base textual obteve resultados interessantes, entretanto, nem tão precisos, vale ressaltar.

Como principais contribuições pode-se destacar a avaliação da rede neuronal *FUZZY-ART* na tarefa de formação de agrupamentos de documentos, bem como a introdução de uma forma original de visualizar os agrupamentos encontrados.

6.2 TRABALHOS FUTUROS

O Intuito deste trabalho foi avaliar a utilização das redes neuronais *FUZZY-ART* para formar agrupamentos de documentos. Todavia, como trabalho futuro pode-se citar o seguinte:

- Aplicar o processo de formação de agrupamento a documentos do tipo PDF e *PostScrip*.

- Fazer experimentos em uma base textual constituída por documentos escritos na língua inglesa e na língua portuguesa. Para tal, torna-se necessário utilizar técnicas dos anagramas para identifcar o idioma no qual uma palavra se encontra contextualizada.

- Utilizar técnicas mais eficiente para identificar de forma mais precisa o núcleo semântico de cada documento. Neste caso torna-se pertinente a aplicação de técnicas de processamento de línguagem natural (NLP).

- Comparar os resultados obtidos neste trabalho com um algoritmo estatístico como o *K-Means*.

- Utilizar a rede SOM para visualizar os agrupamentos obtidos.

Referências Bibliográficas

[1] www.crystalnet.com.br, consultado em agosto de 2004

[2] www.wikipedia.org, consultado em Setembro de 2004

[3] SOUZA, R. P., ALVARENGA, L., "A web semântica e suas contribuições para a ciência da informação" Ciência da Informação. Brasília: IBICT, v.33, n.1 (jan./abril 2004), p. 132-141

[4] www.educarede.org.br/educa/internet/historia.cfm, consultado em maio de 2004.

[5] www.cg.org.br, consultado em julho de 2004

[6] AFONSO, C. A.,
Internet no Brasil, IEEE Communications Magazine, jul / 1996

[7] www.ibope.com.br/noticias_2004_intertjun_no.htm, consultado em agosto de 2004.

[8] www.emarket.ppg.br/news_detalhes.asp?id=2685, consultado em julho de 2004

[9] www.nib.unicamp.br/recursos/www_tutr/introd.html, consultado em março de 2004.

[10] WEI, T., Search Engine Survey, 1999
Disponível em http://www.cc.gatech.edu/~wtang/research/papers/search_eng.pdf, consultado em junho de 2004

[11] www.absolutebusiness.net/help/glossary.htm, consultado em agosto de 2004

[12] www.google.com/press/overview_tech.htm, consultado em maio de 2004

[13] BRIN, S., PAGE, L. (1998) The Anatomy of a Large-Scale Hypertextual Web Search Engine, Proc.Seventh World Wide Web Conference

[14] ROGERS, I. (2002) The Google Pagerank Algorithm and How it Works
Disponível em: http://www.iprcom.com/papers/pagerank/index.html,
Consultado em abril de 2004

[15] www.Searchenginewatch.com, consultado em setembro de 2004

[16] PAL, S. K., TALWAR, V., MITRA, P., "Web Mining in Soft Computing Framework: Relevant, State of the Art and Future Directions", 2000.

[17] GIRARDI, R. "Main Approaches to Software Classification and Retrieval". Em: Ingeniería del Software y reutilización: Aspectos Dinámicos y Generación Automática. Editores J. L. Barros y A. Domínguez. (Universidad de Vigo – Ourense, del 6 al 10 de julio de 1998). Julio, 1998.

[18] www.askscott.com/sec1.html, consultado em julho de 2004

[19] SALTON, G. "An Introduction to Modern Information Retrieval". New York: McGraw-Hill, 1983.

[20] www.webliminal.com/search/glossary.htm, consultado em junho de 2004.

[21] GIRARDI, R.. *"Classification and Retrieval of Software through their Descriptions in Natural Language",* Ph.D. dissertation, No. 2782, University of Geneva, December 1995

[22] FRAWLEY, W.J., PIATETSKY, S. G., MATHEUS C.J. "Knowledge discovery in databases: an overview". In G. Piatetsky-Shapiro & W.J. Frawley, editors, "*Knowledge Discovery in Databases*". AAAI / MIT Press, 1991

[23] SALVINI, R. L., "Uma nova abordagem para Análise de Agrupamentos Baseada no Algoritmo Elástico". Master thesis - COPPE / UFRJ, Programa de Engenharia de Sistemas e Computação, 2000.

[24] MARINHO, L. B; GIRARDI, R. "Mineração na Web". 2003. Disponível em: www.sbc.org.br/reic/edicoes/2003e2/tutoriais/MineracaoNaWeb.pdf. Acesso em 10 de dezembro de 2003

[25] COOLEY, R. W. et al. "Web mining: Information and Pattern Discovery on the World Wide Web". In: *Proceedings of International Conference on Tools with Artificial Intelligence (ICTAI)*, 1997, 10p.

[26] KOSALA, R.; BLOCKEEL H. "Web Mining Research: A survey". In: *SIGKDD Explorations*, vol. 2 -1, June 2000, pp. 1-15.

[27] MACHADO, L. S., BECKER, K. "Distance Education: A Web Usage Mining Case Study for the Evaluation of Learning Sites". ICALT 2003: 360-361

[28] CHAKRABARTI, S., "Data mining for hypertext". ACM SIGKDD Explorations, 2000.

[29] ETZIONE, O., "The World Wide Web Quagmire or gold mine" Communications of the ACM, vol.39, no.11, pp. 65-68, 1996.

[30] AAS, K., EIKVIL, L. "Text Categorisation: A Survey",
Disponível em: www.nr.no/files/samba/bamg/tm_survey.ps,
Acesso em 15 de março de 2004

[31] LEE, B. T., HENDLER, J., LASSILA, O.
The Semantic Web. Scientifc American, May 2001

[32] CUTTING, D.D., KARGER, J., PEDERSON, J., SCATTER J., "A cluster based approach to browsing large document collections". Proceedings of the Fifteenth International Conference on Research and Development in Information Retrieaval, 1992.

[33] WILLET, P. "Recent trends in hierarchic document clustering: a critical review". Information Processing & Management, v.24, n.5, 1988.

[34] LINGRAS, P., "Rough Set Clustering for Web Mining". Saint Mary's University, 2002.

[35] MOBASHER, B., JAIN, N., HAN, E.H., SRIVASTAVA, J., "Web Mining: Patterns from WWW transactions". Tech. Rep. TR96-050, Dept. of Computer Science, University of Minesota, 1997.

[36] MLADENIC, M., GLOBELNIK, M., "Efficient text categorization". In Proceedings of Text Mining Workshop on the 10th European Conference on Machine Learning, 1998.

[37] SANTOS, R. G. "Utilização de técnicas *data mining* na busca de conhecimento na Web". Monografias de conclusão de curso de Ciência da Computação – 2000 (UFPEL) Seção 4.2.2, página 71.

[38] www.galeas.de/webmining.html consultado em agosto de 2004.

[39] KLEINBERG, J.M. "Authoritative Sources in a Hyper-linked Enviroment". In Proc. ACM-SIAM Symposyum on Discrete Alghoritms, 1998.

[40] CHAKRABARTI, S., DOM, B., GIBSON, D., KLEINBERG, J. , KUMAR, S., RAGHAVAN, P., RAJAGOPALAN, S., TOMKINS, A., "Mining the link structure of the World Wide Web", 1999.

[41] KUMAR, S.R. "Trawling the web for emerging cybercommunities". In Proceedings of the Eighth WWW Conference, 1999.

[42] BORGES, J., LEVENE, M., "Mining association rules in hypertext databases". In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98). New York City, New York, USA, 1998.

[43] COOLEY, R., MOBASHER, B., SRIVASTAVA, J., "Data preparation for mining world wide web browsing patterns". Knowledge and Information Systems, 1999.

[44] WARD, J., "Hierarchical grouping to otimize an objective function", Journal of the American Statistical Association, v.58, pp.236-244, 1963

[45] DILLON, W.R., GOLDSTEIN, M., "Cluster Analysis". In: Multivariate Analysis - methods and Applications, 1 ed., chapeter 5, New York, USA, John Willey & Sons, Inc., 1984

[46] FAUSETT, L. *"Fundamentals of Neural Networks: architectures, algortithms, and applications"*. Englewood Cliffs, Prentice Hall, 1994.

[47] CARPENTER, G., GROSSBERG, S., "*A Massively parallel architecture for a Self-Organising Neural Pattern Recognition Machine*", Computer Vision, Graphics, and Image Processing, Vol. 37 pp 54-115, 1987.

[48] LOESCH, C., SARI, S. T. "Redes Neurais Artificiais – Fundamentos e Modelos". Editora FURB, 1996. P.111-126.

[49] CARPENTER, G., GROSSBERG, S. "ART-2: Self-Organization of Stable CategoryRecognition Codes for Analog Input Patterns". Applied Optics, 26:4919-4930,1987

[50] ANDERSON, PELLIONISZ "ART2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns". Applied Optics, 151-162, 1990.

[51] WASSERMAN, P.D., *Neural Computing: Theory and Practice*, VanNostrand Reinhold, New York, 1989.

[52] HERTZ, J., KROGH, A., PALMER R. G. *"Introduction to the Theory of Neural Computation"*. Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley, Redwood City, CA, 1990.

[53] BARALDI, A., BLONDA, P. "*A survey of fuzzy clustering algorithms for pattern recognition*". II. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 29:786--801.

[54] CARPENTER, G. A., GROSSBERG, S., ROSEN, D.B. "*Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System*", Neural Networks, 1991, Volume 4, pp.759-771

[55] BLUME, M., ESENER, S. C., "*An efficient mapping of Fuzzy ART onto a neural architecture*", Neural Networks, Vol. 10, No. 3, pp.409-411

[56] CARPENTER, G. A., GROSSBERG, S., ROSEN, D. B. "*A neural network realization of Fuzzy ART*", Technical Report CAS/CNS-91-021. Boston, MA: Boston University, Center for Adaptive Systems.

[57] ORENGO, V. M., HUYCK, C. "A Stemming Algorithm for Portuguese Language". In: Proceedings of Eigth Symposium on String Processing and Information Retrieval (SPIRE 2001), Chile, (2001) 186-193.

[58] TANG, N, 2001, "Text Categorisation using Support Vector Machines", MSc Dissertation, University of Sheffield, Sheffield,Western Bank, UK

[59] NASCIMENTO, M.A., Cunha, A.C.R., *"An Experiment Stemming Non-Traditional Text"*. SPIRE'98, Proceedings, p. 75-80. Santa Cruz de La Sierra, Bolivia, Sep/98.

[60] PORTER, M. F., "An algorithm for suffix stripping", in S. Jones, K. and P. Willet, Readings in In-formation Retrieval, San Francisco, CA: Morgan Kaufmann, 1997

[61] YANG, Y., PEDERSEN, J. O., "A comparative study on feature selection in text categorization". In Proceedings of ICML-97, 14thInternational Conference on Machine Learning, 412-420, 1997

[62] JOACHIMS, T., "Transductive inference fortext classification using support vector machines". InProceedings of the 16th International Conference onMachine Learning, pages 200–209, 1999.

[63] ZAMIR, O., ETZIONI, O., MADANI, O., and KARP, R. M., "Fast and intuitive clustering of web documents", In KDD'97, pages 287–290, 1997.

[64] KASKI, S., HONKELA, T., LAGUS, K, KOHONEN, T, "WEBSOM-Self-organizing Maps of Document Collections". Neurocomputing, 21, pages 101-117, 1998

Apêndice A

A – ALGORITMOS E FERRAMENTAS IMPLEMENTADAS

A1) *WebMinerAux*

Ferramenta implementada em C++, responsável por fazer varredura de um diretório convertendo todo documento *.HTML num arquivo *.STM. Para lidar com um número elevado de documentos, este módulo foi desenvolvido para executar *threads* do sistema operacional *Windows*.

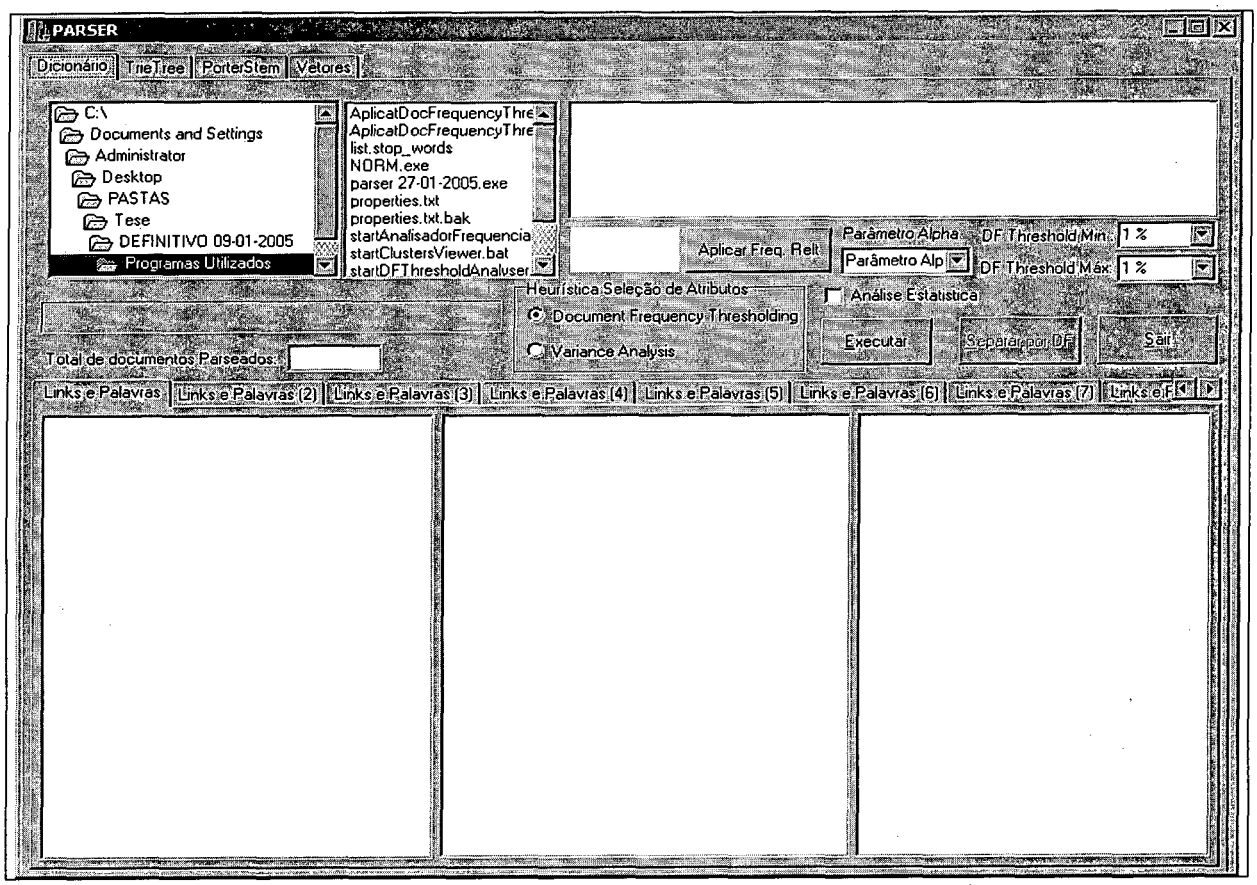

Figura A1: Aplicativo que converte documentos *.HTML em arquivos *.STM

A2) *RFAnalyser*

Ferramenta implementada em *JAVA*, responsável por aplicar a análise de freqüência relativa sob cada arquivo *.STM presente um diretório informado como parâmetro.

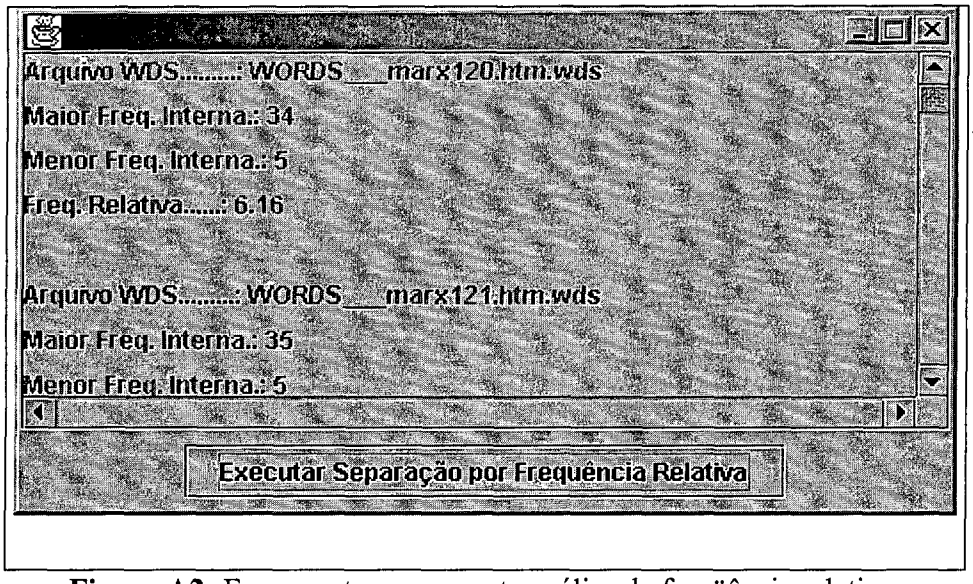

Figura A2: Ferramenta que executa análise de freqüência relativa.

A3) *DFAnalyser*

Ferramenta implementada em *JAVA* responsável por aplicar a heurística de freqüência de documentos, objetivando a construção de um dicionário de raízes a partir de um conjunto de arquivos *.STM (SF) presentes em um diretório.

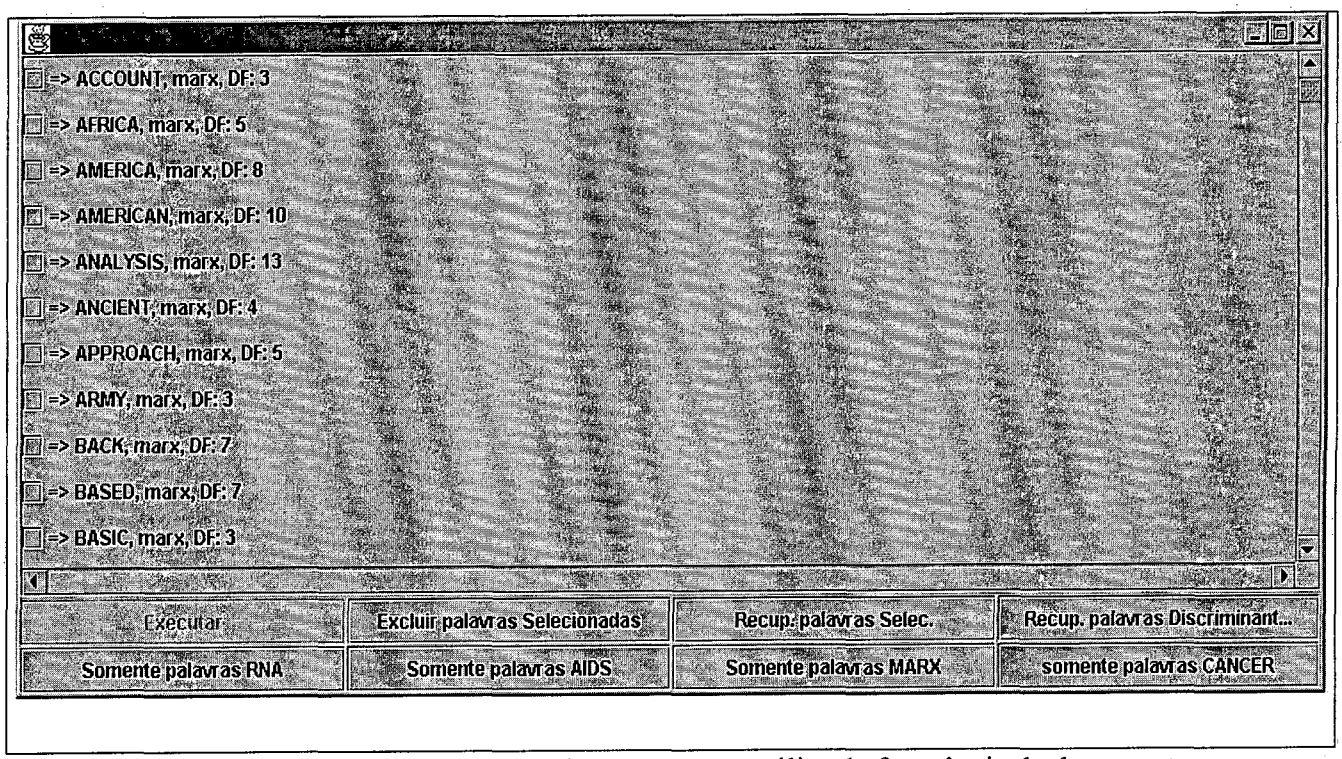

Figura A3: Ferramenta responsável por executar análise de freqüência de documentos.

A4) *Algoritmo responsável pela construção e normalização da matriz de frequências*

```
/**
   * @param wdsFilesDir
   * @param dictionaryFile
   * @param pathSaveDir
   * @throws Exception
   */
  public void buildFrequencyMatrix( String wdsFilesDir,
        String dictionaryFilePath,
              String pathSaveDir, boolean normalize) throws Exception {
        // Leitura de cada arquivo WDS
        BufferedReader bufferReader = null;

        // Escrita da matriz de freqüencias
        BufferedWriter bufferWriter = null;

        // Dicionário de raízes
        ArrayList dictionary = new ArrayList();
```

```
// Mapeamento Nome do Arquivo => Objeto Input
Map wdsFileInputObjectMap = new TreeMap(new TreeMapComparator());

// Array de valores baixos, a serem empregados na normalização
double[] lowerValuesArray = new double[]{};

// Array de valores altos, a serem empregados na normalização
double[] highValuesArray  = new double[]{};

// Menor valor
final double INITIAL_LOWEST_VALUE = 100000F;

// Maior valor
final double INITIAL_HIGHEST_VALUE = 0F;

// Flag indicando se o cabeçalho do arquivo foi escrito
boolean writeHeader = false;

/**
 * Montando ArrayList que representará o dicionário de raízes
 */
try {
   String stemmRead = "";
   bufferReader = new BufferedReader(
           new FileReader(dictionaryFilePath));
   while ( (stemmRead = bufferReader.readLine()) != null ) {
         if (dictionary.contains(stemmRead))
            throw new Exception("Repeated stem");
         dictionary.add( stemmRead );
   }


   if ( dictionary.isEmpty() ) {
     throw new Exception("Dictionary is Empty.");
   }

   // Fechando Stream de leitura
   bufferReader.close();
   bufferReader = null;

   if ( normalize ) {
         lowerValuesArray = new double[ dictionary.size() ];
         highValuesArray = new double[ dictionary.size() ];

         Arrays.fill( lowerValuesArray, INITIAL_LOWEST_VALUE );
         Arrays.fill( highValuesArray, INITIAL_HIGHEST_VALUE );
   }
```

```
// Recuperando todos os arquivos WDS
File[] arquivosWDS = this.scanDirectoryWds( wdsFilesDir ) ;

// Analisando arquivo a arquivo
for ( int i=0; i<arquivosWDS.length; i++ ) {

        /**
         * Dado arquivo, esta estrutura executa o mapeamento de uma
* raiz
         * para o seu número de ocorrencias no documento corrente
         */
        Map termFreqMap = this.
      assemblyTermFrequencyMap( arquivosWDS[i], false );
        String nomeArqWDS = arquivosWDS[i].getName();

        // Verifica se já existe uma referencia para o arquivo
        if (wdsFileInputObjectMap.containsKey( nomeArqWDS )) {
           continue;
        }

        // Instancia um objeto input, a representar um vetor
// de frequencias
        Input input = new Input( nomeArqWDS, dictionary.size() );

        // Varre dicionário de raízes para atribuir frequencias
        for ( int j=0; j<dictionary.size(); j++ ) {

           // Recupera raiz
           String stm = (String) dictionary.get(j);

           // Tentar recuperar a frequencia de ocorrencia no
  // documento corrente
           Integer freqValue = (Integer) termFreqMap.get(stm);

           // Verifica se existe mapeamento para esta raiz, no
  // documento corrente
           if ( freqValue != null ) {
                // Se existe, atribuir valor correspondente a
   // frequencia
                input.inputVector[j] = freqValue.doubleValue();
           }

           // Se vai se executada normalizacao
           if ( normalize ) {
                // Calcula vetor contendo os valores mais baixos
                if ( input.inputVector[j] < lowerValuesArray[j] ) {
```

```
                    lowerValuesArray[j] = input.inputVector[j];
                }

                // Calcula vetor contendo os valores mais altos
                if ( input.inputVector[j] >= highValuesArray[j] ) {
                    highValuesArray[j] = input.inputVector[j];
                }
            }
        }

        // Mapeia o nome de um arquivo para um objeto que encapsula
// vetor de frequencias
        wdsFileInputObjectMap.put( nomeArqWDS, input );
    }

    String fileName =  pathSaveDir +
            (normalize ? "vet.rw" : "vet.nrm");
    bufferWriter = new BufferedWriter (new FileWriter(fileName,
                false));

    bufferWriter.write("NUMBER_OF_VECTORS=" +
        wdsFileInputObjectMap.size() + "\n");
    bufferWriter.write("NUMBER_OF_ATRIBUTES=" + dictionary.size()
        + "\n");

    bufferWriter.write("RAIZES-DIC=");
    Iterator it=dictionary.iterator();

    while (it.hasNext()) {
        String stm = (String) it.next();
        bufferWriter.write(stm + ";");
    }

    bufferWriter.newLine();

    it=wdsFileInputObjectMap.keySet().iterator();
    while ( it.hasNext() ) {
        Input inputObj = (Input) it.next();
        bufferWriter.write(inputObj.getDocumentName() + " ");

        for ( int i=0; i<inputObj.inputVector.length; i++ ) {
            if ( normalize ) {
                inputObj.inputVector[i] =
      (inputObj.inputVector[i] -
        lowerValuesArray[i]) / highValuesArray[i];
            }
```

```
            bufferWriter.
                write(String.valueOf(inputObj.inputVector[i])) ;



                    if ( i < (inputObj.inputVector.length - 1) ) {
                        bufferWriter.write(" ");
                    }
                }

                bufferWriter.newLine();
            }
        } catch (Exception e) {
            throw new NestableException("buildFrequencyMatrix().", e);
        } finally {

            try {
                if (bufferReader != null) {
                    bufferReader.close();
                    bufferReader = null;
                }

                if (bufferWriter != null) {
                    bufferWriter.close();
                    bufferWriter = null;
                }

            } catch (Throwable t) {
                throw new NestableException(
              "buildFrequencyMatrix() fail.", t);
            }
        }
    }
```

A5) *Implementação FUZZY-ART*

```
//*********************************************************************
  // Executa o treinamento da rede
  // - Critério de Parada : 1) Redes Estabilizada (Classes não Varaiam)
  //                        2) Excedido número de épocas Estabelecido
  // - Passos : 1) Habilitar todas as categorias
  //            2) Achar a categoria vencedora entre as habilitadas
  //            3) Testar Ressonância
  //               -- Se ok, vai para o passo 4
  //               -- Se não, voltar para o passo 2
  //            4) Ajustar Categoria, atualizar o seu vetor de pesos
  //               e voltar para o passo 1
//*********************************************************************
  public void executeArtTraining() throws StateException, Throwable {
    File directoryCreator = null;        // Cria o diretório de resultados
    Input input = null;                  // Apontador para a classe Input
    Category winnerCategory = null;      // Apontador para a classe Category
    int nextStep = 1;                    // Flag que controla passos da rede
    boolean exitLoop = false;            // Flag que controla loop de épocas

    // Testando se o diretório de resultados é valido
    if (!NeuralArtLog.verifyPhisicalPath(resultDirectory))
      throw new StateException("NeuralArt executeArtTraining() fail,
            result directory[" + resultDirectory + "] can not be created.",
            StateException.INVALID_RESULT_DIRECTORY);

    // Verificando os vetores de entrada forma lidos
    if (inputArray == null || inputArray.size() == 0)
      throw new StateException("NeuralArt executeArtTraining() fail.",
            StateException.INPUT_ARRAY_IS_EMPTY);
```

```
// Inicialmente começaremos o treinamento com apenas uma categoria presente no
Array,
// e sempre que necessário novas Categorias serão acrescentadas ao Array
addEntriescategoryArray(false);

// Inicializando Rank de Categorias (Inicialmente está vazio !)
activeCategoriesRank = new CategoryRank();

//************************************************************
// Treinamento será executado até que o número máximo de épocas
// seja alcaçado ou a rede ART se estabilize
//************************************************************
for (int epoch=1; epoch<this.maxEpochs && !exitLoop; epoch++) {

  // Quando a rede estabilizar terá valor "zero"
  int stabilityFlag = inputArray.size();

  // Indica que o próximo passo a ser executado é o passo 1
  nextStep = 1;

  // Caminho do arquivo de Log
  String outLog = resultDirectory + "/" + epoch + ".log";

  for (int inputIndex=0; inputIndex<inputArray.size();) {

    //*********************************************
    // Passo 1: Habilitar todas as entradas.....
    //*********************************************
    if (nextStep == 1) {
      NeuralArtLog.getNeuralArtLogInstance().writeLine
            (outLog, "-> passo 1 - habilitar todas categorias.\n");

      // Pegando um vetor de entrada
```

```
        input = (Input) inputArray.get(inputIndex);

        NeuralArtLog.getNeuralArtLogInstance().writeLine
                (outLog, "-> Input Apresentado:\n" + input.toString());

        // Habilitando todas as categorias (1° Passo)
        enableAllCategories();

        // Removendo todas as categorias do Rank
        activeCategoriesRank.clearAll();

        // Vai para o passo 2
        nextStep = 2;
      }

      //************************************************
      // Passo 2: Encontrar a categoria vencedora.....
      //************************************************
      if (nextStep == 2) {
        NeuralArtLog.getNeuralArtLogInstance().
                writeLine(outLog, "-> passo 2 - buscar categoria vencedora.\n");

        // Verifica se existem categorias habilitadas
        if (getNumberOfEnabledCategories() == 0) {
          NeuralArtLog.getNeuralArtLogInstance().
                writeLine(outLog, "-> não existem categoria habilitadas disp.");
          addEntriescategoryArray(true);
        }

        // Buscando a categoria vencedora
        winnerCategory = findWinnerCategory(input.inputVector,
this.epsonParameter, outLog);
```

```
            // "Logando" categoria vencedora
            NeuralArtLog.getNeuralArtLogInstance().
                    writeLine(outLog, "\n-> Categoria Vencedora obtida:\n" +
                            winnerCategory.toString() + "\n");
            nextStep = 3; // Próximo passo
        }

        //***********************************************
        // Passo 3: Teste de Ressonância.....
        //***********************************************
        if (nextStep == 3) {

            // Computando fator ou taxa de ressonancia
            double ressonanceTax =
winnerCategory.computeRessonanceTax(input.inputVector);

            if (ressonanceTax >= vigilanceParameter) {

                // Sucesso, foi gerado ressonância
                NeuralArtLog.getNeuralArtLogInstance().
                            writeLine(outLog, "-> Categoria ["+
                            winnerCategory.getCategoryId() +
                            "] ressonou, tx. ressonancia ==> " +
                            ressonanceTax + ".");
                nextStep = 4;
            } else {

                // Categoria não gerou ressonância
                NeuralArtLog.getNeuralArtLogInstance().
                            writeLine(outLog, "-> Categoria ["+
                            winnerCategory.getCategoryId() +
                            "] não ressonou, tx. ressonancia ==> " + ressonanceTax +
".");
```

```
                // Desativa a categoria
                winnerCategory.setStatus(false);

                // Remove do rank de categorias ativas
                if (activeCategoriesRank.removeCategory(winnerCategory)) {
                   NeuralArtLog.getNeuralArtLogInstance().
                                writeLine(outLog, "-> Categoria ["+
winnerCategory.getCategoryId() +
                                "] removida do rank de categorias ativas.\n");
                } else {
                   throw new StateException(
                                "NeuralArt executeArtTraining() fail while removing
category["
                                + winnerCategory.getCategoryId() + "].",

        StateException.CATEGORY_COULD_NOT_BE_REMOVE_FROM_RANK);
                }

                nextStep = 2;
             }
          }

          //***********************************************
          // Passo 4: Atualização da Categoria.....
          //***********************************************
          if (nextStep == 4) {

             // String que armazenará configuração do vetor
                //    antes e após a atualização da categoria.
             String printString = "categoria [" +
                      winnerCategory.getCategoryId() + "] .... \n" +
                "antes de aprender [abaixo] :\n" + winnerCategory.printWeightVecotr() +
                "\nApós Aprender\n";
```

```
        // Atualizando Categoria Vencedora
        winnerCategory.updateWeightVector(input.inputVector);

        // Configuração do vetor de pesos após o aprendizado
        printString += winnerCategory.printWeightVecotr() + "\n";

        // Logando a categoria, antes da atualização e após a atualização
        NeuralArtLog.getNeuralArtLogInstance().writeLine(outLog, printString);

        // Atualização de Flag de convergencia
        if (input.getCategoryId() == winnerCategory.getCategoryId()) {
          NeuralArtLog.getNeuralArtLogInstance().
                        writeLine(outLog, "-> vetor [" +
                        input.getDocumentName() +
                        "] tem se mantido na mesma categoria.");
          if (stabilityFlag > 0)
            stabilityFlag--;
        }

        // Setando Categoria em que foi classificado o input corrente
        input.setCategoryId(winnerCategory.getCategoryId());

        // Incrementando índice para pegar próximo input
        inputIndex ++;

        // Indo para o passo 1 novamente
        nextStep = 1;
      }
    }

    //**************************
    // Finalização de Uma época
```

```
//**************************
StringBuffer obtainedResults = new StringBuffer();

obtainedResults.append("\n\n\n");
obtainedResults.append("**********************");
obtainedResults.append("\nResultados Obtidos\n");
obtainedResults.append("**********************");

// Escrevendo no arquivo de log uma tabela de classificação no seguinte formato :
// EX....: [Category Index]: [Doc Name], ....., [Doc Name]
for (int categoryIndex=0; categoryIndex<categoryArray.size(); categoryIndex++) {

   int cID = ((Category) categoryArray.get(categoryIndex)).getCategoryId();
   String stringToBeInserted = "\nCategory [" + cID + "]:";

   // Flag que indica quantos inputs foram classificados pela categoria em questão
   int numberOfInputsClassified = 0;
   for (int inputIndex=0; inputIndex<inputArray.size(); inputIndex++) {
      int dID = ((Input) inputArray.get(inputIndex)).getCategoryId();
      if (cID == dID) {
         String docName = ((Input)
inputArray.get(inputIndex)).getDocumentName();
         stringToBeInserted += " [" + docName + "] ";
         numberOfInputsClassified++;
      }
   }

   // Definindo se a categoria classificou alguém ou não.
   if (numberOfInputsClassified <= 0)
      ((Category) categoryArray.get(categoryIndex)).hasClassiffiedNoInputs(true);
   else {
      ((Category) categoryArray.get(categoryIndex)).hasClassiffiedNoInputs(false);
      obtainedResults.append(stringToBeInserted);
```

```
            }
        }

        // Escrevendo no log a tabela de classificação
        NeuralArtLog.getNeuralArtLogInstance().writeLine(outLog, obtainedResults);

        // Limpando o buffer de impressão
        obtainedResults.delete(0, obtainedResults.length()-1);

        obtainedResults.append("\n\n");

obtainedResults.append("************************************************");
        obtainedResults.append("\nResultados Obtidos - Sob uma nova perpectiva\n");

obtainedResults.append("************************************************");

        // Escrevendo no arquivo de log uma tabela de classificação no seguinte formato :
        //  EX....:  [Doc Name]: [Category Index]
        for (int inputIndex=0; inputIndex<inputArray.size(); inputIndex++) {
            String dName = ((Input) inputArray.get(inputIndex)).getDocumentName();
            int dID = ((Input) inputArray.get(inputIndex)).getCategoryId();
            String stringToBeInserted = "\nDocument [" + dName + "]:";
            for (int categoryIndex=0; categoryIndex<categoryArray.size(); categoryIndex++)
{
                int cID = ((Category) categoryArray.get(categoryIndex)).getCategoryId();
                if (cID == dID) {
                    String docName = ((Input)
inputArray.get(inputIndex)).getDocumentName();
                    stringToBeInserted += " [" + cID + "] ";
                }
            }
            obtainedResults.append(stringToBeInserted);
        }
```

```
// Escrevendo no log a tabela de classificação
NeuralArtLog.getNeuralArtLogInstance().writeLine(outLog, obtainedResults);

// Limpando o buffer de impressão
obtainedResults.delete(0, obtainedResults.length()-1);

obtainedResults.append("\n\n");
obtainedResults.append("************************");
obtainedResults.append("\nCategorias Removidas\n");
obtainedResults.append("************************");

// Escrevendo no log conteúdo do buffer
NeuralArtLog.getNeuralArtLogInstance().writeLine(outLog, obtainedResults);

// Limpando o buffer de impressão novamente
obtainedResults.delete(0, obtainedResults.length()-1);

// Removendo categorias que não classificaram ninguém
for (int categoryIndex=0; categoryIndex<categoryArray.size(); categoryIndex++) {
   if (categoryIndex >= 0 && categoryIndex < categoryArray.size()) {

      // Se não classificou ninguém ....
      if (((Category) categoryArray.get(categoryIndex)).hasClassiffiedNoInputs()) {
         obtainedResults.append("\n= = = > [ " +
               ((Category) categoryArray.
                     get(categoryIndex)).getCategoryId() + " ]");
         categoryArray.remove(categoryIndex);  // ... remove categoria
         categoryIndex--;                      // ... e decrementa indice
      }
   }
}

if (obtainedResults.length() <= 1)
```

```
            obtainedResults.append("= = = > [ Nenhuma ]");

        // Escrevendo no log as categorias removidas
        NeuralArtLog.getNeuralArtLogInstance().writeLine(outLog, obtainedResults);

        // Verificando se a rede alcançou a estabilidade
        if (stabilityFlag <= 0) {
            NeuralArtLog.getNeuralArtLogInstance().writeLine(
                        outLog, "\n\n -> Rede Alcançou Estabilidade.");
            NeuralArtLog.getNeuralArtLogInstance().writeLine(
                        outLog, "\n\n -> Taxa de Estabilidade: " + stabilityFlag + ".");
            exitLoop = true;
        } else {
            NeuralArtLog.getNeuralArtLogInstance().writeLine(
                        outLog, "\n\n -> Rede Ainda Não Estabilizou.");
            NeuralArtLog.getNeuralArtLogInstance().writeLine(
                        outLog, "\n\n -> Taxa de Estabilidade: " + stabilityFlag + ".");
        }
    }

    NeuraArtStream.getNeuralArtInstance().writeFinalResult(
                inputArray, categoryArray);
    NeuraArtStream.getNeuralArtInstance().
                saveResultTraining("InputsClassified_vigparam" +
                vigilanceParameter, inputArray);
}
```

A6) *ClusterViewer*

Ferramenta desenvolvida em Java, com a finalidade de proporcionar a visualização dos agrupamentos obtidos.

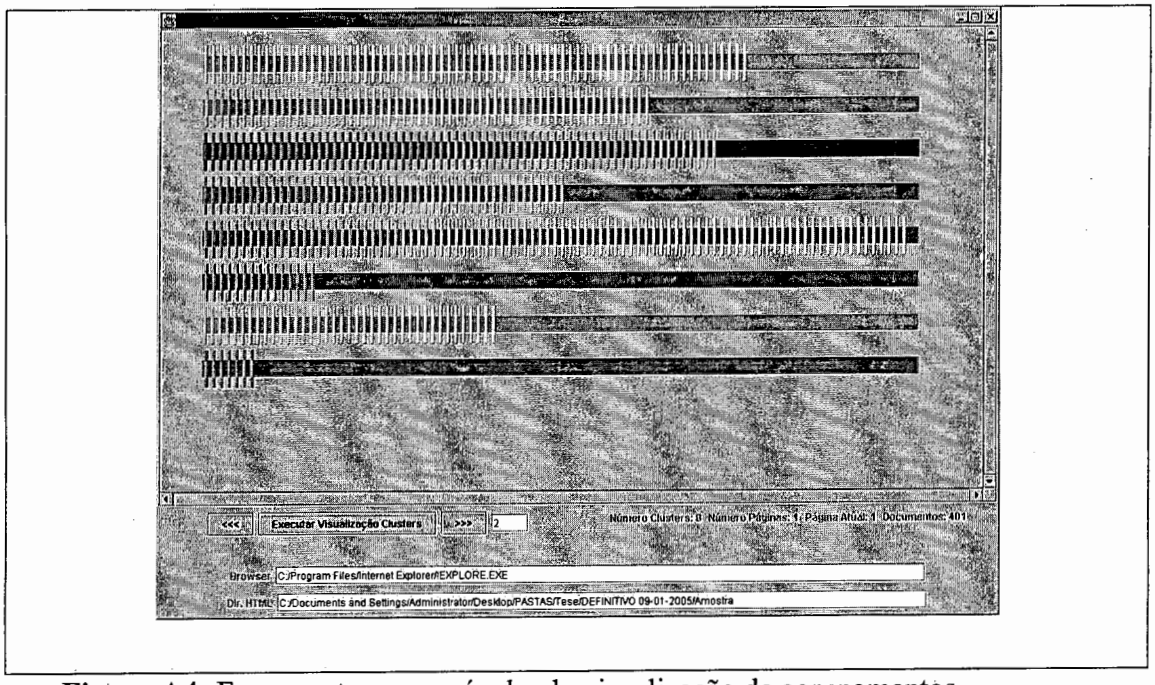

Figura A4: Ferramenta responsável pela visualização de agrupamentos