UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JOSÉ DE LIMA JUNIOR

Descoberta de Equivalência Semântica entre Atributos em Bancos de Dados Utilizando Redes Neurais

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Paulo Martins Engel
Orientador

Porto Alegre, julho de 2004.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Lima Junior, José de

Descoberta de Equivalência Semântica entre Atributos em Bancos de Dados Utilizando Redes Neurais / José de Lima Junior – Porto Alegre: Programa de Pós-Graduação em Computação, 2004.

88 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2004. Orientador: Paulo Martins Engel.

1. Introdução. 2. Trabalhos relacionados. 3. Equivalência semântica. 4. Descoberta de conhecimento em bancos de dados. 5. A ferramenta SEA – Semantic Equivalence Attribute. 6. Estudo de caso. 7. Contribuições e trabalhos futuros. I. Engel, Paulo Martins. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profª. Wrana Maria Panizzi
Pró-Reitor de Ensino: Prof. José Carlos Ferraz Hennemann
Pró-Reitora Adjunta de Pós-Graduação: Profª. Jocélia Grazia
Diretor do Instituto de Informática: Prof. Philippe Olivier A. Navaux
Coordenador do PPGC: Prof. Carlos Alberto Heuser
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Se você tiver uma maçã e eu tiver uma maçã e se fizermos uma troca, cada um terá uma maçã. Se você tiver uma idéia e eu tiver uma idéia e se fizermos uma troca, cada um terá duas idéias.

George Bernard Shaw

AGRADECIMENTOS

A Deus, por mais uma etapa vencida. Obrigado por ter concedido-me forças nos momentos difíceis, por ter iluminado meus caminhos e por ter me proporcionado a oportunidade de conhecer pessoas tão especiais neste período.

Ao meu orientador Professor Dr. Paulo Martins Engel, pelo acompanhamento, pela orientação e pelas excelentes sugestões.

Quero agradecer à Universidade Federal do Rio Grande do Sul e aos funcionários, em especial aos da secretaria do curso de pós-graduação, biblioteca e laboratório de informática, que sempre me atenderam com muita dedicação.

A Universidade Estadual de Londrina, e especialmente a Assessoria de Tecnologia de Informação.

A toda minha família e em especial a minha esposa Veronice de Freitas, pelo incentivo para que eu fizesse o mestrado e pela colaboração nos momentos que precisei.

A todos os que, direta ou indiretamente, contribuíram para a realização deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS

ART	Adaptative Resonance Theory
BDD	Bancos de Dados Distribuídos
DBMS	Data Management Systems
DCBD	Descoberta do Conhecimento em Banco de Dados
DDL	Data Definition Language
DMV	Dependência Multivalorada
DSS	Decision support systems
DW	Data Warehousing
EIS	Executive Information Systems
FD	Functional Dependency
GUI	Graphical User Interface
KDD	Knowledge Discovery in Databases
LHS	Left Hand Side
LTM	Long-Term Mmemory
MD	Mineração de Dados
MDB	Multi-dimensional Database
MDBMS	Multi Database Management System
MLP	Multi-Layer Perceptron
ODBC	Open Database Connectivity
OLAP	On-Line Analytical Processing
PAM	Partitioning Around Medoids
RBF	Radial Basis Function
RDF	Resource Description Format
RNA	Redes Neurais Artificiais
RNSP	Redes Neurais Sem Peso
SBD	Sistemas de Bancos de Dados
SBDF	Sistemas de Bancos de Dados Federados

SBDH	Sistemas de Bancos de Dados Heterogêneos
SGBD	Sistema Gerenciador de Bancos de Dados
SGBDD	Sistema Gerenciador de Bancos de Dados Distribuídos
SOM	Self Organizing Map
SQL	Structured Query Language
STM	Short-Term Memory
XML	Extensible Markup Language
SEA	Semantic Equivalence Attribute
PCA	Principal Component Analysis

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

RESUMO

Com o crescimento das empresas que fazem uso das tecnologias de bancos de dados, os administradores destes bancos de dados criam novos esquemas a cada instante, e na maioria dos casos não existe uma normalização ou procedimentos formais para que tal tarefa seja desempenhada de forma homogênea, resultando assim em bases de dados incompatíveis, o que dificulta a troca de dados entre as mesmas. Quando os Sistemas de Bancos de Dados (SBD) são projetados e implementados independentemente, é normal que existam incompatibilidades entre os dados de diferentes SBD. Como principais conflitos existentes nos esquemas de SBD, podem ser citados problemas relacionados aos nomes dos atributos, armazenamento em diferentes unidades de medida, diferentes níveis de detalhes, atributos diferentes com mesmo nome ou atributos iguais com nomes diferentes, tipos de dado diferentes, tamanho, precisão, etc. Estes problemas comprometem a qualidade da informação e geram maiores custos em relação à manutenção dos dados. Estes problemas são conseqüências de atributos especificados de forma redundante. Estes fatos têm provocado grande interesse em descobrir conhecimento em banco de dados para identificar informações semanticamente equivalentes armazenadas nos esquemas. O processo capaz de descobrir este conhecimento em banco de dados denomina-se DCDB (Descoberta de Conhecimento em Bancos de Dados). As ferramentas disponíveis para a execução das tarefas de DCDB são genéricas e derivadas de outras áreas do conhecimento, em especial, da estatística e inteligência artificial. As redes neurais artificiais (RNA) têm sido utilizadas em sistemas cujo propósito é a identificação de padrões, antes desconhecidos. Estas redes podem aprender similaridades entre os dados, diretamente de suas instâncias, sem conhecimento a priori. Uma RNA que tem sido usada com êxito para identificar equivalência semântica é o Mapa Auto-Organizável (SOM). Esta pesquisa objetiva descobrir, de modo semi-automatizado, equivalência semântica entre atributos de bases de dados, contribuindo para o gerenciamento e integração das mesmas. O resultado da pesquisa gerou uma sistemática para o processo de descoberta e uma ferramenta que a implementa.

Palavras-chave: Descoberta de conhecimento em bancos de dados, Mineração de dados, Inteligência Artificial, Redes Neurais, Equivalência Semântica.

Discovering Semantic Equivalences on Attributes in Databases Using Neural Networks

ABSTRACT

With the increasing number of companies using database technologies, the database's administrators create new schemes at every moment, and in most cases there are no normalization or formal procedures to do this task in a homogeneous form, it results in incompatible databases, that difficult data exchange. When the Database Systems (DBS) are projected and implemented independently, it is normal that data incompatibilities among different DBS. Problems related to the names of the attributes, storage in different measurement units, different levels of detail, different attributes with the same name or equal attributes with different names, different type of data, size, precision, etc, can be cited as main conflicts existing in the DBS schemes. These problems compromise the quality information and generate higher costs regarding the data maintenance. These problems arise as the consequence of redundant attributes' specification. These facts have caused great interest in discovering knowledge in database to identify information semantically equivalent stored in schemes. The process capable to discover this knowledge in database is called KDD (Knowledge Discovery in Database). The available tools to do KDD tasks are generic and derived from other areas of knowledge, in special, statistics and artificial intelligence. The artificial neural networks (ANN) have been used in systems which aim is the identification of previously unknown patterns. These networks can learn similarities among the data directly from instances, without a priori knowledge. An ANN that has been used with success to identify semantic equivalence is the Self-Organizing Map (SOM). This research aims to discover, in a semi-automatic way, semantic equivalence on database attributes, contributing for the management and integration of these databases. This work resulted in a systematic for the discovery process and a tool that implements it.

Keywords: Knowledge discovery from databases, Data mining, Artificial Intelligence, Neural Network

1 INTRODUÇÃO

Várias organizações utilizam bancos de dados em diferentes plataformas, incluindo computadores de grande porte (*mainframes*), estações de trabalho (*workstations*) e servidores configurados para uma rede corporativa. Isto estimula a descentralização das informações dentro de uma organização e induz a uma proliferação de bancos de dados. Geralmente, estes bancos de dados obedecem a diferentes conjuntos de requisitos para modelar objetos idênticos ou similares. Isto possibilita o surgimento de informações relacionadas, bem como a presença de informações redundantes nos diferentes bancos de dados (VILAS BOAS, 2002).

O acesso de forma integrada destas informações é um problema em muitos domínios de aplicação, como por exemplo, integração de dados, *e-business*, *data warehousing* (RAHM; BERNSTEIN, 2001b). Um dos problemas existentes para uso integrado de informações, se refere ao conflito em relação ao nome do atributo. Os atributos podem ter sido especificados com nomes diferentes para armazenar uma mesma informação em diferentes esquemas, ocasionando a imprecisão dos dados, quando acessado de forma integrada. A imprecisão destas informações tem provocado grande interesse em descobrir conhecimento em banco de dados (LI; CLIFTON, 2000). Neste cenário, a mineração de dados (*Data Mining*) surgiu como uma ferramenta útil para descoberta de equivalência semântica. O processo capaz de descobrir este conhecimento em banco de dados denomina-se KDD (*Knowledge Discovery Database*) (FAYYAD et al, 1996).

O processo de Descoberta de Conhecimento em Bases de Dados (DCDB) não é um processo trivial (FAYYAD et al, 1996; ENGEL, 2001). Este processo consiste em identificar padrões válidos, não conhecidos em informações úteis e interpretáveis. A Mineração de Dados (MD) é uma das fases da DCDB, ela incorpora técnicas estatísticas e de inteligência artificial, capazes de fornecer respostas a questões até então desconhecidas, ou mesmo descobrir novos conhecimentos em bancos de dados. MD é especialmente útil em casos onde não se conhece a pergunta, mas, mesmo assim, existe a necessidade de respostas. Para análise de grandes volumes de dados e exploração do valor destes dados, ou seja, a informação neles contida implicitamente, a mineração de dados faz uso de técnicas como Regras de Associação (SRIKANT; AGRAWAL, 1996a), Classificação (HAN et al, 1993), *Clustering* (CARVALHO, 2001; ENGEL, 2001), entre outras.

As redes neurais (RNA) têm sido utilizadas em sistemas cujo propósito é a identificação de padrões, antes desconhecidos. Estas podem aprender similaridades entre os dados, diretamente de suas instâncias, sem conhecimento a priori. Diferentemente das soluções tradicionais, as redes neurais são treinadas e não programadas. A saída natural de uma rede neural é o rótulo da classificação ou do agrupamento (*cluster*).

O objetivo principal da tarefa de identificação de agrupamentos (*clustering*) é separar objetos ou observações em classes naturais de forma que os elementos

pertencentes a um mesmo grupo tenham um alto grau de semelhança ou similaridade, enquanto que, quaisquer elementos pertencentes a grupos distintos, tenham pouca semelhança entre si (ENGEL, 2001).

Um tipo de RNA que tem sido usado com êxito para identificar equivalência semântica é o Mapa Auto-Organizável - SOM (KOHONEN, 1990; LI; CLIFTON, 2000; VARGAS, 2004). O SOM é um tipo de RNA competitiva, cuja função principal é mapear os dados de entrada de dimensão d numa grade de dimensão d', $d' < d$, de forma que esta grade de neurônios, totalmente conectada ao vetor de entrada pelos vetores de pesos, represente, de maneira topologicamente ordenada, os dados de entrada (KOHONEN, 1989; KOHONEN, 1990). Ou seja, o SOM identifica, nos padrões de entrada, os agrupamentos e os correlaciona a regiões específicas da grade de neurônios. É sobre esta propriedade de ordenação topológica dos agrupamentos gerados pelo SOM que surgem aplicações deste para descoberta de equivalência semântica entre dados.

1.1 Motivação

Com o crescimento das empresas que fazem uso das tecnologias de bancos de dados, os administradores destes bancos de dados criam novos esquemas a cada instante e não existe uma normalização ou formalização para que tal tarefa seja desempenhada de forma homogênea, resultando assim, em bases de dados incompatíveis, o que dificulta a troca de dados entre as mesmas (ELMASRI; NAVATHE, 2000). Um esquema é um conjunto de objetos tais como entidades, atributos, relacionamentos, que descrevem uma base dados.

Quando um administrador de dados cria um esquema, ele agrega ao mesmo, informações subjetivas que normalmente são interpretadas pelo próprio administrador ou por aplicações desenvolvidas especificamente para interpretarem e manterem estes esquemas. Na criação de um esquema é possível ser criado um atributo cujo domínio seja {M, F} e seu nome seja ‘sexo’ e em um outro esquema este atributo se encontra com o mesmo domínio {M,F}, porém com o nome de ‘gênero’. Este tipo de problema vem sendo alvo de pesquisas nos últimos anos, principalmente quando o objetivo é a integração de esquemas (CARRIÓN, 1999; LI; CLIFTON, 2000; RAHM; BERNSTEIN, 2001b).

Existem pesquisas e metodologias com o objetivo de solucionar o problema de integração semântica em esquemas de bancos de dados heterogêneos tais como (LEE; MOON, 1993; SPACCAPETRA; PARENT; DUPONT, 1992). Spaccapetra (1994) divide o processo de integração em duas fases: a primeira fase, chamada fase de investigação, consiste em determinar associações e discrepâncias entre esquemas. A segunda consiste na integração semi-automática. O que se percebe nestas pesquisas é que, na maioria das vezes, adotam um esquema conceitual global, no qual toda a informação disponível fica representada de maneira única e não redundante. Uma forma de se obter este esquema conceitual global é através de um mediador que pode ser um software ou uma pessoa ou, na maioria dos casos, ambos.

Quando os Sistemas de Bancos de Dados (SBD) são projetados e implementados independentemente, é normal que existam incompatibilidades entre os dados de diferentes SBD. O importante é identificar as incompatibilidades para fazer os tratamentos necessários. Alguns tipos de incompatibilidade que podem ocorrer são (CARRIÓN, 1999): 1) na integração de esquemas podem ocorrer conflitos de esquemas que envolvem tabelas; 2) o mesmo atributo é armazenado em diferentes unidades de medida; 3) diferentes níveis de detalhes; 4) atributos diferentes com mesmo nome ou atributos iguais com nomes diferentes; e 5) tipo de dado, tamanho etc.

A redundância na especificação de atributos, quando estes fazem parte da chave primária (HEUSER, 2000), faz-se necessária em casos como o deslocamento de chaves, ou seja, as chaves estrangeiras que referenciam outra tabela. Há também casos de decisão de projeto, como por exemplo a necessidade de aumento de performance (OZSU; VALDURIEZ, 1999). Neste último caso, um atributo com o nome do cliente pode existir em uma tabela que está inserida em um banco de dados local e em outro banco de dados remoto.

A imprecisão na informação devido à existência de mais de uma fonte de dados, uma maior necessidade na capacidade de armazenamento e processamento, um custo maior na manutenção das bases de dados, são conseqüências de atributos especificados de forma redundante.

Esta pesquisa visa desenvolver um a método para descoberta de equivalência semântica entre os atributos criados de forma redundante em bancos de dados, bem como a implementação de uma ferramenta que utilize esta metodologia.

1.2 Objetivos do trabalho

1.2.1 Geral

Especificar uma sistemática para descoberta de equivalência semântica entre atributos e implementar um protótipo de uma ferramenta para análise de esquemas de banco de dados que possibilite identificar de modo semi-automático possíveis duplicações semânticas de atributos.

1.1.1 Específico

O objetivo principal deste trabalho é o estudo das técnicas de descoberta de conhecimento em bancos de dados, e aplicação das mesmas para descoberta de equivalência semântica entre atributos de bancos de dados.

Os objetivos específicos são:

- Avaliar esquemas nos bancos de dados da Universidade Estadual de Londrina e identificar equivalência entre atributos nestes esquemas;
- Estudar algumas abordagens do processo de descoberta de conhecimento em bancos de dados com a finalidade de escolher a que melhor se adapte ao estudo de caso;
- Aplicar uma ou mais técnicas de redes neurais;
- Avaliar as técnicas utilizadas;
- Analisar os possíveis resultados.

1.2 Contribuições

Esta pesquisa contribui na resolução de problemas relacionados à redundância de informações encontradas pelos DBA em SGBD.

- Uma sistemática para a descoberta de equivalência semântica entre atributos de bancos de dados, que seja independente do banco de dados e que possa aplicar uma ou mais técnicas que utilizem redes neurais;

- Uma ferramenta para auxiliar no processo de descoberta de equivalência semântica em bases de dados;
- Validação da sistemática e da ferramenta proposta através de um estudo de caso, envolvendo SGBD utilizados na Universidade Estadual de Londrina;
- Estudo de ferramentas relacionadas à ferramenta SEA que tratam de equivalência semântica.

1.3 Organização do texto

Este trabalho está organizado em 7 capítulos como segue:

O capítulo 1 faz uma breve introdução ao problema, apresenta os objetivos e relata as contribuições deste trabalho.

O capítulo 2 aborda uma revisão bibliográfica sobre os trabalhos relacionados e uma comparação entre os mesmos.

O capítulo 3 fala sobre equivalência semântica, tratando do assunto do ponto de vista de banco de dados.

O capítulo 4 aborda o processo de descoberta de conhecimentos, apresentando as suas fases, tarefas e técnicas de mineração de dados, tais como árvore de decisão, indução de regras, análise de séries temporais, redes neurais artificiais.

O capítulo 5 apresenta uma sistemática para descoberta de equivalência semântica de atributos.

O capítulo 6 apresenta o estudo de caso o qual mostra testes realizados na Universidade Estadual de Londrina.

O capítulo 7 apresenta as conclusões referentes a trabalho e sugere novas direções para a expansão da pesquisa apresentada.

2 TRABALHOS RELACIONADOS

Combinação de esquemas é a tarefa de encontrar correspondência semântica entre elementos de dois esquemas. Este problema necessita ser resolvido em muitas aplicações, como por exemplo, para integração de dados e mapeamento de mensagens XML *e-busines*. Nos sistemas atuais, o mapeamento de esquemas é manual; consome um alto tempo e um processo tedioso torna impraticável o mapeamento de um grande número de esquemas (fontes de dados, mensagem em formato XML). Vários sistemas têm sido desenvolvidos recentemente para determinar esquemas correspondentes (semi-) automáticos, como por exemplo, Autoplex (BERLIN; MOTRO, 2001), Automatch (BERLIN; MOTRO, 2002), Clio (YAN et al, 2001; MILLER et al, 2001), COMA (RAHM; BERNSTEIN, 2001a; DO; RAHM, 2002), Cupid (MADHAVAN; BERNSTEIN; RAHM, 2001), Delta (CLIFTON; HOUSMAN; ROSENTHAL, 1996), DIKE (PALOPOLI; TERRACINA; URSINO, 2003), GLUE (DOAN et al, 2002), LSD (DOAN; DOMINGOS; HELEVY, 2001), MOMIS e ARTEMIS (BERGAMASCHI et al, 2001), SemInt (LI; CLIFTON, 1994; LI; CLIFTON, 2000), MUVIS (LI; CLIFTON, 2000), SemMa (SUN, 2003), SKAT (MITRA; WIEDERHOLD; JANNINK, 1999), Similarity Flooding (SF) (MELNIK; GARCIA-MOLINA; RAHM, 2002), e TranScm (MILO; ZOHAR, 1998).Enquanto muitas destas técnicas sugiram do contexto de uma aplicação específica, algumas (Clio, COMA, Cupid e SF), tentam resolver o problema de combinação de esquemas de uma forma genérica, que pode ser utilizada para diferentes aplicações e tipos de esquemas.

2.1 SemInt (SEMantic INTegrator)

SemInt é uma ferramenta para identificar atributos correspondentes em banco de dados heterogêneos. SEMantic INTegrator (SemInt) é uma ferramenta baseada em rede neural para identificar atributos correspondentes em esquemas de banco de dados heterogêneos. Para cada atributo, o SemInt determina valores no intervalo [0,1] através de critérios pré-estabelecidos. Estes valores são usados, primeiramente, para aglomerar atributos similares do primeiro esquema e encontrar similaridades com o segundo esquema. O processo de classificação é realizado por uma rede neural com um treinamento automático, reduzindo, assim, o esforço para se encontrar as combinações de atributos. O resultado da combinação consiste em conjuntos de atributos similares de ambos os esquemas da entrada, conduzindo a uma cardinalidade m:n.

2.2 SemMa (semantic-matching)

SemMa constrói uma base de dados interna para um dicionário de sinônimos e usa um dicionário externo para descobrir o significado semântico dos termos dos esquemas das bases de dados. O acesso à semântica (significado) dos termos da base de dados

permite ao sistema automaticamente mapear atributos entre uma base de dados de origem e uma base de dados de destino, que são semanticamente equivalentes, deste modo o SemMa provê automação no processo de combinação de esquemas.

2.3 Cupid

Cupid é um algoritmo genérico de combinação de esquemas que descobre mapeamentos entre elementos de esquemas baseado em seus nomes, tipos de dados, restrições e estrutura dos esquemas. O enfoque do algoritmo consiste em calcular coeficientes de similaridade entre elementos de dois esquemas e então criar um mapa destes coeficientes.

As entradas para o algoritmo são dicionários de sinônimos, bibliotecas de mapas conhecidos, etc. O cálculo dos coeficientes é feito em dois estágios chamados: mapeamento lingüístico e mapeamento estrutural. Ele envolve normalização e clusterização de elementos dos esquemas em categorias, para reduzir o número de elementos de comparação. A comparação consiste em calcular a similaridade lingüística de cada par de elementos de categorias compatíveis, resultando em uma tabela de coeficientes de similaridade lingüística entre os elementos dos dois esquemas comparados. A geração do mapa envolve a escolha de pares de elementos com pesos máximos de similaridade.

Apesar de ser considerado um algoritmo mais genérico o Cupid foi desenvolvido mais especificamente para trabalhar com similaridade entre elementos organizados em esquemas hierárquicos, portanto ele adapta-se melhor a estruturas como XML.

2.4 SKAT (Semantic Knowledge Articulation Tool)

A SKAT segue o enfoque baseado em descoberta semi-automática de regras associativas entre duas ontologias. Esta técnica é relevante para XML e os esquemas que são transformados em grafos. As regras são formuladas em lógica de primeira ordem e expressam a combinação de relacionamentos e métodos são definidos para derivar uma nova combinação.

A SKAT suporta a combinação de nomes e combinação estrutural simples, baseada na estrutura é-um.

2.5 SF (Similarity Flooding)

SF converte esquemas (SQL DDL, RDF, XML) em grafos rotulados e usa computação de ponto fixo para determinar correspondência de cardinalidade 1:1 local e m:n global entre nós correspondentes do grafo. O algoritmo tem empregado uma combinação híbrida com uma simples combinação de nomes, a qual sugere um mapeamento inicial no nível de elemento para alimentar o SF. Diferentemente de outros sistemas para combinação de esquemas, SF não explora o uso de dicionários externos. Como último passo, vários filtros podem ser especificados para selecionar um subconjunto do conjunto de combinações produzido pelo SF.

2.6 LSD (Learning Source Descriptions) and GLUE

O sistema LSD usa a técnica de aprendizado de máquina para combinar uma nova origem de dados a um esquema global previamente determinado, produzindo um

mapeamento 1:1. Esta técnica foi desenvolvida principalmente para ser utilizada com esquemas XML.

O GLUE, que é uma extensão do LSD, combina diretamente duas origens de dados. As duas técnicas atuam de forma semelhante. Além de combinações de nomes, elas utilizam várias combinações no nível de instância. Durante a fase de aprendizado são descobertas diferentes características dos padrões das instâncias.

2.7 TranScm

O TranScm usa mapeamento de esquemas para gerar uma tradução de dados automática entre instâncias de esquemas.

Esquemas de entrada são transformados em grafos rotulados, que é a representação de esquema interno. Todas as outras informações do esquema (nome, obrigatoriedade, número de filhos, etc) são representados como propriedades dos nodos. O mapeamento é realizado nodo por nodo (nível de elemento, 1:1) começando pelo topo e presumindo um alto grau de similaridade entre os esquemas.

Existem vários mapeadores que são verificados em uma ordem fixa. Eles exigem que o mapeamento seja determinado por exatamente um mapeador por par de nodos. Se nenhuma combinação for encontrada ou se o mapeador determinar múltiplos candidatos, a intervenção do usuário é requerida, para inserir uma nova regra (mapeador) ou selecionar um candidato.

Os mapeadores tipicamente consideram múltiplos critérios podendo assim representar abordagens híbridas. Por exemplo, um dos mapeadores testa as propriedades nome e número de filhos. O mapeamento de nodos pode ser criado dependente do mapeamento parcial ou total de nodos descendentes.

O mapeamento ao nível de esquema é:

- Baseado em nomes: igualdade dos nomes, sinônimos, homônimos, hipernônimo;
- Baseado em Restrições: é-um, cardinalidade de relacionamentos;
- Mapeamento de estrutura: similaridade, com respeito a elementos relacionados.

Sua aplicação é na área de tradução de dados.

2.8 DIKE

DIKE tem como objetivo encontrar pares de objetos em dois esquemas, que são similares. Estes esquemas têm os mesmos atributos e relacionamentos, porém são de tipos diferentes (entidades, atributos, relacionamentos). O algoritmo se baseia num conjunto de sinônimos e homônimos especificados pelo usuário. A técnica requer o agrupamento de objetos em subconjuntos e a produção de um esquema abstrato obtido pela substituição de cada subconjunto por um objeto simples.

A similaridade entre os dois objetos é representada por um valor entre 0 e 1; quando a similaridade exceder um determinado percentual, eles são considerados coincidentes.

A intervenção do usuário é necessária para definir os pesos para verificar a semelhança entre os esquemas.

Os passos envolvidos são:

1. Enriquecimento da descrição do esquema, obtido pela extração semi-automática de conhecimento;
2. Exploração das propriedades entre os esquemas;

3. Exploração do repositório criado, para criação de uma *data warehouse* sobre os dados disponíveis.

O sistema DIKE integra múltiplos esquemas pela exploração do princípio de que a similaridade dos elementos do esquema depende da similaridade dos elementos em seus vizinhos. A relevância dos elementos e inversamente proporcional a sua distância em relação aos elementos comparados, então elementos vizinhos influenciam-se mais que elementos distantes.

2.9 Autoplex e Automatch

Autoplex e seu aperfeiçoamento Automatch representam técnicas de combinação de esquemas baseadas em aprendizado de máquina. Em particular, o aprendizado das redes Naive Bayesian, que, neste caso, exploram as características das instâncias para combinar atributos de uma base de dados relacional com um esquema global previamente construído. Para cada atributo de origem, a probabilidade de combinação é calculada, bem como a probabilidade de não combinação. Estas probabilidades são normalizadas para somar 1 e a probabilidade de similaridade retorna a similaridade entre o atributo de origem e o atributo global. O resultado das combinações consiste em uma correspondência de cardinalidade 1:1 local e global.

2.10 Artemis

O Artemis é um sistema para mapear a integração de esquemas de modelos relacionais e orientados a objetos (OO) de forma híbrida. A granularidade é tratada em nível de elemento, estrutura, entidades, relacionamentos e atributos, sendo sua cardinalidade de 1:1.

O sistema Artemis determina a afinidade entre atributos de dois esquemas. A afinidade é baseada na comparação do nome dos atributos, estrutura e nos tipos de domínio e são escalonadas em um intervalo de [0,1]. O processo requer um dicionário de sinônimos para determinar o relacionamento semântico. O sistema usa o agrupamento hierárquico baseado na afinidade dos valores dos atributos. Finalmente, um conjunto de regras é empregado. Estas regras são obtidas através da interação com um usuário. Ao contrário do Automatch, o Artemis considera informações do esquema. O Automatch considera informações das instâncias, entretanto o conhecimento no Artemis é pré-codificado em dicionários e regras de unificação e no Automach é aprendido de exemplos.

O Artemis é usado em um mediador chamado MOMIS (BENEVENTANO et al, 2001), o qual integra esquemas independentes em um esquema global.

2.11 Clio

Clio é um sistema para gerenciamento e facilitação de tarefas complexas de transformação e integração de dados heterogêneos. O Clio não faz a integração de esquemas, no entanto ele suporta a geração e gerenciamento de esquemas, correspondências entre esquemas e mapeamento (consultas) entre esquemas.

O uso típico do Clio começa com o usuário carregando um ou mais esquemas no sistema. Estes esquemas são lidos de uma base de dados Objeto-Relacional ou de arquivos XML associados com XML *schema*. O módulo de esquema é utilizado para aumentar o esquema com informações adicionais, se necessário. O Clio faz uso de metadados como consultas armazenadas e definições de *views*. Por exemplo, na

ausência de restrições declaradas, o sistema procura pela possibilidade de existência de chaves e chaves estrangeiras. Finalmente, os esquemas são verificados pelo usuário para assegurar a validade das informações geradas. Por exemplo, uma chave estrangeira descoberta ou a inclusão de dependência pode não estar correta e o Clio permite que correções possam ser realizadas pelo usuário.

Para facilitar o processo, o Clio faz uso de uma interface gráfica para comunicar-se com o usuário (MILLER et al, 2001). No modo de visualização de esquemas o usuário vê a representação dos esquemas, inclusive as informações adicionais geradas. Este modo de visualização pode ser utilizado para alterar informações sobre os esquemas. Adicionalmente, o Clio provê um modo de visualização de dados, através do qual o usuário pode ver alguns exemplos de dados do esquema, para ajudá-lo a entender o esquema.

2.12 COMA (COmbining MAtch)

COMA segue uma composição de técnicas, as quais provêm uma biblioteca de diferentes formas de encontrar equivalência. Ele explora informações dos esquemas, como elementos e propriedades estruturais. Um modo especial de encontrar combinações e utilizar resultados prévios de outras combinações. No COMA são utilizadas combinações de diferentes estratégias para resolver o problema de combinação de esquemas, como agregação de combinações e seleção de candidatos a combinações. Os esquemas são transformados em um grafo acíclico direcionado, no qual todos os algoritmos de combinação trabalham. Cada elemento do esquema é identificado de forma única por seu caminho completo da raiz do grafo até o nó correspondente. O COMA produz combinações ao nível de elemento (atributo) com cardinalidade 1:1 local e m:n global.

2.13 Delta

DELTA converte metadados disponíveis sobre os atributos em uma *string* de texto, e submete este texto a uma ferramenta comercial para obtenção de informações em documentos texto.

O integrador, que é um papel desempenhado por uma pessoa, faz as seguintes tarefas: seleciona qual atributo irá investigar e em relação a qual base de dados investigar; para encontrar uma combinação aceitável com a base de dados D em relação ao atributo A, o integrador cria um padrão de busca para A em relação ao documento D. A ferramenta retorna uma lista com combinações ordenadas das melhores até as piores. O integrador (humano) examina a do topo da lista e talvez declare esta como selecionada. Se necessário, o integrador pode usar o resultado para criar um novo padrão de investigação.

2.14 MUVIS (Multi-User View Integration System)

MUVIS é um sistema baseado em conhecimento, para integração de objetos. Ele auxilia os desenvolvedores de bases de dados na representação das visões de usuários e integração destas visões em uma visão conceitual global. O MUVIS determina o grau de similaridade e não similaridade entre dois objetos durante a fase de pré-integração. A similaridade e não similaridade no MUVIS é primeiramente baseada na comparação dos nomes dos atributos. A equivalência de objetos e determinada pela comparação de aspectos de cada um (como nome da classe, nome dos membros e nome dos atributos) e

computa um peso para similaridade e não similaridade. É, então, produzida uma recomendação de como a integração deve ser feita.

A maioria das ferramentas de automação desenvolvidas para auxiliar os criadores de bases de dados a encontrar correspondência entre atributos, que trabalham com nomes de atributos, trabalha bem com homônimos (mesmos nomes e dados diferentes).

2.15 Comparação

A combinação de esquemas está intimamente ligada à descoberta de equivalência semântica entre atributos. Esta descoberta é uma das etapas de ferramentas como SemInt, LSD e CUPID. A Tabela 8.2 mostra características e comparações entre algumas das ferramentas citadas em publicações, relacionadas a combinação de esquemas. São analisados itens como o tipo de esquema suportado, o formato de representação interna dos metadados, tarefas que devem ser executadas manualmente, área de aplicação entre outros. Deste modo, é possível observar o grau de automação destas ferramentas e também o quanto elas podem ser genéricas em seu uso. Exceto o LSD, todas as outras ferramentas suportam técnicas híbridas de combinação de esquemas. Muitas das ferramentas suportam descobrem equivalência em nível de esquema e em nível de atributo, sendo este último na maioria em relação ao nome ou a restrições. Apenas duas consideram o valor das instâncias dos atributos. Varias foram desenvolvidas com um propósito específico. A CUPID é uma das ferramentas que tem um propósito genérico. Várias suportam vários tipos de esquema, enquanto a LSD suporta somente XML e a DIKE está limitada ao modelo ER. Todos os sistemas permitem ao usuário validar os resultados e requerem a interferência do mesmo. A principal forma de reuso do conhecimento é através de dicionários e glossários.

3 EQUIVALÊNCIA SEMÂNTICA

Nos últimos anos, o número de sistemas que requerem acessos integrados a múltiplas fontes de informação heterogêneas, autônomas e distribuídas tem aumentado consideravelmente. Integrar informações de múltiplas fontes é uma tarefa complexa (VILAS BOAS, 2002). Uma das razões para tal complexidade é a heterogeneidade semântica (ou relativismo semântico) existente entre as fontes a serem integradas. A equivalência semântica expressa a multiplicidade de possíveis representações de um mesmo conceito do mundo real, dificultando a identificação das várias correspondências que existem entre conceitos similares ou relacionados em diferentes esquemas (Missier; Rusinkiewicz apud VILAS BOAS, 2002).

A existência da mesma informação em mais do que um registro ou em mais do que um campo da base de dados pode criar problemas graves para as operações de atualização. Se a mesma informação existir em vários locais, é fundamental que os processos de atualização dessa informação atinjam sempre todos esses locais, sob pena da base de dados ficar inconsistente e das operações de consulta retornarem eventualmente informação errada. Embora pareça intuitivamente simples, a identificação do que é informação repetida não resulta de um processo trivial, dependendo fundamentalmente da semântica ou significado dos nomes dos vários objetos informativos. Esta semântica deve tornar-se objetiva, para ser discutida sem ambigüidades, e pode ser capturada em grande parte com recurso a dependências funcionais, e a outras restrições de integridade sobre os atributos de informação (CUNHA, 2001).

Descoberta de equivalência semântica é o processo de descobrir o mapeamento entre dois grafos com a aplicação de algoritmos de combinação. A idéia chave está na exploração dos nós que representam conceitos e dos rótulos dos arcos que representam relacionamentos entre estes conceitos (GIUNCHIGLIA; SHVAIKO, 2003). Existe uma relação semântica entre os nós. Se estes forem entendidos como conjuntos de conceitos, então as relações semânticas possíveis entre os nós são de equivalência, sobreposição, dissimilaridade, contém e está contido.

3.1 Dependência Funcional

Segundo (HEUSER, 2000), em uma tabela relacional, diz-se que uma coluna C_2 depende funcionalmente de uma coluna C_1 (ou que a coluna C_1 determina a coluna C_2) quando, em todas linhas da tabela, para cada valor de C_1 que aparece na tabela, aparece o mesmo valor de C_2.

Este conceito é formalizado da seguinte maneira: sendo R uma relação definida sobre o conjunto de atributos $A=\{A_1, A_2, ..., A_n\}$ e $X \subset A$, $Y \subset A$. Se para cada valor de X em R, há apenas um valor de Y associado, diz-se que X determina funcionalmente Y ou que Y é funcionalmente dependente de X (OZSU; VALDURIEZ, 1999; BELL, 1995).

A notação para esta dependência é $X \rightarrow Y$. A chave em uma relação determina funcionalmente os atributos não chave da mesma relação.

Quando um atributo depende funcionalmente de um conjunto de atributos então está dependência é dita Dependência Multivalorada (DMV) e sua notação é $X \rightarrow\rightarrow Z$. Formalmente, sendo R uma relação definida sobre o conjunto de atributos $A=\{A_1, A_2, ..., A_n\}$ e $X \subset A$, $Y \subset A$, $Z \subset A$. Se para cada valor de Z em R, há apenas um valor para o par (X,Y) e o valor de Z depende apenas do valor de X, diz-se que X multidetermina Z ou que Z é multidependente de X.

A dependência funcional (FD, do inglês *functional dependency*) pode ser classificada como **parcial** quando uma coluna depende apenas de parte de uma chave primária composta. A FD também é classificada como **transitiva** quando uma coluna, além de depender da chave primária da relação, depende de outra coluna ou conjunto de colunas da relação.

Para completar a definição de dependência funcional, é preciso definir a dependência funcional **total**: se Y depende funcionalmente de X, mas não depende funcionalmente de qualquer subconjunto X' de X (isto é, considerando X sem ao menos um dos atributos que o compõem), então Y é totalmente dependente de X. X é chamado também de **determinante funcional** de Y.

É impossível determinar a dependência funcional entre os atributos de uma relação sem conhecer o significado dos atributos. Em outras palavras, a dependência funcional é conseqüência da interpretação semântica dos valores armazenados nos elementos da relação.

3.2 Conflitos de semântica

Quando os Sistemas de Banco de Dados (SBD) são projetados e implementados independentemente, é normal que exista incompatibilidade entre os dados (CARRIÓN, 1999; RAHM; BERNSTEIN, 2001b). É importante identificar as incompatibilidades para fazer os tratamentos necessários. Abaixo são citados alguns tipos de incompatibilidades que são comuns nos SBD:

- O mesmo atributo é armazenado em diferentes unidades de medida;
- Nível de abstração (diferentes níveis de detalhes);
- Nome (atributos diferentes com mesmo nome ou atributos iguais com nomes diferentes);
- Tipo, Tamanho etc.

As diferentes representações de um mesmo conceito podem ocorrer, principalmente, porque projetistas têm diferentes percepções da realidade. Os tipos mais comuns de heterogeneidade semântica são a heterogeneidade terminológica e a extensional (VILAS BOAS, 2002):

- Heterogeneidade terminológica: expressa a diferença no significado ou na interpretação dos mesmos conceitos, conceitos relacionados ou conceitos diferentes representados nos esquemas a serem integrados (mapeamento de esquemas). As causas mais comuns para tal heterogeneidade são quando termos diferentes são usados para representar o mesmo conceito do mundo real (termos sinônimos) ou quando os mesmos termos são usados para representar conceitos diferentes do mundo real (termos homônimos). Por

exemplo, um esquema utiliza o termo estudante, um outro esquema, o termo aluno e um terceiro esquema, o termo universitário para armazenar informações sobre pessoas que estudam em universidades. Neste caso, termos sinônimos são usados para expressar o mesmo conceito do mundo real. Em um outro exemplo, o esquema do setor de produção de uma empresa utiliza o termo equipamento para manter informações sobre os equipamentos utilizados pela empresa, e o esquema do setor de vendas desta mesma empresa utiliza o termo equipamento para disponibilizar informações sobre os equipamentos vendidos pela empresa. Neste caso, termos iguais são usados para representar diferentes conceitos.

- Heterogeneidade extensional expressa diferença entre as coleções de objetos similares nas diferentes fontes de informação. Por exemplo, se existirem duas coleções estudante em duas fontes distintas, então estas coleções podem conter objetos correspondendo ao mesmo objeto do mundo real ou podem referenciar conjuntos disjuntos de objetos do mundo real. Identificar o relacionamento extensional entre as coleções de objetos de diferentes fontes é fundamental para a integração de informações.

3.2.1 Conflitos em atributos

Podem ocorrer diferenças de formato em relação aos atributos, que incluem diferenças de tipo, domínio, escala, precisão e combinação de itens. Por exemplo, um número de série pode ser definido como um inteiro em um SBD e como um texto em outro. Itens de dados podem estar separados em um SBD enquanto em outro podem estar combinados como uma única quantidade, por exemplo, em um SBD pode haver três tuplas para o valor de cada compra e em outro uma única tupla com o total das compras.

SBDH normalmente resolvem as diferenças entre formatos definindo funções de transformação entre a representação global e as locais. O problema nesta área é que as transformações da representação local para a global podem ser simples, contudo as transformações inversas (necessárias em atualizações) podem ser complexas.

Outro conflito que pode ocorrer na integração de esquemas está relacionado a diferenças estruturais, visto que dependendo de como um objeto é utilizado em um SBD, ele pode ser estruturado diferentemente em diferentes SBD. Um item de dado (atributo) pode possuir um valor simples em um SBD e multivalorado em outro. Um objeto pode ser representado como uma única relação em um SBD e como uma relação múltipla em outro. O mesmo item pode ser um valor em um local, um atributo em outro e uma relação em um terceiro local.

Os conflitos usuais que podem ocorrer, relacionados aos atributos são (CARRIÓN, 1999):

- **Conflitos de nomes de atributos**: os conceitos de sinônimo e homônimo aplicado aos conflitos de nomes de objetos (tabelas, atributos, etc) são aplicados a estes tipos de conflitos;
- **Conflitos em valores por ausência**: a definição implícita de alguns valores por ausência pode levar contradição na semântica dos dados;
- **Conflitos por restrições de valores dos atributos**:
    a) **Conflitos nos tipos de dados**: os tipos de dados podem diferir dependendo dos critérios de cada aplicação;

b) **Conflitos em restrição de domínio**: as características de cada atributo podem ser distintas em cada aplicação. Além do mais, cada modelo de informação estabelece restrições e possibilidades de estruturação diferente.

- **Conflitos por cardinalidade e grade de atomicidade:** as características de cada atributo podem ser distintas em cada aplicação. Cada modelo de informação estabelece restrições e possibilidades de estruturação diferentes.
- **Conflitos na representação da informação:** Pode acontecer de um mesmo conceito estar representado com uma entidade em uma aplicação e em outra aparecer como um atributo.

3.2.2 Conflitos em dados

A heterogeneidade semântica ocorre quando existe uma diferença de significado e interpretação em relação aos mesmos dados (SHETH; LARSON, 1990).

Os SBD que modelam os mesmos objetos do mundo real podem estar em conflito quanto aos valores dos dados. Um sistema pode não possuir a mesma informação armazenada por atualizações incompletas, erros no sistema ou demanda insuficiente para manter tais dados. O problema maior ocorre quando dois sistemas de banco de dados gravam o mesmo item de dados, porém com diferentes valores. Os valores podem diferir devido à ocorrência de erros ou porque existem diferenças semânticas válidas.

Os problemas relacionados a conflitos de dados também podem ocorrer, mesmo tendo esquemas de banco de dados equivalentes em relação à estrutura das tabelas e seus atributos.

Conflitos que podem ocorrer, relacionados aos dados (KIM; SEO, 1991):

- **Conflito entre os valores**: ocorre quando se espera que as instâncias equivalentes tenham os mesmos valores, e devido aos dados estarem armazenados de forma inconsistente são capturados de forma incorreta.
- **Deficiências em representação**: são situações de contexto e cultura organizacional, entre outros fatores que podem levar o banco de dados a recuperar uma representação distinta da informação, podendo citar:

  a. **Notação diferente**: quando existem diferentes formas para representar o mesmo dado, por exemplo, codificações numéricas e escalas de letras;

  b. **Unidades distintas**: a diversidade de unidades, principalmente para valores numéricos, traz consigo problemas de representação (por exemplo, o sistema métrico e em polegadas, para medidas);

  c. **Diferenças nas representações**: ocorrem quando existem diferentes formas para representar um valor de um atributo, por exemplo, estado da federação pode ocorrer de estar representado como UF, Estado, etc.

3.3 Equivalência semântica e atributos

A combinação de esquemas consiste em mapear dois grafos distintos através da aplicação de algoritmos de combinação. Sendo assim, a descoberta de equivalência semântica entre atributos consiste em descobrir equivalência entre dois ou mais nós

rotulados destes grafos, aplicando técnicas de combinação. As técnicas que trabalham com atributos dividem-se em dois grupos. O primeiro trata os atributos como sendo conceitos, considerando suas instâncias, e o segundo trata os atributos como sendo rótulos dos nós, formados por conjuntos de caracteres. O segundo grupo pode ser considerado como técnicas com semântica fraca, pois estão sujeitas a grandes falhas. Dentre as técnicas se encontram: análise de *strings*, análise de tipos de dados, análise da pronuncia da *string,* dicionários pré-compilados, rede de palavras.

3.4 Taxionomia de equivalência semântica

Nesta seção serão classificadas as principais técnicas para combinação ou equivalência de esquemas. A Figura 3.1 mostra um diagrama de classificação destas técnicas. Algumas implementações de técnicas de combinação de esquemas usam múltiplos algoritmos de combinação, isto permite que se escolha o algoritmo que melhor se aplica ao domínio do problema em questão. A aplicação de vários algoritmos tem duas correntes: a primeira aplica técnicas que trabalham em conjunto, são chamadas híbridas e a segunda aplica várias técnicas também, porém os resultados são individuais e devem ser combinados ou escolhidos.

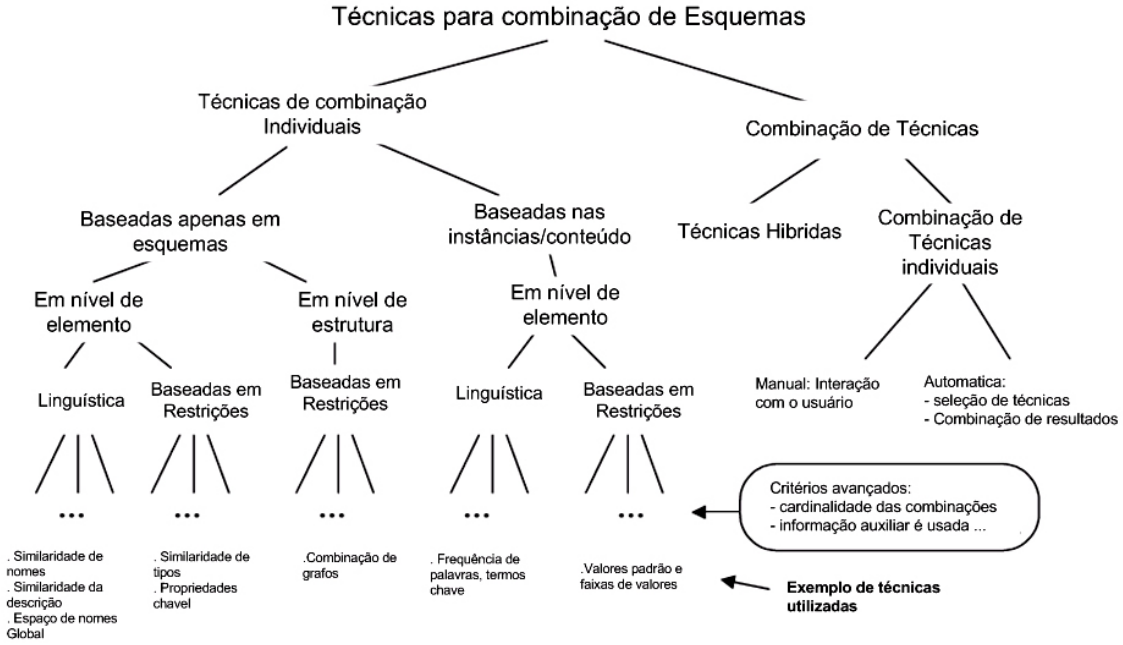

Figura 3.1: Classificação das técnicas para combinação de esquemas

As técnicas que não utilizam múltiplos algoritmos são chamadas técnicas de combinação individuais. Para estas técnicas podem ser considerados os seguintes critérios:

- Instância x esquema: as técnicas podem considerar os dados (ocorrências ou instâncias) ou apenas informações em nível de esquema;
- Atributo x estrutura: as técnicas podem ser aplicadas sobre atributos dos esquemas ou sobre a combinação de elementos dos esquemas como relacionamentos, tabelas, etc;
- Lingüística x restrições: as técnicas podem fazer uso de algoritmos baseados em lingüística, por exemplo, os baseados em nomes ou descrições dos

atributos ou elementos dos esquemas, e podem utilizar algoritmos baseados em restrições, como por exemplo, relacionamentos, chaves, etc;

- Cardinalidade da combinação: cada elemento da combinação pode estar relacionado a um ou mais elementos ou um ou mais elementos podem estar relacionados a um outro elemento. A cardinalidade pode ser 1:1, n:1, n:1 e n:m;
- Informação auxiliar: muitas técnicas fazem uso não somente dos dados dos esquemas (atributos, restrições, etc) como também de dados externos como dicionários, esquemas globais, equivalências prévias, usuário.

4 DESCOBERTA DE CONHECIMENTO EM BANCO DE DADOS

O termo, Descoberta de Conhecimento em Base de Dados (DCBD), ou *Knowledge Discovery in Databases* (KDD), foi desenvolvido com a finalidade de encontrar informações úteis em grandes bases de dados, as quais não estão visíveis ao ser humano. Esse processo tem por objetivo a extração do conhecimento implícito e previamente desconhecido e a busca de informação potencialmente útil dos dados (FAYYAD et al, 1996).

Atualmente, tem-se uma quantidade enorme de dados armazenados em bases de dados, cujo tamanho cresce rapidamente. O curioso é que estes dados proporcionalmente não oferecem as informações estratégicas e necessárias para as tomadas de decisões das organizações. Neste contexto, surge a descoberta de conhecimento em bancos de dados. A DCBD é um processo analítico, exploratório, e iterativo que permite identificar, em banco de dados, tendências e padrões que sejam:

- Válidos (estatisticamente significativos e confiáveis);
- Novos (não trivialmente previsíveis por simples bom senso);
- Úteis (permitem alterar decisões no domínio das aplicações);
- Interpretáveis (suficientemente abstratos e concisos para enriquecer o modelo de um analista humano).

4.1 O Processo de Descoberta de Conhecimento

Descobrir conhecimento significa extrair de grandes bases de dados, sem nenhuma formulação prévia de hipóteses, informações genéricas, relevantes e previamente desconhecidas, que podem ser utilizadas para a tomada de decisões. Associado à descoberta de conhecimento, deve existir um processo de descoberta de conhecimento (CARVALHO, 1999).

O processo de DCBD é um conjunto de atividades contínuas que compartilham o conhecimento descoberto a partir de bases de dados (FAYYAD et al, 1995; FAYYAD et al, 1996). A Figura 4.1 ilustra os passos que constituem o processo de DCBD:

No processo de DCBD cada fase pode possuir uma interseção com as demais. Desse modo, os resultados produzidos numa fase podem ser utilizados para melhorar os resultados das próximas fases. Esse cenário indica que o processo de DCBD é iterativo (FAYYAD et al, 1996; ENGEL, 2001), buscando sempre aprimorar os resultados a cada iteração.

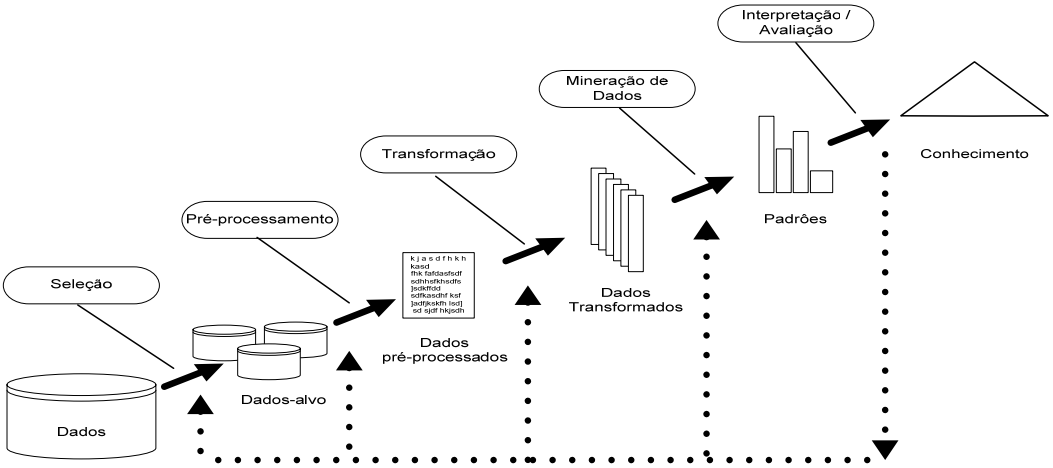

Figura 4.1: Visão geral das fases que compõem o processo de DCBD

O desenvolvimento de sistemas de DCBD está relacionado com diversos domínios de aplicações: marketing, análises corporativas, astronomia, medicina, biologia, entre outros (CARVALHO, 2001). Deste modo, pode-se identificar diversas tarefas de DCBD que são, principalmente, dependentes do domínio da aplicação e do interesse do usuário. De modo geral, cada tarefa de DCBD extrai um tipo diferente de conhecimento do banco de dados, logo, cada tarefa requer um algoritmo diferente para a extração de conhecimento.

4.1.1 Fases do processo de DCBD

A seguir são descritas as fases que compõem o processo DCBD apresentado na Figura 4.1.

4.1.1.1 Seleção

Na fase de seleção o objetivo é a criação do conjunto de dados envolvendo as variáveis necessárias, selecionando ou segmentando os dados de acordo com algum critério. Os conjuntos de dados são chamados de dados alvo.

Apenas os dados relevantes são selecionados do conjunto de atributos do banco de dados. Em resumo, a seleção de atributos consiste da escolha de um subconjunto de atributos disponíveis para o algoritmo de DCBD que seja relevante para o objetivo da tarefa. O subconjunto selecionado é então fornecido para o algoritmo de mineração dos dados. Uma motivação para essa seleção é otimizar o tempo de processamento do algoritmo minerador, visto que ele apenas trabalhará com um subconjunto de atributos, desse modo diminuindo o seu espaço de busca. Existem dois principais métodos para a seleção de atributos: múltiplas iterações e iteração simples. No método de múltiplas iterações o dado a ser minerado é dividido em dois conjuntos: treinamento e avaliação, então uma busca heurística iterativa é executada no espaço dos subconjuntos dos atributos. Cada iteração dessa busca consiste basicamente de três passos. Primeiro, um subconjunto de atributos, chamados atributos candidatos, é selecionado de acordo com algum critério. Segundo, um algoritmo é aplicado no subconjunto de treinamento, composto apenas por atributos candidatos. Terceiro, é medida a qualidade da seleção aplicando-se o resultado obtido no subconjunto de avaliação. Esse ciclo permanece até que se chegue a um resultado satisfatório. Já o método de iteração simples é independente do algoritmo minerador, ou seja, o mesmo conjunto de atributos selecionados pode ser fornecido para qualquer algoritmo minerador.

4.1.1.2 Pré-processamento

Essa fase é responsável por consolidar as informações relevantes para o algoritmo minerador, buscando reduzir a complexidade do problema, inclui duas subfases chamadas: limpeza dos dados e transformação ou codificação dos dados, que são descritas a seguir (LOPES, 1999):

- **Limpeza dos dados**: envolve uma verificação da consistência das informações, a correção de possíveis erros e o preenchimento ou a eliminação de valores nulos e redundantes. Nessa fase são identificados e removidos os dados duplicados e corrompidos. A execução dessa fase corrige a base de dados eliminando consultas desnecessárias que seriam executadas pelo algoritmo minerador e que afetariam o seu processamento. Os métodos de limpeza dos dados são herdados e dependentes do domínio da aplicação, desse modo a participação do analista de dados torna-se essencial. Um exemplo simples de limpeza de dados seria a definição de um intervalo de possíveis valores para um determinado atributo, {0..10}. Caso surgisse qualquer valor diferente dos definidos no intervalo, esse dado seria retirado.
- **Codificação dos dados:** um algoritmo de codificação divide os valores contínuos dos atributos (inteiros ou reais) em uma lista de intervalos representados por um código. Ele efetivamente converte valores quantitativos em valores categóricos. Ou seja, cada intervalo resulta num valor discreto do atributo. Por exemplo, uma codificação para o atributo IDADE pode ser: {0..18} $\rightarrow$ Faixa 1; {19..25} $\rightarrow$ Faixa 2; {26..40} $\rightarrow$ Faixa 3 e assim por diante. Nesse exemplo, os valores contínuos das idades foram discretizados em três faixas. Em alguns casos a transformação de um valor em seu equivalente na base binária pode facilitar o algoritmo minerador a encontrar seu objetivo com melhor qualidade de resultados. Em resumo, essa fase converte os dados para a forma mais adequada para a construção e interpretação do modelo. A codificação de dados é potencialmente a tarefa onde há a necessidade de grande habilidade no processo de DCBD. Tipicamente essa fase exige a experiência do analista de dados e do seu conhecimento nos dados em questão. Embora o processo de DCBD possa ser executado sem essa fase, nota-se que, quando efetivada, os resultados obtidos são mais intuitivos e valiosos, além de que, na maioria das vezes, facilita a construção do modelo.

A fase de pré-processamento consiste na aplicação de um conjunto de processos que melhoram o desempenho dos esquemas de *data mining* usados para a descoberta de padrões. Estes processos constituem uma espécie de engenharia dos dados, ao reconstruir os dados de entrada de forma a adequarem-se ao esquema de aprendizagem escolhido.

4.1.1.3 Transformação

Essa fase consiste em agregar aos dados existentes mais informações de modo que essas contribuam no processo de descoberta de conhecimento. Essas informações serão incorporadas ao processo como um meta conhecimento do analista de dados, ou seja, informações que não estão na base de dados, porém são conhecidas e ratificadas.

4.1.1.4 Mineração de dados

Mineração de dados ou *Data mining* preocupa-se com a extração de padrões dos dados. Um padrão pode ser definido como uma representação de um determinado conjunto de fatos (dados) F, numa linguagem L, e com alguma medida de certeza C. Um padrão é uma declaração S em L que descreve um subconjunto F com uma certeza C tal que S é mais simples em algum sentido que a enumeração de todos os fatos em F.

A mineração de dados consiste na aplicação de técnicas de estatística e inteligência artificial em grandes bases de dados, para encontrar tendência ou padrões a fim de apoiar decisões. Todavia, a mineração de dados é apenas uma etapa do processo maior denominado DCBD. Mineração de dados é um campo multidisciplinar, que emergiu da interseção entre várias áreas, principalmente aprendizado de máquina (uma subárea da inteligência artificial), estatística e banco de dados.

Em (FAYYAD et al, 1996; CARVALHO, 2001) a definição de mineração de dados é apresentada como o uso de técnicas automáticas de exploração de grandes quantidades de dados de forma a descobrir novos padrões e relações que, devido ao grande volume de dados, não são facilmente descobertos pelo ser humano.

O termo mineração de dados, assim como arqueologia de dados, escavação de dados, é um dos termos utilizados na obtenção de conhecimento. Mineração de dados é um processo não trivial de identificar padrões válidos, novos, potencialmente úteis e compreensíveis em dados (SRIKANT; AGRAWAL, 1996b).

Mineração de dados é a etapa mais importante do processo de DCBD (FAYYAD et al, 1996). Caracteriza-se pela existência do algoritmo que diante da tarefa especificada será capaz de extrair eficientemente conhecimento implícito e útil de um banco de dados (LOPES, 1999). Pode-se dizer que mineração de dados é a fase que transforma dados em conhecimento, conseqüentemente um sistema de suporte a decisão baseado em algoritmos de mineração de dados é o que transforma os dados em conhecimento.

Na fase de mineração de dados é necessário que seja definida a técnica e o algoritmo a ser utilizado em função da tarefa proposta. A Tabela 4.1 mostra as principais tarefas de DCBD e as técnicas mais utilizadas para mineração de dados. As principais tarefas de mineração de dados são descritas na seção 4.1.2.

Tabela 4.1: Tarefas de DCBD e suas técnicas de mineração de dados

Tarefas de DCBD	**Técnicas**
Classificação	Algoritmos Genéticos, Redes Neurais e Árvores de Decisão
Agrupamento *(Clustering)*	Redes Neurais e Estatística
Previsão de Séries Temporais	Redes Neurais, Lógica Nebulosa e Estatística

Como pode ser visualizado na Tabela 4.1, há várias tarefas de mineração de dados, sendo que cada tarefa pode ser considerada como um tipo de problema, no qual se busca por um determinado tipo de conhecimento.

Uma tarefa consiste em agrupar objetos, criando grupos (o que é tecnicamente chamado de *clustering*), de modo que objetos dentro do mesmo grupo sejam o mais semelhante possível, enquanto que objetos de grupos diferentes sejam o mais diferente possível. Estes objetos são chamados de atributos previsores.

O fundamental é ter atributos previsores que sejam relevantes para fazer a previsão. Na verdade, esse é provavelmente o maior desafio enfrentado por usuários de *data mining*. Há vários métodos de mineração de dados disponíveis - tais como árvores de decisão, indução de regras, redes neurais, etc. - mas nenhum deles vai funcionar bem se

os dados contiverem muitos erros ou se não houver correlações importantes entre os atributos. Portanto, para que uma aplicação de *data mining* seja bem sucedida, além do método de mineração de dados propriamente dito, deve-se usar métodos de pré-processamento de dados e de interpretação e avaliação dos padrões descobertos.

4.1.1.5 Interpretação e avaliação de padrões

Esta fase tem por objetivo interpretar os padrões descobertos, validando-os ou não, selecionar as descobertas interessantes, utilizar o conhecimento descoberto, gerar relatórios, etc. Essa fase envolve a interpretação do conhecimento descoberto, ou algum processamento desse conhecimento. Esse pós-processamento deve ser incluído no algoritmo minerador, porém algumas vezes é vantajoso implementá-lo separadamente. Em geral, a principal meta dessa fase é melhorar a compreensão do conhecimento descoberto pelo algoritmo minerador, validando-o através de medidas da qualidade da solução e da percepção de um analista de dados. Esses conhecimentos serão consolidados em forma de relatórios demonstrativos com a documentação e explicação das informações relevantes ocorridas em cada etapa do processo de DCBD.

Uma maneira genérica de obter a compreensão e interpretação dos resultados é utilizar técnicas de visualização. Existem também outros tipos de técnicas de pós-processamento criados especialmente para um dado tipo de algoritmo minerador, ou para uma dada tarefa de DCBD. Por exemplo, converter os pesos das conexões de uma rede neural artificial num conjunto de regras.

4.1.2 Tarefas em DCBD

A seguir serão abordadas as principais tarefas DCBD, sendo elas: classificação, agrupamento e padrões de seqüências (LOPES, 1999). Algoritmos de classificação ou geração de perfis encontram perfis de diferentes grupos. Algoritmos de agrupamento segmentam o banco de dados em subconjuntos ou grupos. Algoritmos de padrões seqüenciais identificam tipos de padrões seqüenciais em restrições mínimas especificadas pelo usuário.

No desenvolvimento da ferramenta SEA proposta neste trabalho foi implementada a tarefa de agrupamento hierárquico por permitir uma interação melhor com o usuário especialista, uma vez que não seria necessário a especificação do número de grupos antecipadamente.

4.1.2.1 Classificação ou perfil

Classificação consiste em examinar os atributos de uma determinada entidade e baseado nestes atributos, atribuir esta entidade a uma determinada classe ou categoria.

Técnicas de classificação (HAN et al, 1993; ROMÃO; FRETIAS; PACHECO, 2000) permitem o desenvolvimento de perfis de itens com atributos em comum que podem ser usados para classificar novos itens adicionados ao banco de dados. A classificação é feita a partir de um conhecimento apriorístico sobre as classes e categorias utilizadas.

Para um dado conjunto de registros com seus atributos correspondentes, um conjunto de rótulos (representando registros) e um rótulo para cada registro, uma função de classificação examina o conjunto de registros rotulados e produz descrições das características dos registros para cada classe.

Por exemplo, em uma empresa os registros dos funcionários possuem atributos que descrevem, entre outros, o tipo de freqüência e a utilização de vale transporte. Então, para um funcionário que falta pouco, o registro é rotulado como "assíduo". Os registros

podem ser rotulados com "normal", "faltador" ou "inadimplente". A regra de classificação pode ser:

Funcionários que utilizam vale transporte têm uma taxa de faltas mensais menor que 2%.

Esta regra pode ser utilizada para a classificação de novos conjuntos de dados. Outro exemplo é o marketing direcionado. Qualquer companhia que pretenda realizar uma divulgação através de mala direta utilizará a sua lista de correspondência ou uma adquirida de terceiros. A lista de correspondência pode ter tido respostas em malas diretas anteriores e, através de um gerador de perfis, um modelo de classificação ou perfil é desenvolvido caracterizando as pessoas que tenham respondido às correspondências anteriores. O perfil então é tomado como uma previsão de resposta da correspondência atual. A lista de correspondência é então filtrada de maneira que os materiais promocionais sejam direcionados àqueles que correspondem ao perfil.

4.1.2.2 Agrupamento (Clustering)

A técnica de agrupamento é uma tarefa descritiva que procura identificar grupos homogêneos de objetos baseado nos valores de seus atributos (AGRAWAL et al, 1998). Esta técnica segmenta um conjunto de dados em subconjuntos ou grupos (*clusters*). O objetivo principal de *clustering* é separar objetos ou observações em classes naturais de forma que os elementos pertencentes a um mesmo grupo tenham um alto grau de semelhança ou similaridade, enquanto que, quaisquer elementos pertencentes a grupos distintos, tenham pouca semelhança entre si (Andeberg apud NEVES, 2001 p. 7)

Os critérios mais comuns adotados em *clustering* são: homogeneidade e heterogeneidade ou dissimilaridade. A homogeneidade refere-se a objetos pertencentes a um mesmo *cluster*, que devem ser tão similares quanto possível; Enquanto que a heterogeneidade, está relacionada a objetos de diferentes *clusters*, que devem ser distintos entre si, tanto quanto possível (Maravalle apud NEVES, 2001 p. 9)

Na análise de agrupamentos (*clustering analysis*), é feita a reunião de itens ou usuários com características semelhantes somente com base empírica, sem qualquer conhecimento prévio de quais serão os grupos formados. Neste caso, busca-se construir modelos que encontrem itens de dados similares entre si, a partir de alguma métrica de "distância" entre os itens, que serão assim colocados juntos em novos agrupamentos (*clusters*) ou em agrupamentos já existentes. Para esta finalidade, podem ser utilizados métodos geométricos, estatísticos e redes neurais.

A qualidade do resultado obtido pela utilização de técnicas de *clustering* depende de uma série de definições coerentes por parte do usuário. Alguns elementos importantes presentes no desenvolvimento dos procedimentos de *clustering* são: escolha dos atributos; homogeneização das variáveis; medidas de dissimilaridade; critérios de agrupamento; escolha do algoritmo; e definição do número de *clusters*.

Métodos de *clustering* normalmente utilizam uma medida de dissimilaridade para avaliar o grau de semelhança entre dois objetos durante o processo de agrupamento. Muitas vezes, esta medida é apresentada como sendo a distância entre dois objetos. A combinação entre a escolha das variáveis, transformações das variáveis (homogeneização) e as medidas de dissimilaridade escolhidas, é que traduz operacionalmente o termo "associação natural" entre objetos.

As métricas comumente utilizadas são: a Distância euclidiana (Equação 4.1) e a Distância de Manhattan (*city block*) (Equação 4.2), dadas pelas seguintes equações:

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - x_{jk} \right)^2}$$

Equação 4.1: Distância euclidiana

$$d_{ij} = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|$$

Equação 4.2: Distância de Manhattan

Onde:

- d_{ij} é a dissimilaridade entre os objetos *i* e *j*;
- x_{ik} é o valor é do atributo *k* para o objeto *i*, e
- x_{jk} é o valor é do atributo *k* para o objeto *j*.

Os algoritmos de *clustering* podem ser classificados em duas categorias principais: métodos hierárquicos e métodos de particionamento (não hierárquico) ou relocação iterativa (CARVALHO, 2001; MENDONÇA NETO, 2001).

O que define os processos hierárquicos 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 (gráfico em forma de árvore) (CARVALHO, 2001). A Figura 4.2 mostra um exemplo de dendograma, onde cada linha representa um grupo criado em algum momento do processo. Métodos aglomerativos formam grupos da direita para a esquerda, e métodos divisíveis se movem da esquerda para a direita do dendograma.

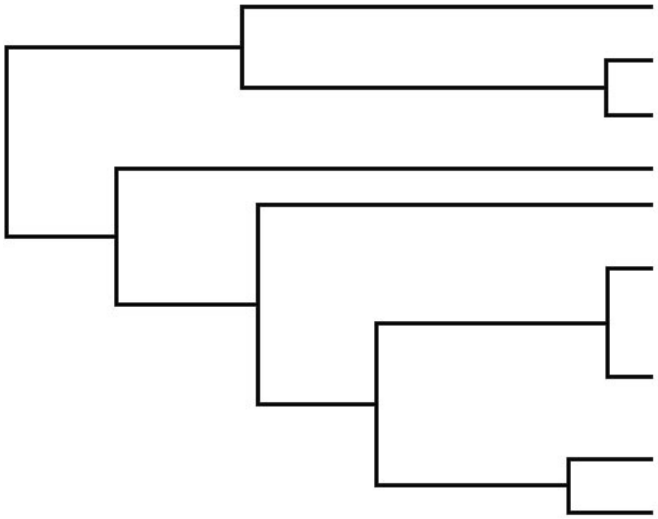

Figura 4.2: Exemplo de dendograma (CARVALHO, 2001)

4.1.2.2.1 Métodos hierárquicos

Os métodos hierárquicos podem ser aglomerativos ou divisivos (NEVES, 2001).Nos métodos hierárquicos aglomerativos, inicialmente, cada objeto é um *cluster* e, a cada passo do procedimento, os dois *clusters* mais próximos (similares) são fundidos, até que, ao final, exista somente um grande *cluster*, contendo todos os objetos. Este algoritmo é chamado de hierárquico por permitir obter vários níveis de agrupamento.

O que caracteriza os algoritmos de produzir agrupamento é o critério usado para definir a distância entre grupos. A seguir, são apresentadas técnicas aglomerativas para descoberta de agrupamentos (CARVALHO, 2001):

- **Método de centróide**: é o mais direto deles, pois substitui 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 intergrupos é definida pela distância entre os seus centros. Em cada etapa procura-se fundir grupos que tenham a menor distância entre si;
- **Método de ligação simples (vizinho mais próximo)**: define-se como semelhança entre dois grupos aquela com pelos dois membros mais parecidos. Ele acha os dois indivíduos (objetos) separados pela menor distância e os coloca no primeiro grupo. Então, a próxima menor distância é encontrada e um terceiro indivíduo se junta aos dois primeiros para formar um grupo. O processo continua até que todos os indivíduos estejam em um grupo. A distância entre quaisquer dois grupos é a menor distância de qualquer ponto em um grupo para qualquer ponto no outro;
- **Método de ligação completa (vizinho mais longe)**: é parecido com o anterior, exceto que o critério de similaridade é baseado na distância máxima. A distância máxima entre indivíduos em cada grupo é representada pela menor esfera (diâmetro mínimo) que pode incluir todos os objetos. Este método é chamado de ligação completa porque todos os objetos no grupo são ligados a alguma distância máxima ou similaridade mínima.

Os passos do algoritmo hierárquico aglomerativo são:

- **Passo 1**: Iniciar com *n* clusters, cada um contendo um objeto.
- **Passo 2**: Calcular as dissimilaridades entre os objetos.
- **Passo 3**: Procurar o par de clusters com menor dissimilaridade;
- **Passo 4**: Recalcular a dissimilaridade do *cluster* fundido com os demais clusters.
- **Passo 5**: Repete os passos 3 e 4, 1 – *n* vezes.

A Figura 4.3 mostra um exemplo de como a distância mais curta (ligação simples) e a mais longa (ligação completa) representam similaridades entre os pontos. Ambas as medidas refletem apenas um aspecto dos dados. O uso de ligação simples reflete apenas um único par de objetos (os mais próximos), enquanto a ligação completa reflete novamente um único par, porém os dois mais externos. Isto é útil para visualizar medidas, refletindo a similaridade do par mais similar ou menos similar de objetos.

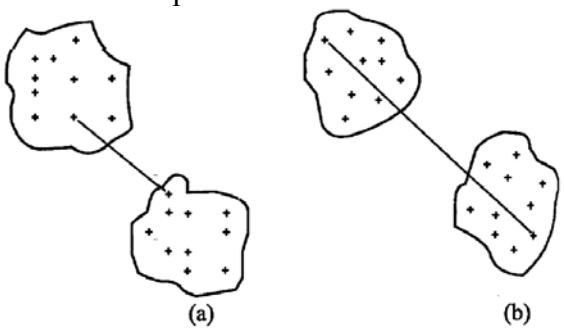

Figura 4.3: (a) Ligação simples e (b) Ligação completa

A Figura 4.4 mostra um exemplo do funcionamento de um algoritmo hierárquico com a hierarquia de módulos correspondente. Os pontos pretos são os componentes, as

linhas definem os módulos, as linhas tracejadas os módulos que serão criados nos passos futuros.

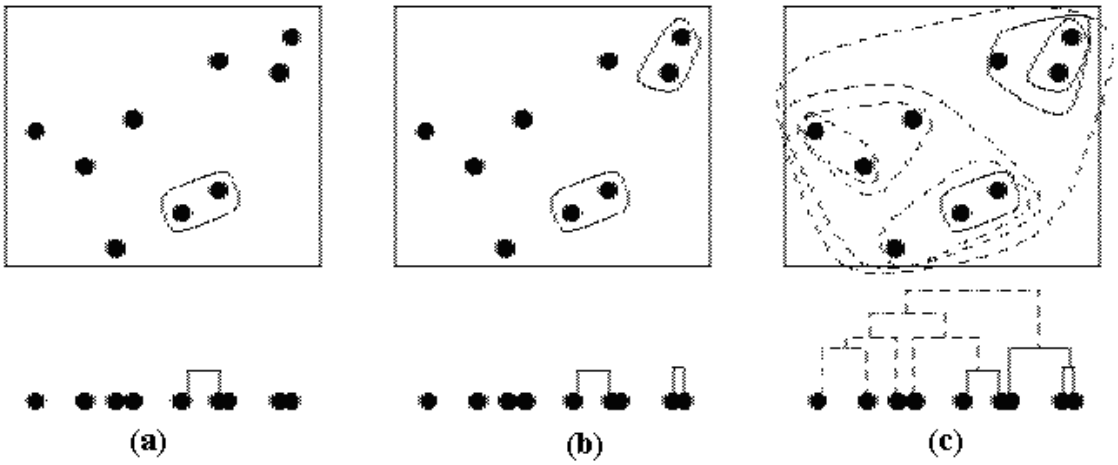

Figura 4.4: Exemplo do funcionamento de um algoritmo hierárquico

4.1.2.2.2 Métodos não hierárquicos

Os métodos de particionamento buscam encontrar, iterativamente, a melhor partição dos *n* objetos em *k* grupos. Freqüentemente os *k* clusters encontrados pelos métodos de particionamento são de melhor qualidade (grupos internamente mais homogêneos) do que os *k* clusters produzidos pelos métodos hierárquicos. Os métodos de particionamento mais utilizados são baseados em um ponto central (*k*-médias) ou em um objeto representativo para o *cluster* (*k*-medoids).

A diferença básica do *k*-medoids em relação ao *k*-médias está na utilização de um objeto representativo, chamado de *medoid*, localizado mais ao centro possível do *cluster*, ao invés de um centro médio. Ele realiza em cada passo uma busca exaustiva pela troca de um dos *k*-medoids previamente selecionados por um dos demais (*n*-*k*) objetos, que minimize as dissimilaridades entre os *k* medoids e os membros dos *k* clusters.

Existem muitos algoritmos de agrupamento (*clustering*) diferentes, com parâmetros a definir dependentes dos resultados que se quer obter e dos fatos disponíveis. Alguns constroem uma partição do conjunto de componentes (os módulos são mutuamente exclusivos).

Nos métodos de partição procura-se diretamente uma partição dos *n* objetos, de modo que satisfaçam as duas premissas básicas de coesão interna de isolamento dos grupos. O uso dos métodos de partição pressupõe também o conhecimento do número *k* de partições desejadas.

Ao contrário do método hierárquico, os procedimentos não hierárquicos não envolvem construção de um grafo em árvore. 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 seis grupos não é apenas a combinação de dois grupos da solução anterior, mas é baseado em apenas encontrar a melhor solução com seis grupos. De forma geral, o processo funciona da seguinte maneira: o primeiro passo é selecionar um registro (uma semente) como o centro inicial do grupo, e todos os objetos (indivíduos) dentro de uma distância pré-especificada são incluídos no grupo resultante. O critério mais usado é o da soma de quadrados residual, inspirado em análise de variância. Então, outra semente de grupo é escolhida e as atribuições continuam até que todos os objetos

estejam atribuídos a algum grupo. Existem várias abordagens diferentes para selecionar sementes e atribuir objetos. A seguir são descritas três abordagens típicas elaboração dos grupos (CARVALHO, 2001):

- **Seqüencial**: seleciona uma semente de grupo e inclui todos os objetos dentro de uma distância da semente mais próxima. Conforme o processo evolui, as distâncias podem ser ajustadas para incluir menos ou mais objetos no grupo;
- **Paralelo**: seleciona várias sementes de grupo simultaneamente no começo e atribui objetos dentro da distância da semente mais próxima. Conforme o processo evolui, as distâncias podem ser ajustadas para incluir menos ou mais objetos nos grupos;
- **Otimizado**: similar aos outros dois, exceto que ele permite a relocação de objetos. O centro do agrupamento é a média das variáveis de todos os registros pertencentes ao grupo. Conseqüentemente, o centro do agrupamento se modifica a cada vez que um novo registro é adicionado. Se no decorrer da atribuição de objetos, um objeto fica mais próximo de outro grupo que não é o grupo a que ele foi atribuído originalmente, então ele é trocado para o grupo mais similar.

Os passos básicos de um algoritmo baseado em *k*-médias, são (MENDONÇA NETO, 2001):

- **Passo 1**: Escolha de *k* objetos para serem centros iniciais dos *k clusters*.
- **Passo 2**: Cada objeto é associado a um *cluster*, para o qual a dissimilaridade entre objeto e o centro deste *cluster* é a menor que as demais.
- **Passo 3**: Os centros dos clusters são recalculados, redefinindo cada um, em função dos atributos de todos os objetos pertencentes ao *cluster*;
- **Passo 4**: Volta ao passo 2, até que os centros dos *clusters* se estabilizem.

A cada iteração, os objetos são agrupados em função do centro do *cluster* mais próximo e, por conseqüência, os centros dos *clusters* são reavaliados (passo 3). Isto provoca no espaço de atributos um deslocamento dos centros médios. O algoritmo é interrompido quando as médias não são mais deslocadas, ou há uma insignificante relocação de objetos entre os *clusters*.

A Figura 4.5 mostra o algoritmo graficamente. *K* é igual a 3 neste caso. Os três aglomerados, nomeados *A*, *B*, e *C*, são mostrados na Figura 4.5. A parte (i) mostra os passos 2 e 3 do algoritmo. No passo 2, três registros são escolhidos como centróides dos aglomerados *A*, *B*, e *C*. No passo 3, os outros registros são assinalados ao aglomerado que tiver o centróide mais próximo. A parte (ii) mostra os novos centróides para os aglomerados *A*, *B*, e *C*. Eles são recalculados a partir das médias de todos os registros assinalados a eles na parte (i).

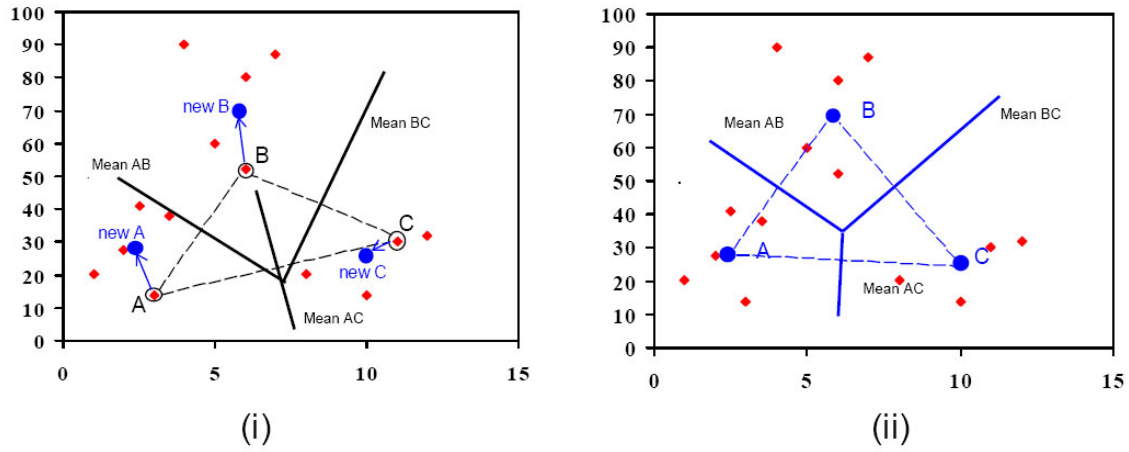

Figura 4.5: Método *k*-médias (MENDONÇA NETO, 2001)

4.1.2.3 Padrões Seqüenciais

Esta técnica procura eventos que ocorreram em seqüência no tempo (SRIKANT; AGRAWAL, 1996a). Por exemplo, uma empresa pode descobrir que 20% dos funcionários que recebem pagamento na sexta-feira faltam na segunda-feira.

Agrawal em (AGRAWAL, SRIKANT, 1995), define padrões seqüenciais da seguinte forma: dada uma base de dados D de transações, o problema de mineração de padrões seqüenciais é encontrar o máximo de seqüências em todas as seqüências, que tenham um determinado suporte mínimo especificado por um usuário.

4.2 Redes neurais como técnicas de mineração de dados

As RNA (Redes Neurais Artificiais) procuram aprender padrões diretamente dos dados através de um processo de repetidas apresentações dos dados à rede, ou seja por experiência. Dessa forma, uma RNA procura por relacionamentos, constrói modelos automaticamente, e os corrige de modo a diminuir seu próprio erro. Estas redes utilizam um conjunto de elementos de processamento (ou nós) análogos aos neurônios no cérebro. Estes elementos de processamento são interconectados em uma rede que pode identificar padrões nos dados, uma vez expostos aos mesmos, ou seja, a rede aprende através da experiência, tal como as pessoas. Esta característica distingue redes neurais de tradicionais programas computacionais, que simplesmente seguem instruções em uma ordem seqüencial fixa.

Para o desenvolvimento da sistemática apresentada neste trabalho optou-se pela utilização de redes neurais auto-organizáveis pelo fato de estas não utilizarem algoritmos de aprendizagem não-supervisionados. A rede neural SOM foi escolhida como primeira implementação devido a outras ferramentas já a utilizarem. Esta escolha não teve o objetivo de comparação de performance, mas de implementação e validação da rede neural escolhida.

4.2.1 Processos de Aprendizado

A propriedade mais importante das RNA é a habilidade de aprender de seu ambiente e com isso melhorar seu desempenho. Isso é feito através de um processo iterativo de ajustes aplicado a seus pesos, o treinamento. O aprendizado ocorre quando a rede neural atinge uma solução generalizada para uma classe de problemas.

O objetivo do treinamento de uma RNA é fazer com que a aplicação de um conjunto de entradas produza um conjunto de saídas desejado ou no mínimo um conjunto de saídas consistente. Cada conjunto de entrada ou saída é chamado de vetor. O treinamento é realizado pela aplicação seqüencial dos vetores de entradas e em alguns casos também os de saída, enquanto os pesos da rede são ajustados de acordo com um procedimento de treinamento pré-determinado. Durante o treinamento, os pesos da rede gradualmente convergem para determinados valores, tal que a aplicação dos vetores de entrada produza as saídas necessárias.

Os procedimentos de treinamento que levam as RNA a aprender determinadas tarefas podem ser classificados em duas classes de treinamento (HAYKIN, 2001):

- **Supervisionado**: quando é utilizado um agente externo que indica à rede a resposta desejada para o padrão de entrada. Este tipo de treinamento necessita de um par de vetores composto do vetor de entrada e do vetor alvo que se deseja como saída. Juntos estes vetores são chamados de par de treinamento ou vetor de treinamento, sendo interessante ressaltar que geralmente a rede é treinada com vários vetores de treinamento. O procedimento de treinamento funciona da seguinte forma: o vetor de entrada é aplicado. A saída da rede é calculada e comparada com o correspondente vetor alvo. O erro encontrado é então realimentado através da rede e os pesos são atualizados de acordo com um algoritmo determinado a fim de minimizar este erro. Este processo de treinamento é repetido até que o erro para os vetores de treinamento tenha alcançado níveis aceitáveis;
- **Não supervisionado (auto-organização):** quando não existe um agente externo indicando a resposta desejada para os padrões de entrada. Este tipo de treinamento não requer vetor alvo para as saídas e não faz comparações para determinar a resposta ideal. O conjunto de treinamento modifica os pesos da rede de forma a produzir saídas que sejam consistentes, isto é, tanto a apresentação de um dos vetores de treinamento, como a apresentação de um vetor que é suficientemente similar, irá produzir o mesmo padrão nas saídas. O processo de treinamento extrai as propriedades estatísticas do conjunto de treinamento e agrupa os vetores similares em classes. A aplicação de um vetor de uma determinada classe à entrada da rede irá produzir um vetor de saída específico, mas não existe maneira de se determinar, antes do treinamento, qual o padrão que será produzido na saída para um vetor de entrada de uma determinada classe. Desta forma, a saída de algumas RNA deve ser transformada em uma forma compreensiva após o processo de treinamento, o que é um simples problema de identificação das relações entrada-saída estabelecidas pela rede.

Denomina-se época uma apresentação de todos os *N* pares (entrada e saída) do conjunto de treinamento no processo de aprendizado. A correção dos pesos em uma época pode ser executada de dois modos:

- **Modo padrão**: A correção dos pesos acontece a cada apresentação à rede de um exemplo do conjunto de treinamento, esta apresentação se denomina iteração. Cada correção de pesos baseia-se somente no erro do exemplo apresentado naquela iteração. Assim, em cada época ocorrem N correções;
- **Modo em lote**: Apenas uma correção é feita por época. Todos os exemplos do conjunto de treinamento são apresentados à rede, seu erro médio é calculado e a partir deste erro, fazem-se as correções dos pesos.

A Tabela 4.2 (MURAD; LOMBARDI, 2003) apresenta de maneira sumária vários algoritmos de aprendizagem e suas arquiteturas de rede associadas. Tanto paradigmas de aprendizagem supervisionada quanto não-supervisionada empregam regras de aprendizagem baseadas em correção do erro, Hebb e aprendizagem competitiva. Regras de aprendizagem baseadas em correção do erro podem ser usadas para treinar redes em sem realimentação, enquanto que regras de aprendizagem hebbianas têm sido usadas para todos os tipos de arquiteturas de rede. Entretanto, cada algoritmo de aprendizagem é projetado para treinar uma arquitetura específica. Desta forma, quando se discute um algoritmo de aprendizagem, tem-se associada uma arquitetura particular de rede. Cada algoritmo pode realizar adequadamente somente algumas tarefas. A última coluna da tabela lista as tarefas que cada algoritmo realiza adequadamente.

Tabela 4.2: Algoritmos de aprendizagem e suas arquiteturas de rede associadas

Paradigma	Regra de aprendizagem	Arquitetura	Algoritmo de aprendizagem	Tarefa
Supervisionado	Correção de erros	Perceptron com uma camada	Algoritmos de aprendizagem do perceptron, Adaline	Classificação de padrões
		Perceptron com várias camadas	Backpropagation; Madaline	Aproximação de funções, predição e controle
	Hebb	Multicamadas sem realimentação	Análise discriminante linear	Análise de dados; classificação de padrões
	Competitiva	Competitiva	Quantização vetorial adaptativa	Categorização em classes internas; compressão de dados
		ARTMAP	ARTMAP	Classificação de padrões; categorização em classes internas
Não Supervisionado	Hebb	Sem realimentação ou competitiva	Análise da componente principal	Análise de dados; compressão de dados
		Rede Hopfield	Aprendizagem de memória associativa	Memória associativa
	Competitiva	Competitiva	Quantização de vetores	Categorização; compressão de dados
		SOM de Kohonen	SOM de Kohonen	Categorização; análise de dados
		Rede ART	ART1, ART2, …	Categorização
Híbrido	Correção de erros e competitiva	Rede RBF	Algoritmo de aprendizagem RBF	Classificação de padrões; aproximação de funções; predição; controle

4.2.2 Redes neurais auto-organizáveis

As RNA auto-organizáveis possuem a capacidade de auto-organização de seus pesos. Existem vários modelos de RNA auto-organizáveis, cada um com suas peculiaridades e propósitos. Os algoritmos de treinamento utilizados podem ser divididos em dois grandes grupos:

- Aprendizado competitivo;
- Aprendizado hebbiano.

Quando o aprendizado competitivo é utilizado, os nodos competem entre si pelo direito de atualizar seus pesos. Estes algoritmos de aprendizado têm sido mais utilizados em problemas de classificação, de extração de características, compressão de dados e formação de clusters (agrupamentos). Exemplos de redes que utilizam aprendizado competitivo são os modelos ART (CARPENTER; GROSSBERG, ROSEN, 1991; CARPENTER; GROSSBERG, 1988; CARPENTER; GROSSBERG, 1990) e SOM (KOHONEN, 1989).

O aprendizado hebbiano, por outro lado, é baseado no princípio para atualização dos pesos proposto pelo neurofisiologista Donald Hebb em *The Organization of Behaviour* (HEBB, 1949). As principais áreas onde este algoritmo tem sido utilizado são extração de características, análise de dados e memória associativa. Exemplos de Redes que utilizam esta forma de aprendizado são os modelos de Hopfield (HOPFIELD, 1982), Linsker (LINSKER, 1986) e redes PCA (*Principal Component Analysis*) (BALDI; HORNIK, 1989)**.**

De acordo com Von der Malsburg (MALSBURG, 1973), os algoritmos auto-organizáveis acompanham os seguintes princípios:

- Modificações dos pesos associados às conexões tendem a aumentar estes pesos.
- Limitações de recursos levam à competição entre as sinapses e à seleção das mais aptas em detrimento das outras.
- Diferentes modificações nos pesos tendem a cooperar entre si.

4.2.2.1 Rede de SOM de Kohonen

Os mapas auto-organizáveis de Kohonen fazem parte de um grupo de redes neurais chamado redes baseadas em modelos de competição, ou simplesmente redes competitivas (FAUSETT apud SILVA, 1998). Estas redes combinam competição com uma forma de aprendizagem para fazer os ajustes de seus pesos. Kohonen, em sua obra "*Self-Organization and Associative Memory*" de 1989, sugeriu um modelo de topologia de rede e um sistema de aprendizagem que dispensasse o supervisor durante a fase de treinamento, chamado SOM (*Self Organizing Map*) (HAYKIN, 2001). De fato, este trabalho apresenta uma compreensão inicial de como o cérebro humano consegue aprender determinadas tarefas, sobretudo na fase infantil, sem requerer um "tutor" para o aprendizado. Outra característica importante deste tipo de rede é que elas utilizam treinamento não supervisionado, onde a rede busca encontrar similaridades baseando-se apenas nos padrões de entrada. O principal objetivo dos mapas auto-organizáveis de Kohonen é agrupar os dados de entrada que são semelhantes entre si formando classes ou agrupamentos denominados clusters (SILVA, 1998).

A rede de Kohonen é uma estrutura de duas camadas de neurônios. A primeira camada é a de entrada e seus neurônios estão completamente interconectados aos neurônios da segunda camada – denominada competitiva - que é organizada numa grade bi-dimensional ou em um arranjo dependente do objeto a ser mapeado (ALMEIDA, 1995). Quando padrões de aprendizado são apresentados à rede, os pesos das unidades são adaptados de maneira a preservar a topologia do objeto.

A rede de Kohonen (Figura 4.6) é composta por um número de entradas correspondente ao tamanho dos padrões e um conjunto de neurônios de saída, sendo que

cada padrão a ser reconhecido deverá ter, no mínimo, um neurônio de saída correspondente. Isto é, se for necessária à distinção entre 10 padrões, a quantidade mínima na camada de saída são 10 neurônios.

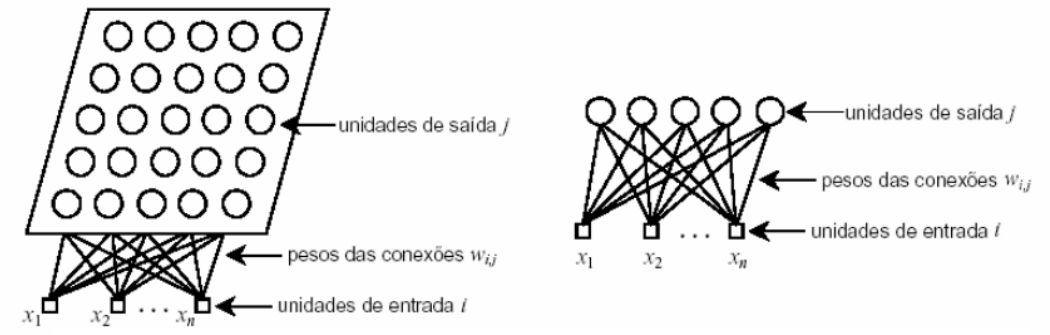

Figura 4.6: Arquitetura típica

O sistema proposto por Kohonen é formado por uma camada de elementos neurais que estão fortemente conectados entre si. Estas conexões obedecem a um arranjo, em que a saída de cada elemento é apresentada à entrada de todos os seus vizinhos e vice-versa. Deste modo, o nível de excitação de um elemento da rede é conhecido e influencia os outros. Além disso, cada elemento possui uma entrada e uma saída externa, de forma que cada elemento possa receber e reagir às excitações colocadas na entrada da rede. Inicialmente, todos os pesos de todos elementos da rede são ajustados aleatoriamente, de modo que quaisquer elementos da rede têm igual probabilidade de apresentar uma saída alta, para um padrão apresentado na entrada da rede. A seguir, o método de treinamento consiste em aplicar uma regra de ajuste de pesos que é comumente chamada de *the winner takes all*, ou seja, o vencedor leva tudo. Esse nome deve-se ao fato de que a essência do método é premiar o elemento da rede que tiver a maior excitação em resposta ao padrão de entrada.

Em função da topologia de conexões, os elementos vizinhos também serão influenciados pela excitação do elemento "vencedor" e apresentarão níveis também altos de excitação. O algoritmo de treinamento ajustará os pesos somente daqueles elementos que apresentarem um determinado limiar de excitação. Além disso, somente esses elementos apresentarão uma saída externa tendo os outros elementos suas saídas zeradas. Como resultado, a rede apresentará uma função de excitação de saída dos seus elementos que variará com a distância tomada a partir do elemento vencedor.

O algoritmo básico de treinamento do SOM consiste de três fases. Na primeira fase, competitiva, os neurônios da camada de saída competem entre si, segundo algum critério, geralmente a distância euclidiana, para encontrar um único vencedor, também chamado de BMU (*Best Match Unit*). Na segunda fase, cooperativa, é definida a vizinhança deste neurônio. Na última fase, adaptativa, os vetores de código do neurônio vencedor e de sua vizinhança são ajustados.

A relação de vizinhança entre os neurônios é estabelecida segundo alguma função. O principal objetivo da função de vizinhança é controlar o nível de atuação dos neurônios em torno do neurônio vencedor do processo competitivo. Seguindo o modelo neurobiológico tem-se que o nível de atuação dos neurônios vizinhos decai à medida que o mesmo se distancia do BMU.

Seja $h_{j,i}$ a vizinhança topológica centrada no neurônio i e com um conjunto de neurônios cooperativos J, $j \in J$. Seja $d_{i,j}$ a distância lateral entre o neurônio vencedor i e o neurônio j. Para que $h_{j,i}$ atenda aos requisitos neurobiológicos, a mesma tem que ser simétrica em relação ao ponto de valor máximo ($d_{i,j} = 0$), e $h_{j,i}$ deve decair

monotonicamente com o aumento da distância lateral ($d_{i,j}$), decaindo para próximo de 0 quando $d_{i,j} \rightarrow \infty$.

A função gaussiana $h_{j,i} = exp(-d^2_{i,j}/2\delta^2)$ satisfaz estas exigências e é invariante à translação. δ representa o raio da vizinhança topológica e o grau que os neurônios vizinhos do BMU participam do processo de aprendizagem adaptativa. A rede SOM converge mais rapidamente com este tipo de função de vizinhança e definindo δ como uma função monotonicamente decrescente em função do tempo (épocas), $\delta(t) = \delta(0)exp(-t/\tau^j)$, sendo τ^j uma constante.

Seja Ξ o conjunto dos padrões de entrada composto por x_k, $k = 1, ..., n$, tem-se o algoritmo de aprendizagem padrão ou seqüencial, como segue (SILVA, 2004):

a) Os vetores de código, $w_j = [w_{j1}, ..., w_{jp}]^T$, são iniciados aleatóriamente
b) Para cada época t

   1) Para todo $x_k \in \Xi$, $k = 1, ..., n$, para o tempo discreto t, encontre o neurônio vencedor c segundo a distância euclidiana:

      $$c = \arg\min_j \{\|x_k - w_j\|\},\ j = 1,\ j = 1,..., m$$

      onde m corresponde ao número de neurônios na rede. A ordem de apresentação dos padrões deve ser aleatória.

   2) Os vetores de código w_j do neurônio vencedor e dos seus vizinhos são, então, atualizados segundo a equação:

      $$w_{ji}(t+1) = w_{ji}(t) + \alpha(t)h(t)[x_{ik}(t) - w_{ij}(t)]$$

      onde $\alpha(t)$ é uma função que determina a taxa de aprendizagem na iteração t e $h(t)$ é a função que determina a vizinhança entre o neurônio vencedor c e seus vizinhos.

Uma vez escolhido o neurônio, seu vetor de pesos e os vetores de pesos dos seus vizinhos mais próximos são atualizados. Existem várias formas de se escolher a vizinhança sendo que as duas apresentadas na Figura 4.7 são as mais utilizadas (VARGAS, 2004). No início do processo de treinamento a vizinhança de neurônios atualizados é relativamente grande, porém ao longo do processo de treinamento, a vizinhança é reduzida de forma gradual (LO; FUJITA; BAVARIAN, 1991; RITTER; MARTINETZ; SCHULTEN, 1992). Uma técnica que tem se mostrado efetiva para acelerar o processo de convergência de treinamento de um SOM é reduzir a vizinhança de acordo com uma função gaussiana (LO; FUJITA; BAVARIAN, 1991; ERWIN; OBERMAYER; SCHULTEN, 1992).

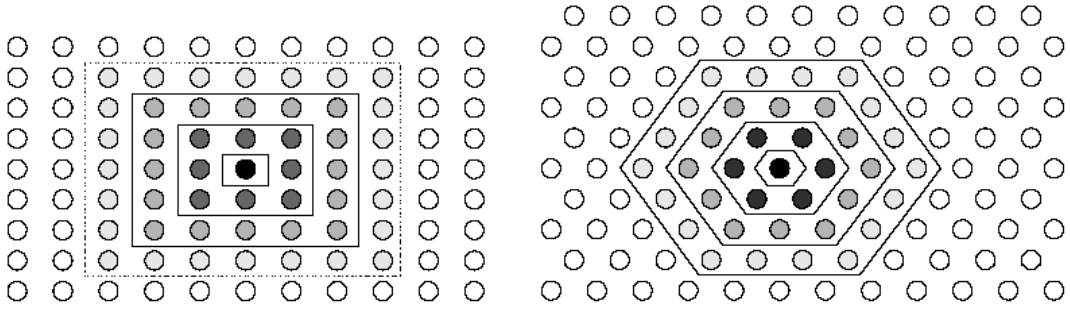

Figura 4.7: Configurações de vizinhança do SOM

O processo de treinamento de um SOM cria, de forma natural, agrupamentos ou clusters. Cada *cluster* corresponde a um grupo de padrões que compartilham características similares, sendo que o centro corresponde ao padrão que melhor representa aquela classe. Na Figura 4.8 pode-se observar a representação bidimensional de um *cluster* junto com três das imagens que foram classificadas, esta figura mostra a correspondência entre a distancia física dos neurônios de uma rede de Kohonen e o grau de semelhança dos padrões por eles reconhecidos. Após ter realizado o processo de treinamento de um SOM, unidades próximas fisicamente na rede representam padrões similares (KOHONEN, 1990).

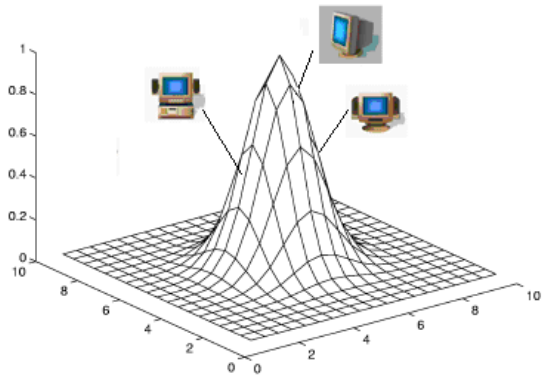

Figura 4.8: Relação entre distância física e semelhança no SOM

O treinamento da rede SOM (BRAGA; LUDERMIR, 2000) ocorre em duas fases: fase de ordenação e fase de convergência. Durante a fase de ordenação, ocorre a ordenação topológica dos vetores de pesos, que são inicialmente orientados de forma aleatória (com pesos iniciais aleatórios). Esta fase dura em torno de 1000 ciclos ou iterações. Nesta fase, o treinamento busca agrupar os nodos do mapa topológico em *clusters* ou agrupamentos, de modo a refletir a distribuição dos padrões de entrada. Desta forma, a rede descobre quantos clusters ela deve identificar e suas posições relativas no mapa. O mapeamento realizado neste estágio é um mapeamento grosseiro dos padrões de entrada. Durante essa fase, a taxa de aprendizado é inicialmente alta, próxima de 1, sendo gradualmente reduzida até um valor próximo de 0,1. Assim, nesta fase ocorrem grandes mudanças nos pesos. Também durante esta fase, a vizinhança é reduzida. Inicialmente, até atingir um raio de um ou dois vizinhos.

A segunda fase faz um ajuste mais fino do mapa. Durante essa fase, que requer 100 a 1000 vezes mais ciclos que a fase anterior, é utilizada uma taxa de aprendizado baixa, da ordem de 0,001 ou menos, e o raio da vizinhança envolve um ou nenhum vizinho. Esta fase aperfeiçoa o mapeamento realizado no estágio anterior, fazendo um ajuste fino dos pesos dos agrupamentos.

O SOM de Kohonen pode ser usado para projeção de dados multivariados, aproximação de densidade e agrupamento. Ele tem sido utilizado com sucesso em áreas de reconhecimento de voz, processamento de imagens, robótica e controle de processos. Os parâmetros de projeto incluem a dimensionalidade da lista de neurônios, o número de neurônios em cada dimensão, a forma da vizinhança, o escalonamento de encolhimento da vizinhança e a taxa de aprendizagem.

O aprendizado de Kohonen difere significativamente das regras de Hebb e de Widrow, por ser baseado em um princípio de treinamento auto-organizado. A idéia básica é existir uma camada de elementos processadores, que organize seus vetores de pesos de tal forma que estes sejam distribuídos com um número de densidade

aproximadamente proporcional à função densidade de probabilidade, de acordo com os vetores de entrada usados para treinar as camadas selecionadas.

A rede de Kohonen tem aplicação em reconhecimento de padrões.

4.2.3 Algoritmos de aprendizado

Denomina-se algoritmo de aprendizado um conjunto de regras bem definidas para solução de um problema de aprendizado. Existem muitos tipos de algoritmos de aprendizado específicos para determinados modelos de redes neurais. Esses algoritmos diferem entre si principalmente pelo modo como os pesos são modificados (HAYKIN, 2001). A rede neural se baseia nos dados para extrair um modelo geral. Portanto, a fase de aprendizado deve ser rigorosa e verdadeira, a fim de se evitar modelos espúrios. Todo o conhecimento da rede neural está armazenado nas sinapses, ou seja, nos pesos atribuídos às conexões entre os neurônios. Cerca de 50 a 90% aleatoriamente do total de dados deve ser separado para o treinamento da rede neural. Estes dados são escolhidos aleatoriamente, a fim de que a rede "aprenda" as regras e não "decore" os exemplos.

4.2.3.1 Regras de Hebb

A regra de aprendizado de Hebb (Figura 4.9) propõe que o peso de uma conexão sináptica deve ser ajustado se houver sincronismo entre os níveis de atividade das entradas e saídas. Se dois neurônios, em lados distintos da sinapse, são ativados sincronamente, teremos um fortalecimento da sinapse. Entretanto, se os neurônios forem ativados assincronamente, a sinapse será enfraquecida ou mesmo eliminada. Em outras palavras, se o neurônio pré-sináptico tiver grande influência na ativação do neurônio pós-sináptico, a conexão entre eles deve ser reforçada.

A idéia básica é que se duas unidades são ativadas simultaneamente, suas interconexões tendem a se fortalecer. Se *i* recebe o sinal de entrada de *j*, o peso W_{ij} é modificado de acordo com (ALMEIDA, 1995):

$$\Delta W_{ij} = \lambda a_i a_j$$

Equação 4.3: Modificação dos pesos Hebb

onde λ é uma constante de proporcionalidade representando a taxa de aprendizado e a_i e a_j são ativações das unidades *i* e *j* respectivamente. Alguns autores representam alternativamente a matriz W_{ij} por W_{ji}.

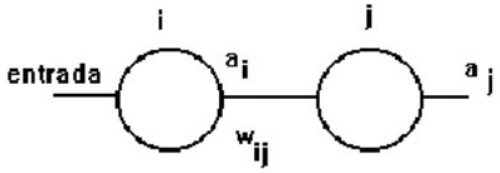

Figura 4.9: Representação da Regra de Hebb

Um único neurônio treinado usando a regra de Hebb apresenta uma seletividade de orientação. A Figura 4.10 demonstra esta propriedade. Os pontos indicados são desenhados a partir de uma distribuição gaussiana bidimensional e são usados para treinar um neurônio. O vetor de pesos dos neurônios é inicializado em w_0 como indicado. Conforme o treinamento prossegue, o vetor de pesos move-se progressivamente para mais próximo da direção *w* de máxima variância nos dados. De

fato, *w* é o autovetor da matriz de covariância dos dados correspondendo ao maior autovalor.

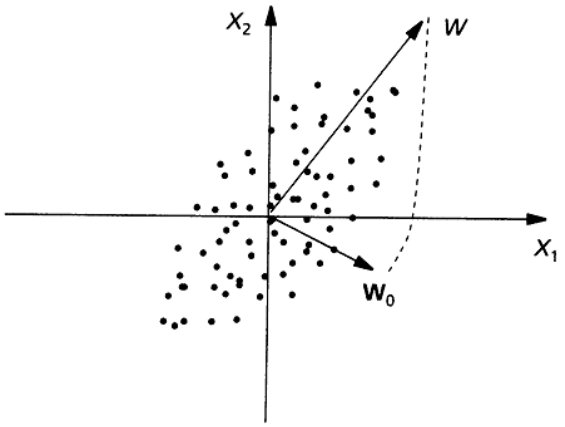

Figura 4.10: Neurônio treinado usando a regra de Hebb

4.2.3.2 Regra Delta ou Regra de Widrow-Holff

O treinamento supervisionado do modelo de rede Perceptron consiste em ajustar os pesos e bias ou limiar de suas unidades para que a classificação desejada seja obtida. Para adaptação do bias juntamente com os pesos pode-se considerá-lo como sendo o peso associado a uma conexão cuja entrada é sempre igual a +1 e adaptar o peso relativo a essa entrada. Quando um padrão é inicialmente apresentado a rede, ele produz uma saída. Após medir a diferença entre a resposta atual e a desejada, são realizados os ajustes apropriados nos pesos das conexões de modo a reduzir este erro. Esse procedimento é conhecido como regra de delta (HAYKIN, 2001). É uma variante da Regra de Hebb, introduzida por Widrow-Hoff; a diferença quanto a de Hebb é que possui uma saída desejada d_j , assim o peso será proporcional ao erro de saída. Sendo a_j e a_i os níveis de ativações das unidades *j* e *i* respectivamente.

$$\Delta W_{ij} \propto (d_j - a_j)a_i$$

Equação 4.4: Modificação dos pesos Widrow-Holff

A Regra Delta depende da função de ativação dos neurônios e minimiza o erro entre a saída desejada e o valor de ativação do neurônio (ALMEIDA, 1995). É aplicável para o treinamento de redes neurais com aprendizado supervisionado.

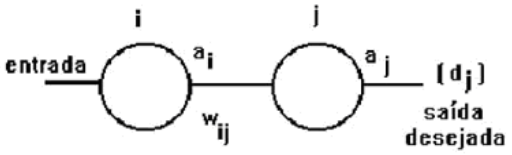

Figura 4.11: Regra de Delta

A Regra Delta é baseada na descida do gradiente da curva de erro. Este erro é obtido pela diferença entre o valor da resposta correta desejada e o valor obtido na saída do elemento. O erro na saída de um elemento é determinado em função dos pesos usados

em correspondência a cada entrada, ou seja, de acordo com a alteração dos pesos pode-se subir ou descer a curva de erro resultante na saída do neurônio.

4.2.3.3 Aprendizado Competitivo

No aprendizado competitivo a arquitetura da rede pode possuir duas camadas: a camada de entrada que é constituída de unidades de processamento que recebem os padrões de entrada e a segunda camada que é a camada competitiva. As duas camadas são completamente interconectadas e cada conexão tem um peso associado. Na camada competitiva, as unidades competem pela oportunidade de responder ao padrão de entrada. A unidade vencedora neste caso representa a categoria de classificação do padrão de entrada.

A rede mais simples com aprendizagem competitiva consiste de uma única camada de unidades de saída. Cada unidade de saída i na saída da rede conecta-se a todas as unidades de entrada (x_j) através de pesos w_{ij}, $j = 1, 2, \ldots, n$. Cada unidade de saída também se conecta a todas as outras unidades de saída através de pesos inibitórios mas tem uma auto-realimentação com um peso excitatório. Como resultado da competição, somente a unidade i^* com a maior (ou menor) entrada total torna-se a vencedora, isto é, $w_i^* \cdot x \geq w_i \cdot x$, $\forall i$, ou $\left\| w_i^* - x \right\| \leq \left\| w_i - x \right\|$, $\forall i$. Quando todos os vetores de pesos estão normalizados estas duas desigualdades são equivalentes.

Uma regra simples de aprendizagem competitiva pode ser colocada como

$$\Delta w_{ij} = \begin{cases} \eta\left(x_j^{\,u} - w_{i^*j}\right), & i = i^*, \\ 0, & i \neq i^*. \end{cases}$$

Equação 4.5: Regra de aprendizagem competitiva

Observe-se que somente os pesos da unidade vencedora são atualizados. O efeito desta regra de aprendizagem é o de mover o padrão armazenado na unidade vencedora (pesos) para um pouco mais próximo do padrão de entrada. A Figura 4.12 apresenta uma interpretação geométrica da aprendizagem competitiva (HAYKIN, 2001). Neste exemplo assume-se que todos os vetores de entrada tenham sido normalizados ao comprimento unitário. Eles estão apresentados como pontos pretos na Figura 4.12. Os vetores de peso das três unidades são inicializadas aleatoriamente. Suas posições inicial e final na esfera após a aprendizagem competitiva estão marcados como X's na Figura 4.12(a) e Figura 4.12(b) respectivamente. Na Figura 4.12 cada um dos três grupos naturais de padrões ("*clusters*") forem encontrados por uma unidade de saída cujo vetor de pesos aponta para o centro de gravidade do grupo encontrado.

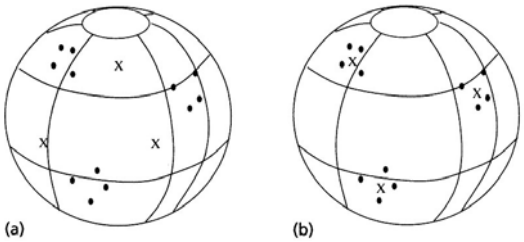

Figura 4.12: Interpretação geométrica da aprendizagem competitiva

A Figura 4.12 ilustra a habilidade de uma rede neural de realizar agrupamento (clustering) através da aprendizagem competitiva.

5 A FERRAMENTA SEA – SEMANTIC EQUIVALENCE ATTRIBUTE

A ferramenta SEA objetiva encontrar equivalência semântica entre atributos, este processo será denominado Descoberta de Equivalência Semântica entre Atributos (DESA). Partindo deste objetivo foi elaborada a metodologia para que a ferramenta pudesse ser implementada.

Os algoritmos de programação tradicionais são eficientes quando os termos e regras que sustentam estes algoritmos são definidos e conhecidos. Este fato não acontece na comparação semântica entre atributos de bancos de dados. Isto se deve ao fato de esta relação seguir uma lógica que normalmente é difusa.

As redes neurais têm sido utilizadas em sistemas cujo propósito é a identificação de padrões, antes desconhecidos. Estas podem aprender similaridades entre os dados, diretamente de suas instâncias, sem conhecimento a priori. Diferentemente das soluções tradicionais, as redes neurais são treinadas e não programadas. A saída natural de uma rede neural é o rótulo da classificação ou do agrupamento (*cluster*).

O objetivo principal da tarefa de identificação de agrupamentos (*clustering*) é separar objetos ou observações em classes naturais de forma que os elementos pertencentes a um mesmo grupo tenham um alto grau de semelhança ou similaridade, enquanto que, quaisquer elementos pertencentes a grupos distintos, tenham pouca semelhança entre si (ENGEL, 2001).

5.1 Modelo Conceitual - Framework

Segundo Haykin (2001), um mapa contextual, também chamado de mapa semântico, assemelha-se aos mapas corticais que são mapas computacionais formados no córtex cerebral. Os mapas contextuais encontram aplicações em campos diversos como a classificação não-supervionada de classes fonéticas a partir de textos, sensoriamento remoto e exploração de dados ou mineração de dados.

O modelo conceitual aqui proposto baseia-se parcialmente nos conceitos descritos por Haykin (2001) com relação aos mapas semânticos. Considere um conjunto de discriminadores de atributos (x_a) de bancos de dados representados por valores entre zero e um. Para cada atributo existe um vetor x_a de discriminadores. Considere um vetor coluna x_s cujo k-ésimo elemento representando o atributo $k=1,2, \ldots , n$, recebe um valor fixo $a;$ os outros elementos são igualados a zero. O vetor x_s representa a influência do código simbólico em relação ao vetor x_a. O parâmetro a representa a influência relativa do código simbólico (no vetor x_s) comparado ao código de discriminadores (vetor x_s). O valor de a deve ser escolhido de tal forma que não seja determinante sobre o vetor de discriminadores. Então o vetor de entrada x para a rede neural é um vetor composto da concatenação entre os vetores x_a e x_s.

Uma vez que os valores para os discriminadores dos atributos pertencem ao intervalo [0,1], é difícil atribuir um valor para a, pois qualquer valor poderia ser determinante no processo de DESA. Daí o fato de este trabalho estar parcialmente baseado nos mapas semânticos como descrito.

A ferramenta SEA baseia-se nas fases de DCDB. A DESA consiste em agrupar atributos semanticamente equivalentes em grupos (*clusters*) onde o usuário possa visualizar os atributos, assim como o agrupamento entre os mesmos, e permitir que este usuário possa interagir com a ferramenta para refinar o processo de descoberta. A DESA pode ser considerada uma especialização do processo de DCBD.

O modelo conceitual da ferramenta pode ser observado na Figura 5.1. Ele está dividido em quatro fases:

- A seleção, onde os discriminadores são extraídos das bases de dados;
- A transformação, onde os discriminadores passam por um processo de normalização para que os valores sejam convertidos para valores entre 0 e 1;
- A modelagem, onde os valores obtidos na fase de preparação servirão de entrada para a rede neural do tipo SOM (*Self-Organizing Map*) (KOHONEN, 1989);
- E análise, onde a rede neural treinada é utilizada para agrupar o conjunto de discriminadores (metadados), ou seja, é feita a aplicação do modelo obtido. Nesta fase um especialista observa o resultado do agrupamento, assim como faz uso de outras ferramentas, como o dendograma e a U-matriz (ULTSCH, 1993), para auxiliá-lo.

Após o processo de aprendizagem da rede, espera-se que os neurônios que compõem o mapa de neurônios do SOM agrupem os vetores de entrada (discriminadores) de cada atributo, de acordo com seu grau de similaridade.

O modelo conceitual da ferramenta admite a possibilidade de utilização de outros tipos de redes neurais. A SEA foi concebida segundo o modelo conceitual exposto na Figura 5.1 e para que outras redes neurais possam ser, futuramente, utilizadas. A rede SOM é tratada como uma classe com funcionamento independente.

Na ferramenta SEA as fases descritas na Figura 4.1 são implementadas pelos módulos representado através dos ícones descritos na Tabela 5.1.

Tabela 5.1: Representação das fases de DESA na ferramenta SEA

Fase	Ícone
Seleção	Opção Add e Romove (menu)
Transformação	
Modelagem	
Aplicação do modelo / análise	 

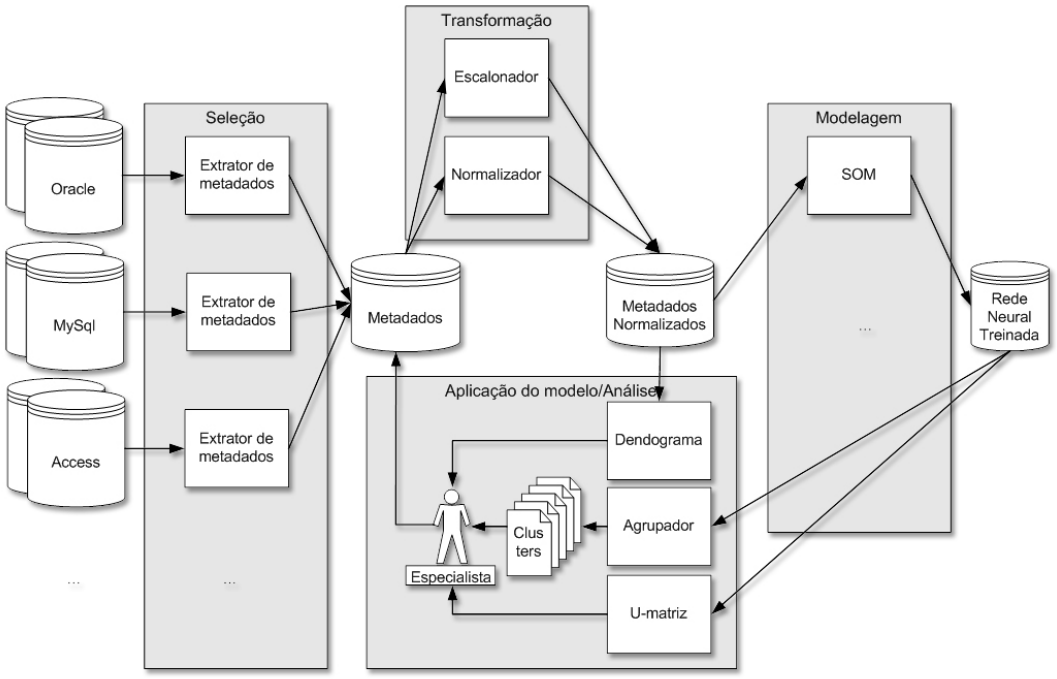

Figura 5.1: Modelo conceitual da ferramenta SEA

5.2 Fases para descoberta semântica de atributos da SEA

O processo de utilização da ferramenta está descrito na Figura 5.1. Este foi dividido em quatro fases: a seleção, a transformação, a modelagem e aplicação do modelo e análise.

5.2.1 Seleção

Esta fase está dividida em três tarefas distintas. A configuração do acesso às fontes de dados, a configuração da ferramenta com relação à extração dos metadados e a extração dos metadados.

5.2.1.1 Acesso às fontes de dados

Inicialmente o usuário deve escolher uma ou mais fontes de dados como entrada para a ferramenta. A SEA permite estabelecer conexão com banco de dados Oracle, MySQL, Access e arquivo texto. As fontes de dados devem ser cadastradas pelo usuário para que possam ser estabelecidas conexões com as mesmas. Neste cadastro é necessário informar um nome para a fonte de dados, o *driver* de conexão com o banco de dados, o nome do usuário do banco de dados e a senha do mesmo.

O modelo prevê que estas fontes de dados são bancos de dados relacionais com suporte a conexão via ODBC (*Open Database Connectivity*). Porém, assim como a rede neural, o extrator que é o módulo que fará contato com a fonte de dados, está também isolado dos demais módulos, em termos de independência de funcionamento, e, sendo assim, outros módulos podem ser implementados para que haja acesso a outros bancos de dados, relacionais ou não, e também a outras fontes de dados onde a descoberta de equivalência semântica entre atributos seja importante.

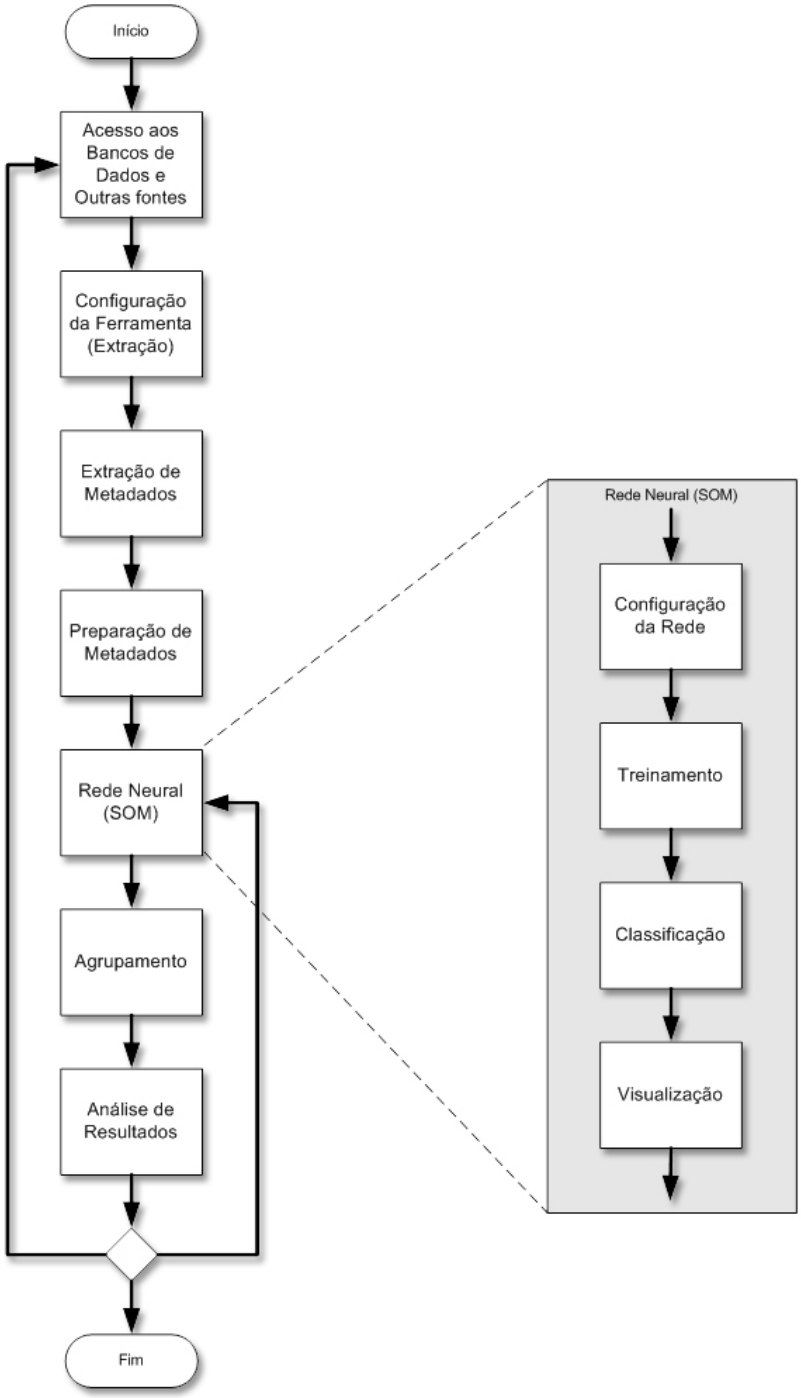

Figura 5.2: Processo de utilização da ferramenta SEA

5.2.1.2 Configuração da ferramenta

A ferramenta deve permitir ao usuário a possibilidade de interação com o processo de extração e preparação dos metadados. Sendo assim, o mesmo, pode decidir entre escalonar ou não os metadados, e se optar em escalonar, este escalonamento pode ser de forma linear ou não linear, sendo a segunda forma chamada de sigmoide. Ele pode também optar por adicionar o vetor x_s descrito anteriormente ao vetor de discriminadores (metadados). Esta alternativa foi prevista para que se pudessem observar os resultados obtidos com a metodologia, porém os testes comprovaram que é

quase impossível o vetor x_s não influenciar erroneamente nos resultados. Outra possibilidade prevista para interação com o usuário é a possibilidade de normalizar os discriminadores, e a normalização pode ser realizada sobre todos os discriminadores ou sobre o vetor x_a.

A influência da mudança destes parâmetros será descrita com mais detalhes na descrição da fase de preparação dos metadados.

5.2.1.3 Extração de metadados

Segundo Li e Clifiton (2000), existem três níveis de metadados que podem ser automaticamente extraídos de um banco de dados: nomes de atributos (nível de dicionário), informações sobre o esquema (nível de campo) e conteúdo dos dados e estatísticas (nível de dados). O conjunto destes metadados forma o domínio de conhecimento sobre o banco de dados.

Os metadados ou discriminadores de cada SGBD podem ser diferentes, contudo eles são fixos para o mesmo tipo de SGBD. Neste sentido, deve ser criado um extrator para cada tipo de SGBD com os qual se deseja trabalhar.

A SEA trabalha com um conjunto de quinze discriminadores para cada atributo, que podem ser observados na Tabela 5.2. O conjunto destes discriminadores forma o vetor de entrada de cada atributo. O vetor de entrada x pode ser composto dos vetores x_a e x_s ou apenas do vetor x_a, conforme a parametrização do usuário. Eles são metadados dos atributos.

Tabela 5.2: Discriminadores

Discriminador	**Função**
nulable	Representa o fato de atributo poder ter (1) ou não (0) nulos
size	Tamanho que o tipo do Atributo pode armazenar
name	Nome do Atributo
count	Quantidade de registros na tabela, com instâncias (valores) para este atributo
average	Média calculada sobre as instâncias (valores) do atributo em questão
sum	Soma das instâncias dos atributos
stddev	Desvio padrão (raiz quadrada da variância)
variance	Variância
min	Valor mínimo das instâncias do atributo
max	Valor máximo das instâncias do atributo
bin	1, se o atributo em questão for do tipo binário e 0, caso contrário
text	1, se o atributo em questão for do tipo texto e 0, caso contrário
char	1, se o atributo em questão for do tipo caractere e 0, caso contrário
date	1, se o atributo em questão for do tipo data e 0, caso contrário
number	1, se o atributo em questão for do tipo numérico e 0, caso contrário

Os bancos de dados, comerciais ou não, não possuem apenas os tipos básicos de dados mencionados na Tabela 5.2. Por exemplo, o banco de dados Oracle em sua versão 8i possui tipos como *long*, *long raw*, *blob*, *clob* entre outros. Cabe ao módulo extrator da ferramenta mapear o tipo nativo do sistema de entrada, que pode ser um banco de dados ou outro, para um dos tipos básicos que a ferramenta SEA compreende.

Os discriminadores *bin*, *text*, *char*, *date* e *number*, recebem o valor zero ou um. Este valor é atribuído conforme o tipo básico do atributo, encontrado pelo módulo extrator de acordo com o tipo nativo no banco de dados de origem. Seria possível haver um único discriminador chamado *type* e este receberia um valor conforme uma tabela onde, por exemplo, o tipo básico *bin* valeria 0,10, *text* teria o valor 0,20, e assim por diante. Porém, optou-se por criar novos discriminadores e assim o valor numérico atribuído a cada tipo não necessita ser diferente, portanto não influenciando no comportamento da rede neural.

Quando o tipo do atributo for numérico, os discriminadores *count*, *average*, *sum*, *stddev* e *variance* são calculados sobre os valores de suas instâncias, conforme as equações na Tabela 5.3. Todas as instâncias do atributo são levadas em consideração. Se o atributo possuir instâncias com valor nulo, estas são desconsideradas para o cálculo, ou seja, se um atributo possui 1000 instâncias, das quais 70 são nulas, *n* tem o valor de 930.

Tabela 5.3: Equações usadas para discriminadores

Discriminador	Equação
count	n
average	$\frac{\sum_{i=1}^{n} X_i}{n}$
Sum	$\sum_{i=1}^{n} X_i$
Variance	$S = \frac{\sum_{i=1}^{n} X_i^2 - \frac{1}{n}\left[\sum_{i=1}^{n} X_i\right]^2}{n-1}$
Stddev	$S^2 = \left[S\right]^2$

Para atributos com tipo básico igual a texto e data, os discriminadores *count*, *average*, *sum*, *stddev* e *variance* são calculados sobre a quantidade de caracteres dos valores de suas instâncias, sem espaços em branco no início e no final de cada valor. A equação utilizada para o cálculo segue a Tabela 5.3.

Para o cálculo dos discriminadores *max* e *min* são utilizados os mesmos procedimentos dos atributos com tipo básico igual a numérico, porém os atributos com tipo básico igual data retornam valores com o formato data.

O vetor de entrada de cada atributo, para a rede neural, deve ser um valor numérico. Nos bancos de dados podem ser encontrados discriminadores numéricos ou quantitativos e discriminadores categóricos ou qualitativos. Os quantitativos podem ser utilizados como entrada para as redes neurais, porém os categóricos devem passar por um processo de transformação. O discriminador *name* para todos os tipos de dados e os discriminadores *min* e *max* para o tipo básico data devem ser convertidos para valores numéricos. Esta conversão é realizada através da função listada na Figura 5.3. A função getString2Number recebe como parâmetro uma seqüência de caracteres e retorna um valor numérico com base na posição de cada caractere e seu valor na tabela *unicode*.

```
/**
 * @param s
 * @return
 */
private String getString2Number(String s)
{

    int x = 0;
    int i = 0;
    double n = 0.0;
    Character c;
    String ret = "0.0d";

    try
    {
        for (i = 0; i < s.length(); i++)
        {
            x = Character.getNumericValue(s.charAt(i));
            n = n + (i * x);
        }
        ret = new Double(n / i).toString();
    }
    catch (NullPointerException npe)
    {
        ret = "0.0d";
    }
    return ret;

}
```

Figura 5.3: Listagem da função getString2Number

5.2.2 Transformação

Os discriminadores podem ser apresentados sob diferentes granularidades ou unidades de medida, como por exemplo, litros, quilos, toneladas, unidades, moeda, etc. Devido a isso, é necessário submetê-los a tratamentos de normalização, que possam tornar possíveis a busca de possíveis relacionamentos entre eles, bem como a identificação de similaridade.

Os discriminadores que já estão entre 0 e 1 e cujos valores foram atribuídos pela ferramenta SEA, não são escalonados. Estes discriminadores são: *nulable, bin text, char, date e number*. Nos dados utilizados no capítulo 6 tem-se a bela 0. com os atributos e os metadados não escalonados. A Tabela 8.1 mostra os metadados transformados em discriminadores após o escalonamento conforme Equação 5.1.

A ferramenta prevê dois tipos de escalonamento. O primeiro é o escalonamento do tipo sigmoide, que tem este nome devido ao gráfico gerado pela função deste tipo (ver Figura 5.4 (d)). Como função de escalonamento sigmoide a ferramenta utiliza-se da Equação 5.1, onde x_{es} representa o discriminador escalonado pela função e x_i corresponde ao discriminador antes do escalonamento.

$$x_{es} = 2\left(\frac{1}{\left(1+1.01^{-x_i}\right)} - 0.5\right)$$

Equação 5.1: Escalonamento sigmoide

O segundo é o escalonamento linear, que tem este nome devido ao gráfico da função de escalonamento ser uma reta (ver Figura 5.4 (c)). Para escalonar, de modo linear, os discriminadores a ferramenta SEA utiliza a Equação 5.2, onde x_{el} representa o discriminador escalonado linearmente e x_{min} e x_{max} correspondem respectivamente aos valores mínimo e máximo do discriminador.

$$x_{el} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

Equação 5.2: Escalonamento linear

A utilização de uma função *sigmoid* é recomendável uma vez que ela minimiza os seguintes problemas de escalonamento:

Reconhecimento incorreto:

O vetor *x* (discriminadores) de um atributo *t* em uma base de dados *b* é representado por x_{tb} = {1, 2, 3, 4, 5, 6, 7, 8 , 9 , 10 , 11, 12, 13, 14, 15}. Na primeira posição tem-se o primeiro discriminador $x_{tb}[1] = 1$. O maior valor para *x*[1] em *b* é 10, para todos os *n* atributos. Um outro vetor *x* (discriminadores) de um atributo *z* em uma base de dados *c* é representado por x_{zc} = {100, 200, 300, ... , 1500}. Na primeira posição tem-se o primeiro discriminador $x_{zc}[1] = 100$. O maior valor para *x*[1] em *c* é 1000, para todos os *n* atributos. Linearmente, $x_{tb}[1]/\max(x[1]$ em $b) = x_{zc}[1]/\max(x[1]$ em $c)$, ou seja, 1/10 é igual a 100/1000, conseqüentemente o valor máximo para normalização dos valores dever ser único e o valor deverá ser o maior entre max(*x*[1] em *b*) e max(*x*[1] em c).

Diferenciação incorreta:

O vetor *x* (discriminadores) de um atributo *t* em uma base de dados *b* é representado por x_{tb} = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150}. Na primeira posição tem-se o primeiro discriminador $x_{tb}[1] = 10$. O maior valor para *x*[1] em *b* é 10, para todos os *n* atributos. Um outro vetor *x* (discriminadores) de um atributo *z* em uma base de dados *c* é representado por x_{zc} = {10, 20, 30, ... , 150}. Na primeira posição tem-se o primeiro discriminador $x_{zc}[1] = 10$. O maior valor para *x*[1] em *c* é 1000, para todos os *n* atributos. Linearmente, $x_{tb}[1]/\max(x[1]$ em $b) \neq x_{zc}[1]/\max(x[1]$ em $c)$, ou seja, 10/10 é diferente de 10/1000, conseqüentemente o valor máximo para normalização dos valores dever ser único e o valor deverá ser o maior entre max(*x*[1] em *b*) e max(*x*[1] em c).

Na Figura 5.4 é possível notar que para intervalos iguais e pequenos como [0,100] tanto a o escalonamento linear (a) quanto o escalonamento sigmoide (b) produzirão resultados parecidos. Porém quando o intervalo utilizado é maior, no escalonamento sigmoide os valores das faixas maiores são ajustados para os limites (d) enquanto que no escalonamento linear o comportamento permanece alterado, surgindo os problemas de reconhecimento incorreto e diferenciação incorreta.

O escalonamento trabalha sobre o vetor de discriminadores dos atributos x_a mencionado na seção 5.1. O vetor de símbolos x_s pode ser normalizado também. A normalização ou escalonamento de todos os discriminadores é realizada pela aplicação da Equação 5.3, onde *N* representa o discriminador normalizado e x_i corresponde ao discriminador. Para a normalização dos vetores x_a e x_s , *n* tem o valor igual à soma do número de elementos de x_a , que é quinze mais o número de elementos de x_s que varia de acordo com o número de atributos a serem comparados.

$$\overline{x}_i = \frac{x_i}{\sqrt{\sum_{i=1}^{n} x_i^2}}$$

Equação 5.3: Normalização

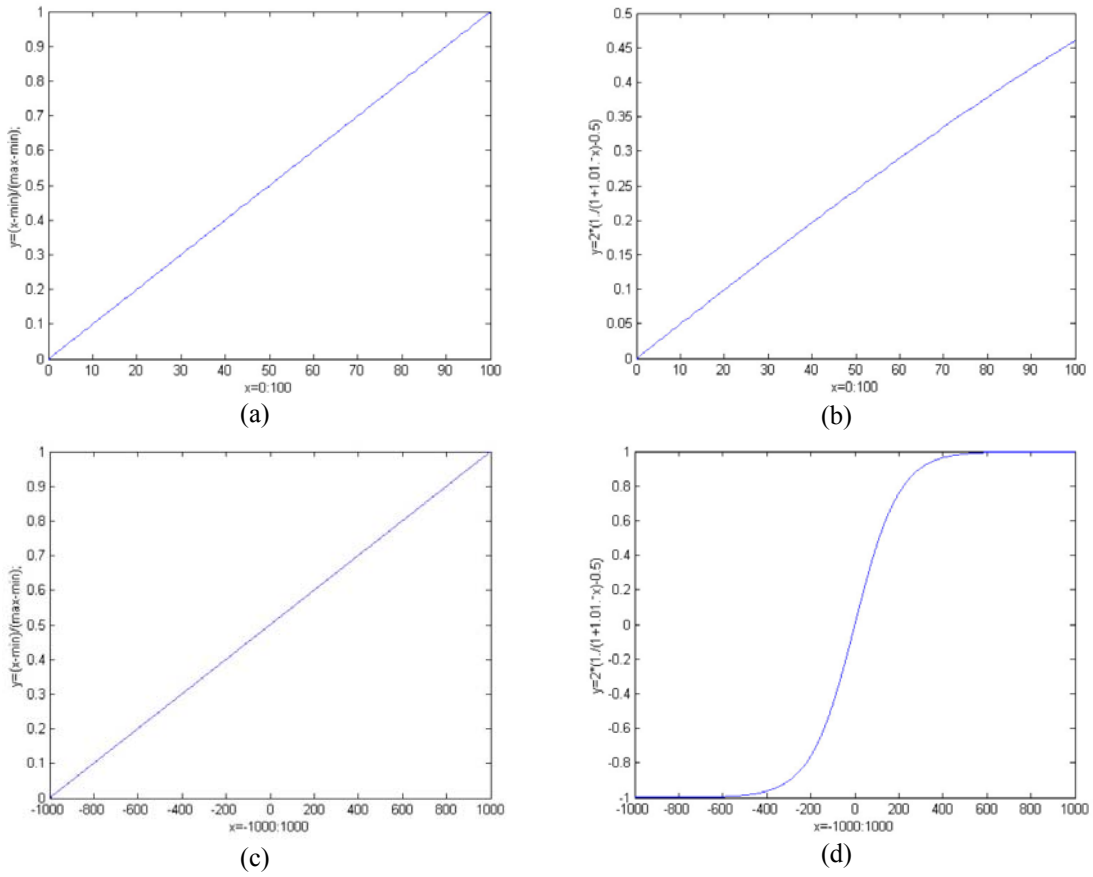

Figura 5.4: Escalonamento

5.2.3 Modelagem

Uma matriz de dimensões n x m, onde n representa o número de atributos e m o número de discriminadores, é passada para a rede neural. O módulo de rede neural também recebe outra matriz de dimensões n x 3, onde são passados os dados dos atributos, que são esquema, tabela e nome. Outros módulos de redes neurais podem ser implementados fazendo uso destes parâmetros.

O módulo de rede neural implementado é uma rede do tipo SOM, desenvolvida por Kohonen (1989). Este módulo monta um mapa de neurônios e com valores padrão, definidos pela ferramenta SEA. Em uma rede neural SOM o mapa de neurônios é bidimensional. A ferramenta define a dimensão inicial do mapa como sendo o número de atributos somado ao número de discriminadores, ou seja, para 16 atributos e 15 discriminadores a ferramenta montará um mapa inicial de 31 x 31 neurônios. Não há como predizer qual a melhor dimensão para o mapa de neurônio, o que existem são heurísticas para esta tarefa. O usuário tem como alterar a dimensão do mapa de neurônios, porém o módulo da rede SOM está preparado para trabalhar com dimensões iguais n x n.

Cada neurônio possui um vetor de pesos. No início do treinamento são escolhidos aleatoriamente valores entre 0 e 1 para cada valor do vetor de pesos dos neurônios do mapa. A dimensão do vetor de pesos é igual à dimensão do vetor de discriminadores do atributo. Uma representação gráfica destes valores pode ser visualizada na Figura 5.5, que é também o estado inicial da rede neural.

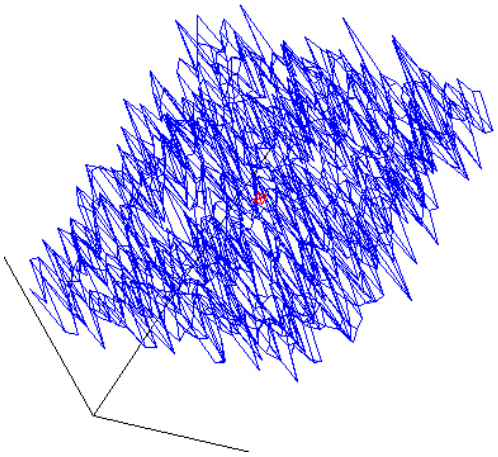

Figura 5.5: Mapa de ativações com pesos aleatórios

Após sortear os valores aleatórios para cada neurônio, é necessário estabelecer o número de iterações, também chamado de ciclos. O número de iterações é necessário para finalizar o treinamento, ele representa o número de vezes que vetores de discriminadores dos atributos serão submetidos à rede neural SOM. O número de iterações definido inicialmente é igual ao dobro do número de atributos. Os atributos são escolhidos aleatoriamente e o atributo que já foi submetido à rede não se repete até que uma nova época comece. Por exemplo, se houver 16 atributos com 32 iterações, cada atributo será sorteado duas vezes, e duas épocas ocorrerão. Época é uma apresentação do conjunto de vetores de todos os atributos à rede neural.

Durante as iterações ocorre o treinamento da rede. A cada iteração é sorteado um dos atributos do vetor de entrada e o vetor de descrição deste atributo é comparado com o vetor de pesos de cada um dos neurônios do mapa que forma a rede neural SOM. Esta comparação é feita verificando-se a distância euclidiana (Equação 5.4) entre os vetores. O neurônio que possuir a menor distância em relação ao vetor de discriminadores do atributo sorteado é eleito o neurônio vencedor, também denominado BMU (*Best Match Unit*).

$$d_E = \sqrt{\sum_{i=1}^{n} (x_i - w_i)^2}$$

Equação 5.4: Escolha do neurônio vencedor (Distância euclidiana)

Uma vez encontrado o neurônio vencedor ($w_{b(i),b(j)}$) em relação a este é definida uma região circular denominada vizinhança, que é dada através da Equação 5.5, onde i_b = índice de coluna do mapa de neurônios, do neurônio vencedor e j_b = índice de linha do mapa de neurônios, do neurônio vencedor.

$$\pi_{ic} = \exp\left(\frac{-\left((i - i_b)^2 + (j - j_b)^2 \right)}{2\sigma^2(n)} \right)$$

Equação 5.5: Função gaussiana de vizinhança

O raio de atuação (σ) desta vizinhança deve diminuir com o aumento do número de iterações, segundo Kohonen (1989). Isto permite que a rede possa convergir. A redução

de σ é dada por σ(n) conforme a Equação 5.6, onde, *r*max = raio máximo, *r*min = raio mínimo, *lc* = ciclos realizados, *nc* = número de ciclos. Os valores para raio máximo e mínimo são pré-fixados pela ferramenta, respectivamente, em *i*-1 e 1, onde *i* é o número de colunas no mapa de neurônio. Para i pode ser o número de linhas no mapa de neurônios, uma vez que o mapa é representado por um quadrado. O valor do raio máximo é igual ao número de discriminadores.

$$\sigma(n) = round\left(r\max.\exp\left(\frac{lc}{nc}.\log\left(\frac{r\min}{r\max}\right)\right)\right)$$

Equação 5.6: Função de largura de vizinhança

Como o módulo de rede neural SOM utiliza uma equação gaussiana para cálculo da região de vizinhança, a região obtida no mapa de neurônios é circular, como pode ser observado na Figura 5.6, onde o neurônio vencedor (BMU) está indicado ao centro da região de vizinhança e seus neurônios vizinhos estão destacados. É possível observar outras regiões que foram escolhidas em outras iterações, estas regiões sobrepõem-se dando à rede a característica de adaptabilidade aos padrões dos vetores de entrada de cada atributo.

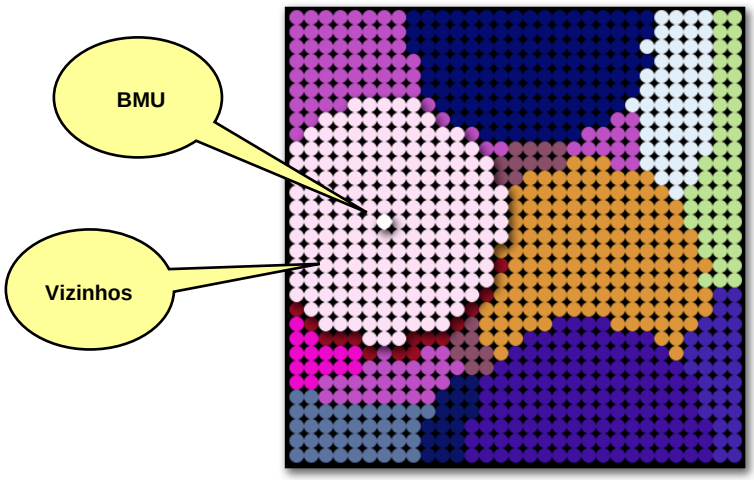

Figura 5.6: Neurônio vencedor

Os pesos dos neurônios vizinhos ao neurônio vencedor devem ser ajustados segundo a Equação 5.7.

$$w_i(n+1) = w_i(n) + \eta(n).\pi_{ic}(n).(x(n) - w_i(n))$$

Equação 5.7: Atualização de pesos (w)

A taxa de aprendizagem deve cair suavemente como mostra a Figura 5.7, partindo de um valor máximo até um valor mínimo pré-estabelecidos. Na ferramenta SEA estes valores são respectivamente pré-fixados em 0,9 e 0,1, porém o usuário pode alterar estes valores.

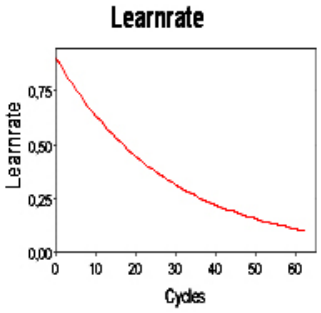

Figura 5.7: Taxa de aprendizagem

Para que a taxa de aprendizado diminua de modo suave foi utilizada uma equação exponencial mostrada na Equação 5.8, onde, *l*max = taxa de aprendizado máxima, *l*min = taxa de aprendizado mínima, *lc* = ciclos realizados e *nc* = número de ciclos.

$$\eta(n) = l\max.\exp\left(\frac{lc}{nc}.\log\left(\frac{l\min}{l\max}\right)\right)$$

Equação 5.8: Taxa de aprendizagem

A cada iteração ou ciclo, é calculado o erro absoluto médio. O valor do erro absoluto médio mostra a média das diferenças entre a distância euclidiana de cada um dos vetores de entrada e o vetor de pesos do neurônio que melhor combina (BMU) com o vetor de entrada. O gráfico do erro absoluto médio pode ser observado na Figura 5.8. O valor do EAm dever cair em função do número de iterações ou ciclos, porém é possível perceber que em determinados momentos o EAm cresce.

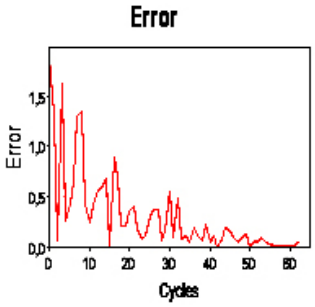

Figura 5.8: Erro absoluto médio

Para o cálculo do EAm foi utilizada a Equação 5.9, onde $w_{b(i)}$ representa o vetor de pesos do neurônio vencedor em relação ao vetor de pesos do atributo x_i.

$$EAm = \frac{1}{n}\sum_{i=1}^{n}\left|x_i - w_{b(i)}\right|$$

Equação 5.9: Erro absoluto médio

Após a ocorrência do número de iterações previstas espera-se que o mapa de neurônios agrupe as características de similaridade dos vetores de discriminadores dos atributos que serviram como entrada para a rede SOM.

Os vetores de discriminadores de cada atributo são novamente submetidos à rede neural, onde o BMU é encontrado e atribuído a este atributo. Esta etapa esta representada na Figura 5.2 pelo item classificação. Cada atributo é assinalado no mapa

de ativação da rede neural com uma marca vermelha, conforme observado na Figura 5.9.

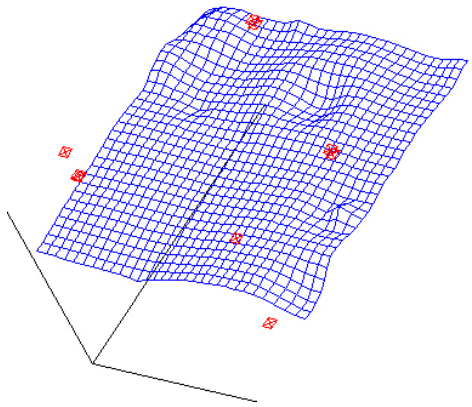

Figura 5.9: Mapa de ativação após treinamento

É montada uma matriz de saída do módulo de rede neural, cuja dimensão será n x 6, onde n corresponde ao número de atributos. Os seis valores para cada atributo são: esquema, tabela, nome do atributo, neurônio BMU, cor, distância em relação ao BMU. Esta matriz será utilizada para elaboração de agrupamentos dos atributos.

5.2.4 Aplicação do modelo e análise

Após o treinamento da rede SOM é necessário que o usuário especialista clique no botão ( ). Através desta opção o especialista pede para a ferramenta agrupar os atributos conforme sua representatividade no mapa de neurônios SOM. Esta representatividade é dada pela matriz fornecida pelo módulo de rede neural. Esta tarefa não está automatizada para permitir que o especialista tenha a possibilidade de interagir com a rede neural, através da observação do diagrama de atuação na superfície do mapa de neurônios, da observação do gráfico de superfície e da U-matriz (ULTSCH, 1993).

O diagrama de atuação na superfície do mapa do mapa de neurônio, visto na Figura 5.6, auxilia o especialista a definir e entender parâmetros como o raio mínimo e máximo na definição da largura da vizinhança do BMU.

O mapa de superfície da Figura 5.9, complementa o diagrama de atuação na superfície. Como se trata de um gráfico com três dimensões é possível a visualização da distância euclidiana do vetor de pesos de cada um dos neurônios.

A matriz de distâncias unificada, U-matriz, tem o objetivo de permitir a detecção visual das relações topológicas entre os neurônios. Usa-se a mesma forma de cálculo de distância usada no treinamento, distância euclidiana, para calcular a distância entre os vetores de discriminadores dos neurônios adjacentes. O resultado gerado a partir da aplicação da U-matriz sobre o mapa é uma imagem *f(x, y)* onde o nível de intensidade de cada pixel corresponde a uma distância calculada dos pesos dos neurônios vizinhos mais próximos.

Dado um mapa bidimensional encontra-se a U-matriz calculando-se as distâncias d_u, d_d, d_l e d_r (Figura 5.10), para cada neurônio. O valor d_m da U-matriz é calculado em função dos valores dos elementos circunvizinhos do neurônio relativo ao dm. O valor d_m pode ser a média, mediana, valor máximo ou mínimo destes valores. Na ferramenta SEA para o cálculo de d_m foi utilizada a média.

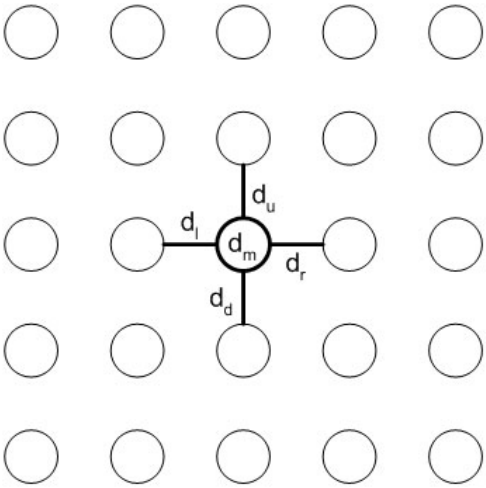

Figura 5.10: Cálculo da U-matriz

A matriz de distância unificada pode ser interpretada como uma imagem através da coloração dos pixels de acordo com a intensidade de cada componente da matriz. Valores altos correspondem a neurônios vizinhos dissimilares e valores baixos correspondem a neurônios vizinhos similares. Regiões com baixos valores do gradiente correspondem a vales que agrupam neurônios especializados em padrões similares. Regiões com valores altos correspondem a fronteiras entre agrupamentos, Veja a Figura 5.11.

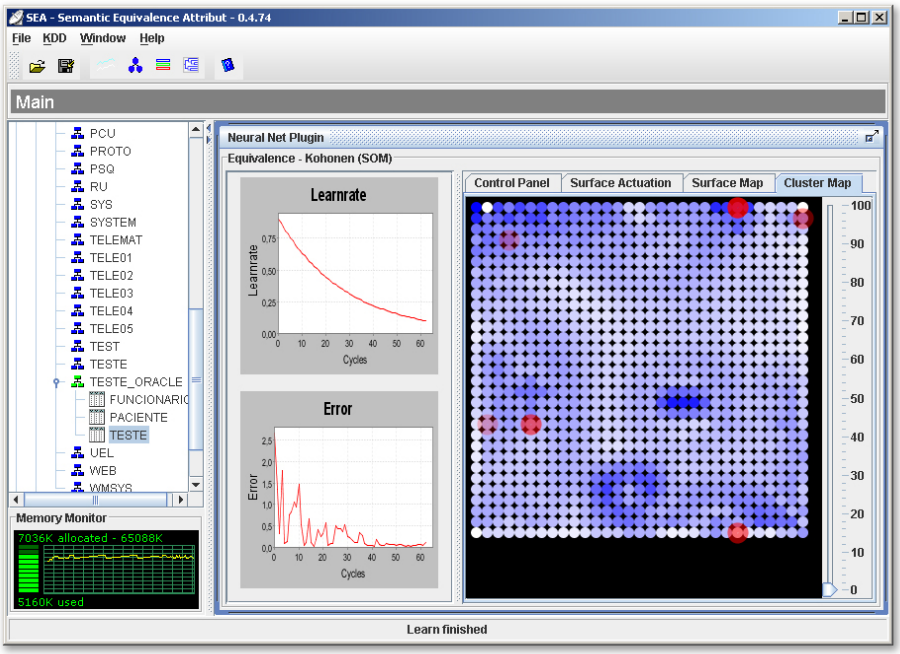

Figura 5.11: U-matriz

Pelo fato da U-matriz gerar uma imagem relativamente complexa (Figura 5.11), sua principal aplicação é a visualização do mapa para separação manual dos agrupamentos.

Porém, já existe alguma iniciativa para detecção automática dos agrupamentos por meio de técnicas de processamento desta imagem (COSTA; NETTO, 2001).

A U-matriz é um método cujo objetivo é permitir a detecção visual das relações topológicas dos neurônios (ULTSCH, 1993). Esta técnica é extremamente útil quando

se tem os vetores de código com dimensão maior que três. Para estes casos não se pode representar graficamente, ou pelo diagrama de Voronoi ou por superfícies de influência, a organização final dos neurônios.

Após a análise opcional dos gráficos citados, o especialista aciona o botão agrupar e volta para a janela que mostra os atributos e seus discrimadores. As linhas da tabelas são coloridas com as cores informadas na matriz criada pelo módulo de rede neural. Estas cores representam o agrupamento dos atributos conforme sua similaridade.

5.3 Implementação

A implementação da metodologia resultou na ferramenta SEA. Foi desenvolvido um ambiente interativo baseado em janelas, como se vê no estudo de caso apresentado na seção 6, onde o usuário especialista pode interagir com a ferramenta, conectando-se a uma ou mais bases de dados, que podem ser gerenciadas por diferentes tipos de SGBD.

A SEA está sendo implementada com base em uma plataforma de software livre, o que não impede que seja utilizada com SGBD comerciais. A linguagem de desenvolvimento escolhida foi a Java 1.4.1_02. Tal escolha deve-se ao fato de esta linguagem ser portável e ter apresentado desempenho satisfatório. A SEA está sendo avaliada em uma plataforma Windows 2000 Server e Windows XP rodando em máquinas Pentium III de 700 MHz e Pentium IV de 2.8 GHz. As bases de dados acessadas são gerenciadas pelos SGBD Oracle versão 8.0.5.0.0 e 9.2.0.4.0. Foram feitos testes com outros tipos de SGBD como: MySQL 4.0.13-nt e Microsoft Access 2000. Estes SGBD diferem em plataforma e escalabilidade e o objetivo destes testes é verificar a capacidade de integração que a SEA possui.

6 ESTUDO DE CASO

Neste capítulo a ferramenta SEA é demonstrada através de um estudo de caso envolvendo esquemas de banco de dados utilizados pela Universidade Estadual de Londrina. Através deste estudo de caso serão demonstradas as fases para descoberta de equivalência semântica de atributo, contempladas pela ferramenta.

O estudo de caso apresentado envolve a combinação de dados reais e dados fictícios para que se tenha um ambiente controlado, no entanto a ferramenta SEA foi utilizada para testes reais na referida universidade e relatos dos usuários comprovaram que a ferramenta auxilia no processo de descoberta de equivalência semântica entre atributos de bancos de dados.

6.1 Conexão com o banco de dados

Para ter acesso aos dados relacionados aos atributos é necessário estabelecer a conexão com um ou mais bancos de dados. Para configurar a conexão é necessário especificar o tipo de conexão, nome da conexão, usuário e senha. Um exemplo de conexão pode ser visualizado na Figura 6.1.

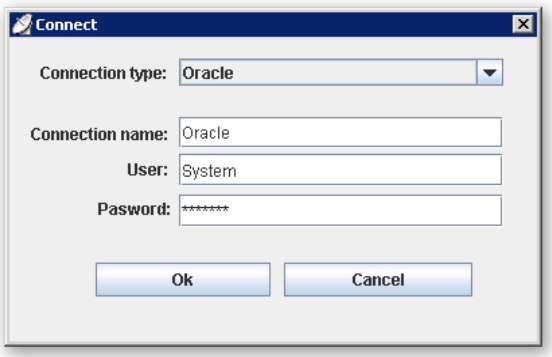

Figura 6.1: Configuração das conexões com bancos de dados

Os banco de dados são apresentados na área esquerda da janela da aplicação, em forma de árvore, como mostra a Figura 6.2. As conexões apresentadas são Oracle, mySQL, Access e File. Para que a ferramenta SEA estabeleça a conexão e necessário clicar no nome da conexão. A conexão efetuada com o banco de dados Oracle pode ser observada na Figura 6.3.

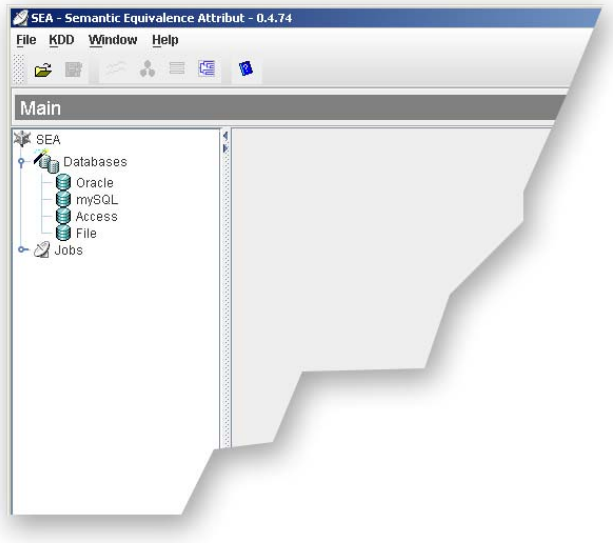

Figura 6.2: Conexões com bancos de dados cadastradas

A conexão com o banco de dados fica ativa até o encerramento da ferramenta. Sua ativação pode ser verificada pela mudança no ícone que a representa. Antes do estabelecimento da conexão este ícone é um tambor e após a conexão este ícone é alterado para um tambor com uma pequena tomada ao lado, indicando que a mesma esta efetivada ou ligada.

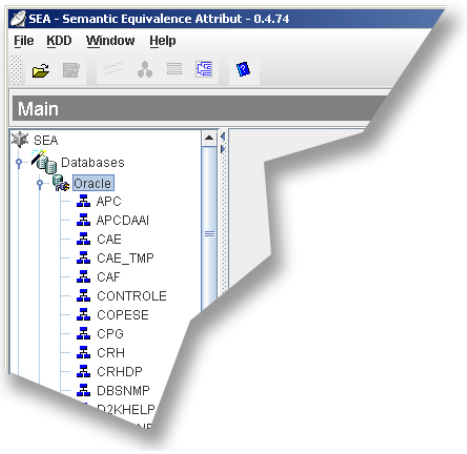

Figura 6.3: Conexão estabelecida com o banco de dados Oracle

6.2 Preparação do ambiente

Uma vez estabelecida a conexão, é necessária a escolha do conjunto de atributos a serem tratados. Os bancos de dados relacionais normalmente trabalham com as seguintes subdivisões: esquemas, tabelas e colunas. A ferramenta SEA está preparada para trabalhar com estas subdivisões. Portanto é possível adicionar um ou mais esquemas, tabelas e colunas. As colunas são tratadas com a nomenclatura atributos.

Para este estudo de caso foi criado um esquema especial com o nome de TESTE_ORACLE. Este esquema é composto de três tabelas: FUNCIONARIO, PACIENTE e TESTE.

A tabela FUNCIONARIO é composta de colunas da tabela que armazena os dados dos funcionários da universidade, conforme descrito na bela 0. em anexo. Ela tem 11.235 tuplas.

A tabela PACIENTE é composta de colunas modificadas e não da tabela FUNCIONARIO, conforme descrito na bela 0.. Ela possui 11.253 tuplas. Ela é uma cópia da tabela FUNCIONARIO com modificações nos atributos em nível de esquema.

A tabela TESTE é composta de colunas modificadas e não modificadas da tabela PACIENTE, conforme descrito na bela 0.. Ela possui 10.136 tuplas e é um subconjunto da tabela PACIENTE.

Nesta fase a ferramenta foi configurada. A SEA permite que parâmetros relacionados a cada fase de utilização da ferramenta sejam alterados. São adicionados os atributos dos esquemas de banco de dados e definidas as operações a serem realizadas com atributos.

Conforme a Figura 6.4, foram adicionados os atributos das três tabelas mencionadas no item 6.1, gerando uma lista de dezesseis atributos com seus respectivos discriminadores.

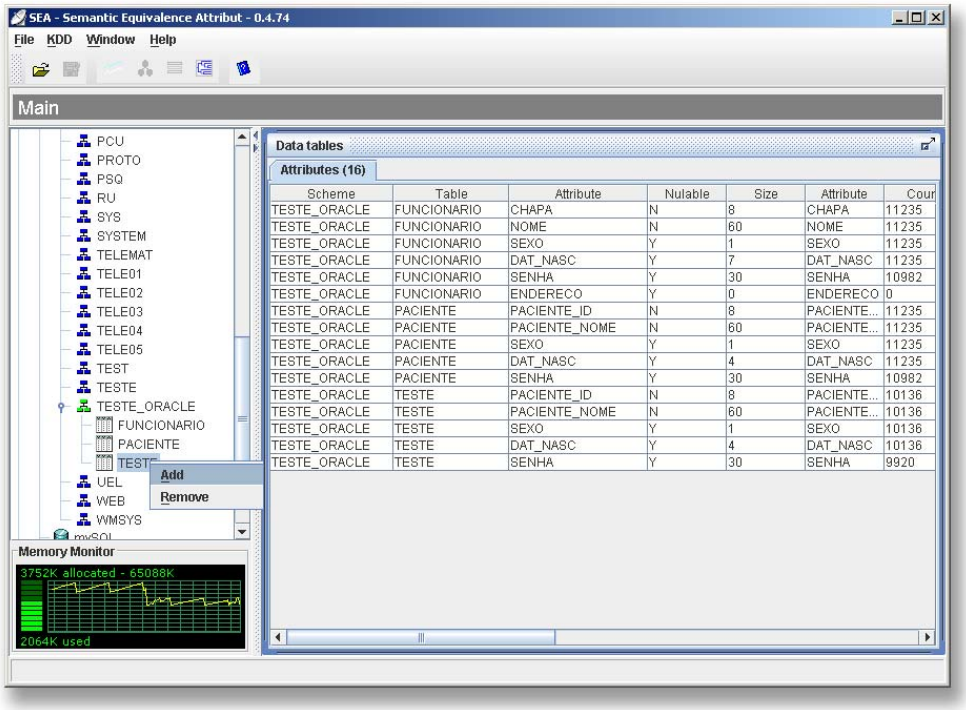

Figura 6.4: Adição de atributos

6.2.1 Configurações para extração dos discriminadores

Nesta etapa foram configurados os valores que influenciam na extração dos discriminadores de cada atributo. O painel de configurações possui duas guias: a) configurações gerais e b) discriminadores. A Figura 6.5 apresenta as opções contidas em configurações gerais.

As opções contidas na guia *General* foram configuradas conforme as Figura 6.5 e Figura 6.6, ou seja, sem os valores simbólicos, com escalonamento sigmoide. Uma vez escolhida a opção de normalizar os discriminadores, não é possível a utilização da opção normalizar todos os discriminadores, pois elas são mutuamente exclusivas. A opção normalizar todos os discriminadores faz sentido quando a opção adicionar valores simbólicos é selecionada.

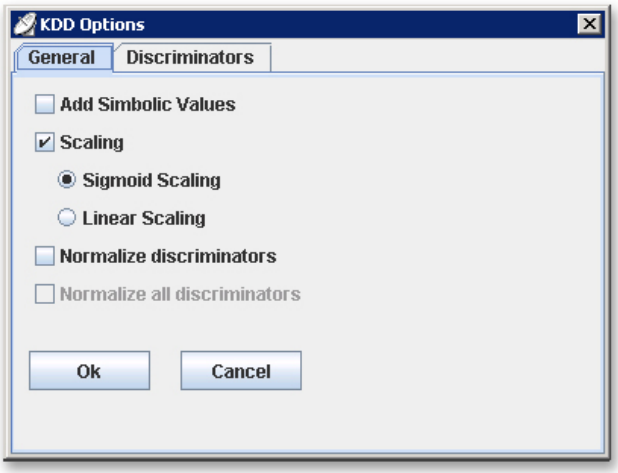

Figura 6.5: Configurações da fase de KDD

O painel também contém os discriminadores dos atributos. Para selecioná-los deve-se alternar para a guia *Discriminators* mostrada na Figura 6.5. Ao selecionar esta guia uma lista com o conjunto de discriminadores pré-estabelecidos é apresentada, conforme se vê na Figura 6.6. Do conjunto de 15 discriminadores, todos foram utilizados para este estudo, porém, a possibilidade de isolar somente os discriminadores que o usuário da ferramenta considere mais significativos é importante e deve ser contemplada nesta fase.

Figura 6.6: Discriminadores

6.2.2 Atributos e seus discriminadores

Nesta fase são adicionados os atributos do(s) esquema(s) de Banco de Dados e os discriminadores de cada atributo.

É possível a adição de todos os atributos de um esquema de cada uma das conexões. Outra possibilidade é a escolha de uma tabela e a adição de seus atributos à lista de atributos a serem analisados. A menor granularidade permitida, até o momento, na ferramenta é a tabela, porém a possibilidade de adição de um único atributo deve ser levada em consideração.

Para adicionar os atributos deve-se clicar com o botão direito do mouse no esquema de banco de dados desejado e usar a opção *add*. A Figura 6.4 mostra um exemplo de atributos adicionados contendo seus discriminadores. Os atributos são apresentados do lado direito da janela.

Para a remoção dos atributos de uma tabela deve-se clicar com o botão direito do mouse sobre a tabela em questão e escolher a opção *remove.*

Após a escolha dos atributos das tabelas com os quais se deseja trabalhar, é necessária a preparação dos valores referentes aos metadados dos atributos. O processo de preparação destes valores é descrito no item 5.2.1.3. Para preparar os metadados e criar os vetores de discriminadores e necessário clicar no botão ( ), este botão ficará ativo, somente após a escolha dos atributos. Surgirá numa tabela a relação dos atributos e seus metadados preparados segundo os critérios estabelecidos pelo usuário. A Figura 6.7 mostra o resultado desta operação. Note que, conforme o critério estabelecido pelo usuário, os valores estão normalizados entre [0,1].

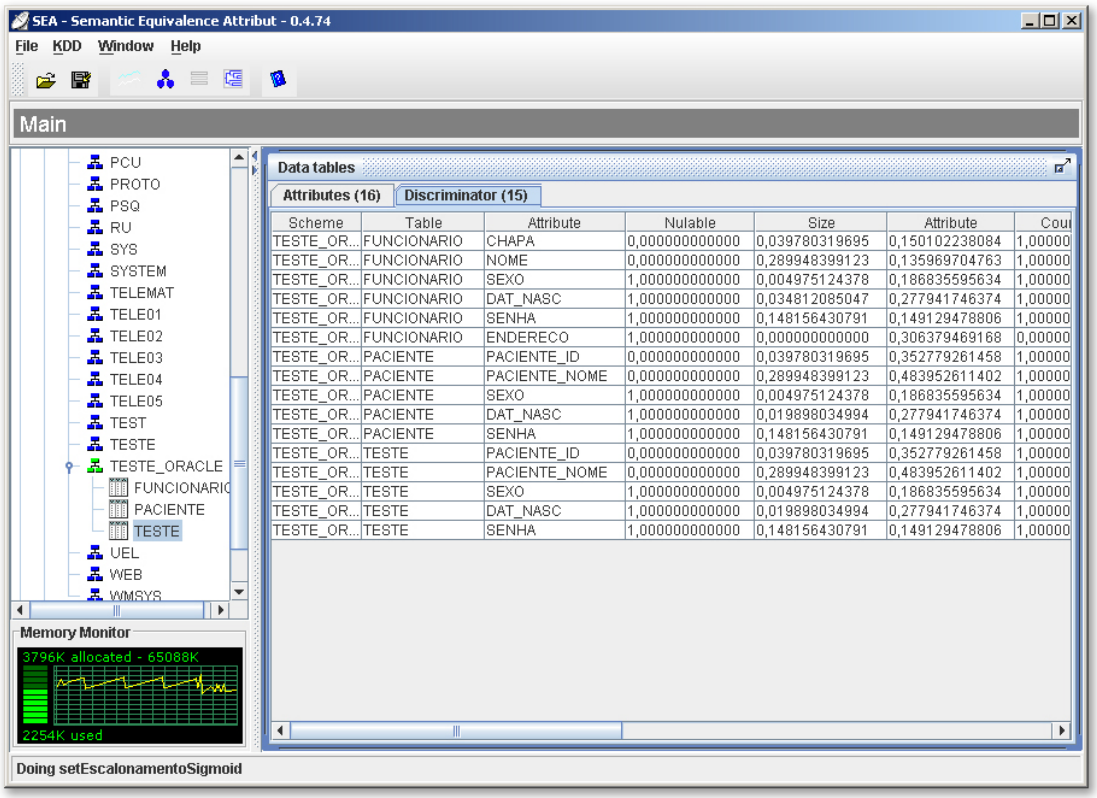

Figura 6.7: Discriminadores dos atributos

6.3 Equivalência de atributos

Os atributos estão escolhidos e seus discriminadores passaram pelo processo de preparação conforme os critérios estabelecidos pelo usuário. É necessário, então, executar o módulo da rede neural, neste caso a rede SOM, para proceder com a

configuração dos valores definidos pelo usuário, que influenciam o comportamento da rede.

6.3.1 Configurações da rede neural

O módulo com a rede neural é executado com um clique no botão ( ). Este botão somente estará ativo após o processamento dos metadados dos atributos e conseqüentemente a criação dos vetores de discriminadores destes atributos.

Antes de iniciar o processo de aprendizagem é necessário estabelecer as configurações para a rede. A Figura 6.8 apresenta o painel de controle contendo um exemplo de configurações para a rede. Estas configurações são descritas a seguir.

O número de neurônios para a rede foi mantido no valor indicado pela ferramenta, gerando uma grade de neurônios de 31 x 31. O número de iterações foi ajustado para 62 e o número de inspeções para 12. Estes dois valores definem respectivamente quantas vezes um vetor de discriminadores do atributo é apresentado à rede e a visualização parcial dos resultados ocorrerá a cada 12 iterações. A visualização parcial é importante, pois permite ao usuário observar o comportamento destes parâmetros. Os valores raio mínimo de 1 e máximo de 15 para a função de vizinhança foram mantidos com os valores sugeridos pela ferramenta, bem como os valores de 0,1 para taxa mínima de aprendizagem e 0,9 para taxa máxima de aprendizagem, sendo que esta varia no intervalo [0,1].

Outras quatro informações são mostradas nesta janela, porém não são passíveis de configuração, servem apenas para auxiliar a visualização da execução do treinamento. São elas: em qual iteração o algoritmo está, quantas épocas já ocorreram, quantos atributos estão sendo tratados e quantos discriminadores cada atributo possui.

Na Figura 6.8 pode-se visualizar um gráfico de aprendizado da rede e outro gráfico que contém a taxa de erro que ocorre durante a aprendizagem.

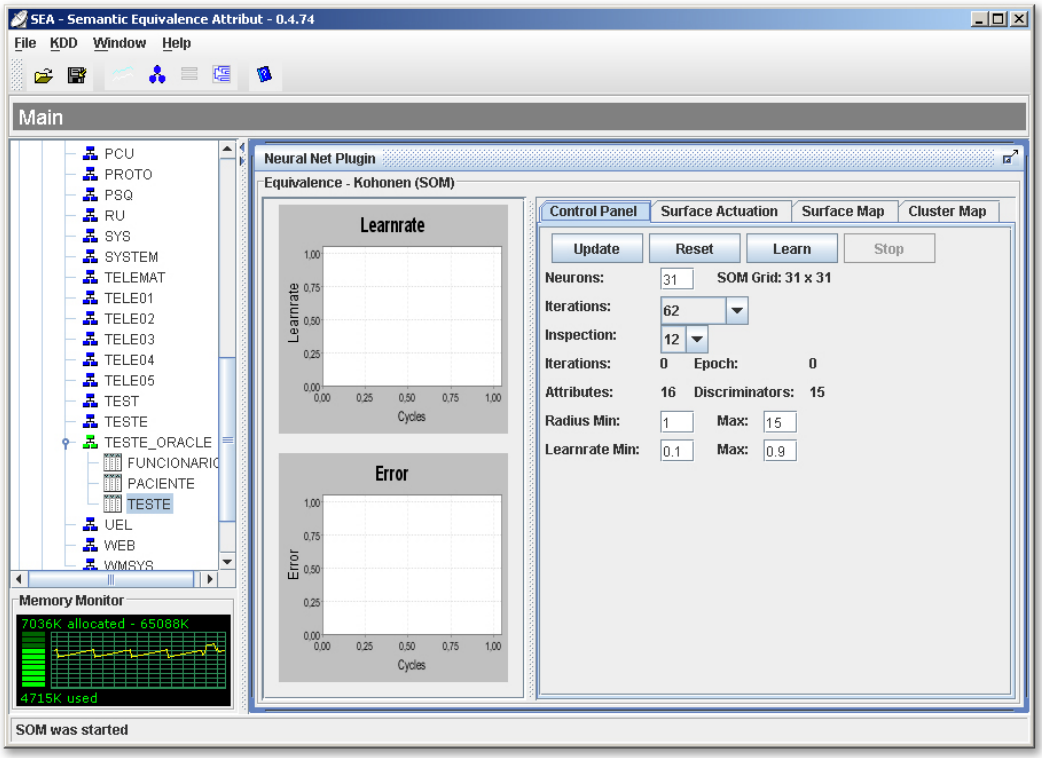

Figura 6.8: Configurações da rede

6.3.2 Aprendizado

Após definir as configurações, pode-se dar início ao treinamento da rede através do botão *Learn* mostrado na Figura 6.8. Inicialmente é atribuído a cada um dos neurônios da grade da rede SOM, um valor aleatório entre 0 e 1 como peso inicial. Durante o processo de aprendizado os pesos são ajustados formando assim novos grupos. Durante a fase de treinamento pode-se alternar de guia para visualizar graficamente o mapa de ajuste dos pesos. Um exemplo do processo de aprendizagem poder ser visualizado na Figura 6.9. É possível a visualização da grade bidimensional de neurônios através da guia *Surface Map*.

Estes modos de visualização podem ajudar o usuário a redefinir os parâmetros de treinamento da rede, porém há a necessidade de o usuário compreender o funcionamento do algoritmo da rede SOM. Porém, testes realizados com pessoas da Universidade Estadual de Londrina, indicaram que mesmo não compreendendo totalmente o funcionamento do algoritmo, estas pessoas conseguiram interagir com o processo de parametrização da rede, de maneira satisfatória, ou seja, alteraram valores com base na percepção dos resultados e obtiveram melhores resultados.

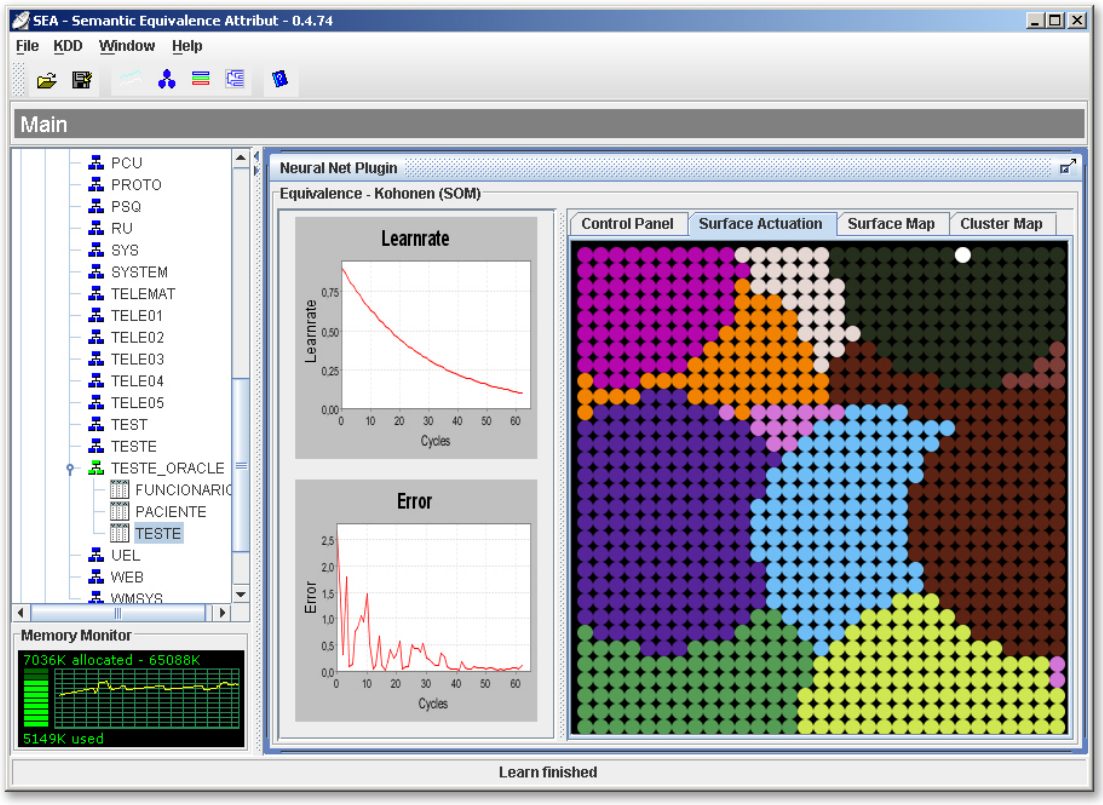

Figura 6.9: Treinamento

6.4 Análise dos resultados

Na Tabela 6.1 podem ser vistos os grupos esperados para o conjunto de atributos proposto neste estudo de caso. Foram encontrados, de forma manual, oito grupos, sendo que os grupos 6 e 8 possuem um atributo cada, portanto os atributos a eles pertencentes não são semanticamente equivalentes a outros atributos.

Tabela 6.1: Agrupamentos esperados

Grupos	Esquema	Tabela	Atributo
1	TESTE_ORACLE	FUNCIONARIO	SEXO
	TESTE_ORACLE	TESTE	SEXO
	TESTE_ORACLE	PACIENTE	SEXO
2	TESTE_ORACLE	PACIENTE	SENHA
	TESTE_ORACLE	TESTE	SENHA
3	TESTE_ORACLE	FUNCIONARIO	SENHA
4	TESTE_ORACLE	TESTE	PACIENTE_NOME
	TESTE_ORACLE	PACIENTE	PACIENTE_NOME
	TESTE_ORACLE	FUNCIONARIO	NOME
5	TESTE_ORACLE	FUNCIONARIO	CHAPA
	TESTE_ORACLE	PACIENTE	PACIENTE_ID
	TESTE_ORACLE	TESTE	PACIENTE_ID
6	TESTE_ORACLE	FUNCIONARIO	DAT_NASC
7	TESTE_ORACLE	TESTE	DAT_NASC
	TESTE_ORACLE	PACIENTE	DAT_NASC
8	TESTE_ORACLE	FUNCIONARIO	ENDERECO

6.4.1 Agrupamento por cores

A análise de agrupamento engloba uma variedade de técnicas e algoritmos cujo objetivo é classificar, com respeito a alguns critérios predeterminados, uma amostra de entidades (indivíduos ou objetos) em grupos mutuamente exclusivos baseados em similaridades entre as entidades. Os grupos de objetos resultantes devem exibir alta homogeneidade interna (dentro do grupo) e alta heterogeneidade externa (entre os grupos). Logo, se houver sucesso na classificação, os objetos dentro dos grupos estarão todos juntos quando apresentados geometricamente, e diferentes grupos estarão separados.

O passo seguinte ao treinamento da rede SOM foi a obtenção do agrupamento por cores. Para este procedimento foi pressionado o botão *Groups* ( ) da barra de ferramentas da SEA. Após este procedimento foi necessário voltar à janela *Data tables* para observar os grupos sugeridos pela ferramenta. As cores foram escolhidas aleatoriamente pela ferramenta e aplicadas à tabela com os atributos e seus discriminadores antes da normalização e à tabela com os atributos e seus discriminadores após a normalização.

A estas tabelas foram acrescentadas três novas colunas: *Neuron*, que mostra em qual neurônio do mapa de neurônios o atributo mais se assemelha; *Color*, que mostra o código da cor que foi atribuída ao neurônio e finalmente *Distance*, que mostra o valor da distância euclidiana do vetor de entrada (discriminadores) do atributo em relação ao vetor de saída (pesos) do neurônio que mais se assemelha ao atributo. Veja a Figura 6.10.

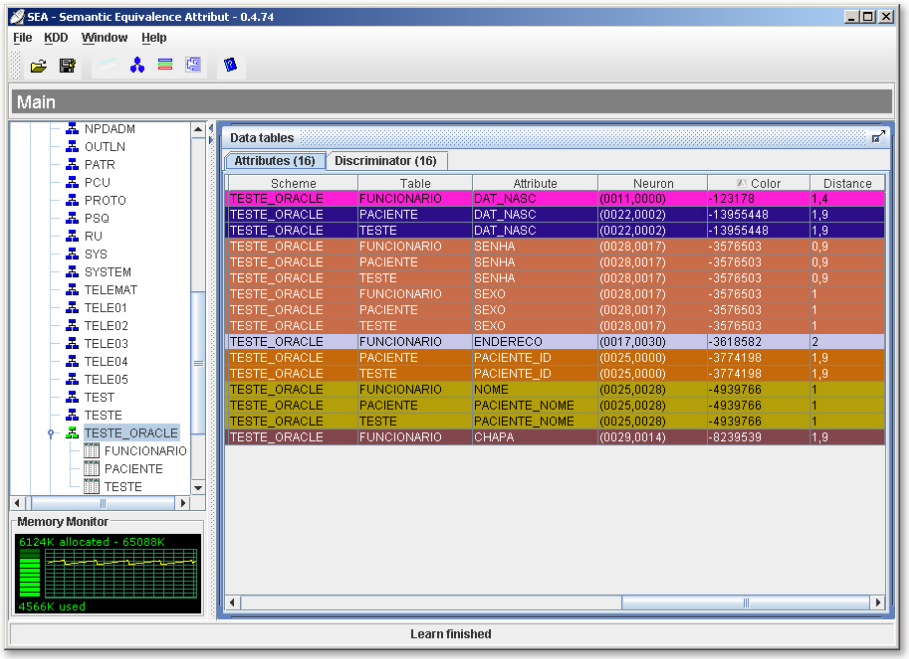

Figura 6.10: Atributos agrupados

A possibilidade de a ferramenta agrupar atributos semanticamente diferentes existe e pode ser observada na Figura 6.11, onde os atributos DAT_NASC e ENDERECO da tabela FUNCIONARIO foram incorretamente agrupados, porém note que a distância dos vetor de discriminadores dos atributos em relação ao neurônio da rede SOM é significativamente diferente. Este agrupamento incorreto é chamado de falso positivo, que é o fato da ferramenta indicar um agrupamento e este não refletir a realidade dos grupos.

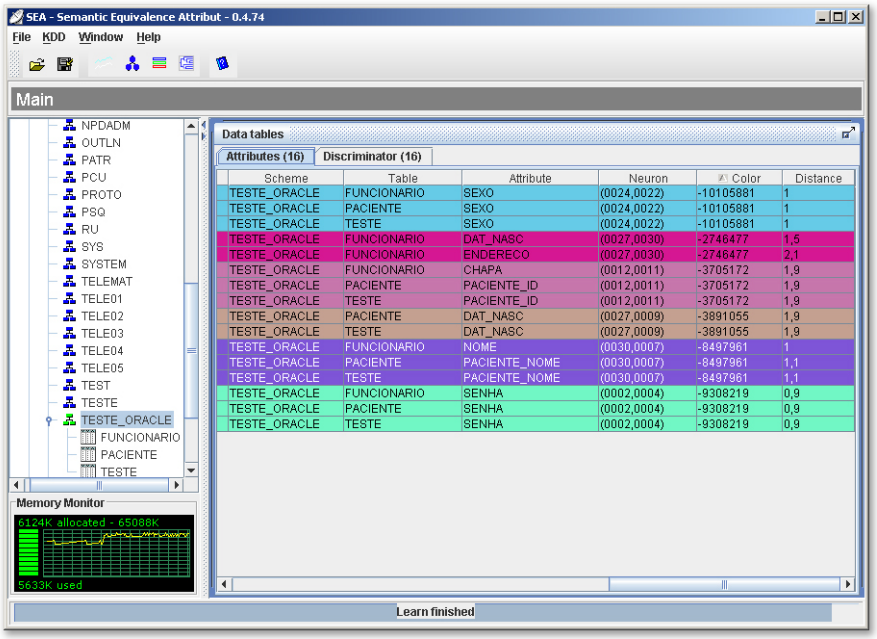

Figura 6.11: Falso positivo

6.4.2 Dendograma

O dendograma foi implementado com a finalidade de auxiliar a visualização dos agrupamentos obtidos. Na Figura 6.12, os agrupamentos podem ser observados de acordo com vários níveis de distância. Para este estudo de caso os parâmetros foram alterados para média, distância euclidiana e utilização de todos os atributos.

Os agrupamentos obtidos são os mesmos indicados pela ferramenta utilizando a rede SOM, conforme o nível de distância ser observado.

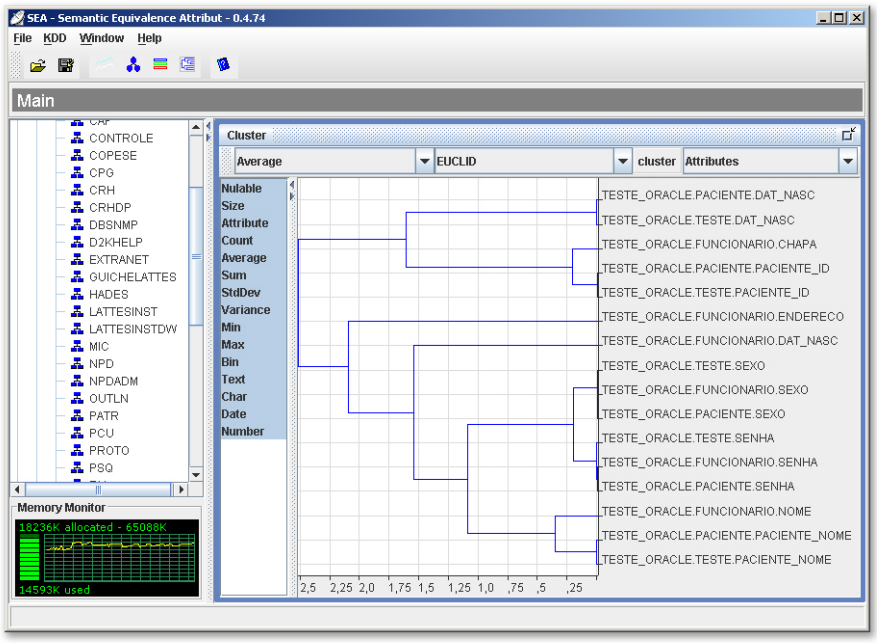

Figura 6.12: Dendograma

6.4.3 Formação dos grupos

Objetivando a compreensão dos resultados foram aplicadas duas unidades de medida de sucesso / falha dos resultados obtidos. Estas unidades de medida são a acurácia e a precisão. A acurácia indica a relação existente entre o número de grupos de atributos que, comprovadamente, existem e os encontrados pela ferramenta SEA que realmente estão corretos. A precisão indica a relação existente entre o número de grupos de atributos que, comprovadamente, existem (ver Tabela 6.1) e os encontrados pela ferramenta SEA. Por exemplo, caso a ferramenta encontre quatro grupos, dos quais três estão corretos e o número de grupos existente seja cinco a acurácia será de 60% (3/5) e a precisão será de 75% (3/4).

Para este estudo decidiu-se por executar a ferramenta por dez vezes sobre o mesmo conjunto de dados (atributos e seus discriminadores) e fazer a análise dos resultados.

Os resultados para a acurácia e precisão para cada execução podem ser observados na Figura 6.13.

Para as dez execuções do teste, a média para a acurácia foi de 62,0% e para a precisão foi de 65,7%.

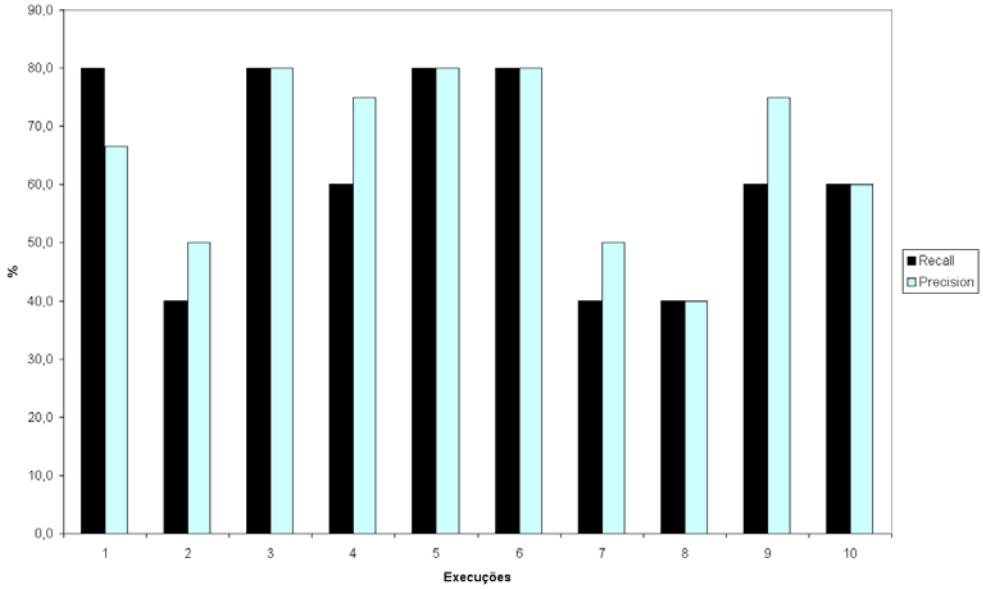

Figura 6.13: Acurácia e precisão

O processo de intervenção humana não está implementado na ferramenta SEA até o momento. Porém, é possível a sua simulação.

Os grupos apresentados pela ferramenta são formados pelos atributos que supostamente são semanticamente equivalentes. Estes grupos são formados quando se encontra o neurônio na rede SOM cujos valores do vetor de saída mais se aproximam dos valores do vetor de entrada dos atributos analisados. Esta verificação e feita através da distância euclidiana entre os vetores. Para cada atributo é calculada a distância euclidiana. Com base neste valor o usuário pode restringir a diferença máxima entre as distâncias, criando assim, a possibilidade de surgirem novos grupos baseados no desmembramento dos grupos propostos pela ferramenta. Na Figura 6.14 podem ser observados os valores para acurácia e precisão resultantes de cada teste realizado sobre os atributos, após a simulação da intervenção humana.

Para simular a intervenção humana foi considerado que distâncias com mais de 10% de diferença fariam com que os atributos correspondentes deixassem de fazer parte do grupo, podendo fazer parte de outro grupo com distâncias similares, apesar dos atributos estarem mais próximos dos mesmos neurônios. Isto pode ser observado na Figura 6.10, onde os atributos senha e sexo das tabelas FUNCIONARIO, PACIENTE e TESTE (6 atributos) estão no mesmo grupo pois o vetor de entrada dos mesmos (discriminadores) está mais parecido com o vetor de saída do neurônio 28,17 do mapa de neurônios rede SOM, após o treinamento. Porém, a distância dos atributos sexo em relação ao neurônio 28,17 é diferente (1) em relação a dos atributos senha (0,9).

Com a intervenção humana o caso citado gerará um novo grupo, dividindo o grupo de seis atributos em dois com três.

Os resultados para a acurácia e a precisão podem ser observados na Figura 6.14. Algumas execuções não tiveram seus resultados alterados. Isto demonstra que o treinamento foi suficientemente capaz de separar estes atributos, porém tal fato não pode ser garantido para todas as execuções. De modo geral o desempenho tende a melhorar com a intervenção humana.

Após a simulação de intervenção humana, a média para a acurácia ficou em 70,0% e para a precisão em 71,0%. Com relação à intervenção humana, há dois fatores a se destacar: o primeiro está relacionado ao fato de o usuário não necessitar, previamente, saber qual a natureza dos atributos para aplicar um limiar de poda para as diferenças

entre as distâncias no mesmo grupo. O segundo diz respeito ao fato de este tipo de intervenção não ser a única possível. Por exemplo, o usuário, a partir dos resultados obtidos, pode realimentar o processo voltando à fase de extração dos discriminadores e selecionar os que melhor evidenciam a formação dos grupos.

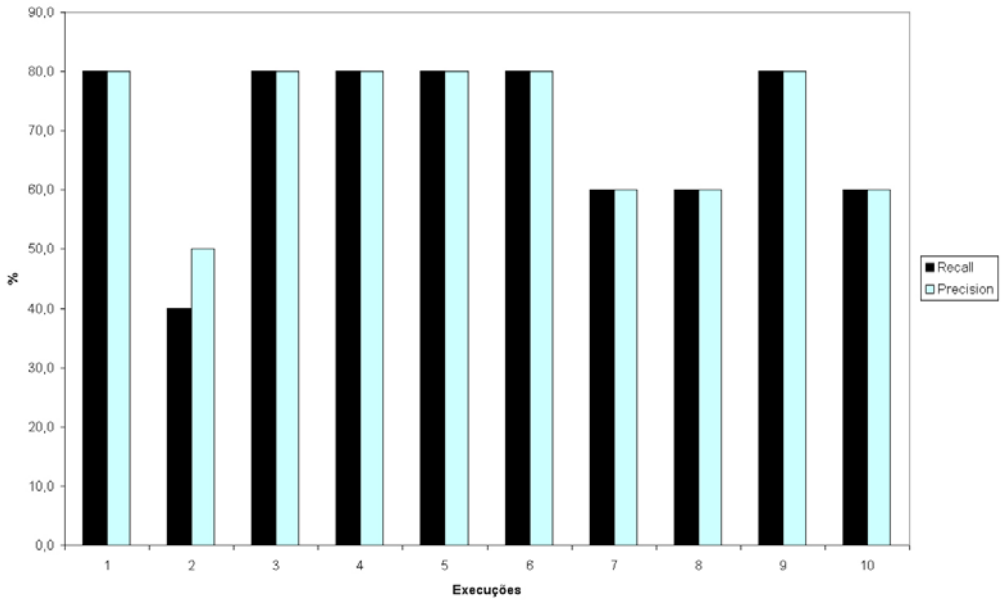

Figura 6.14: Acurácia e precisão após intervenção humana

7 CONCLUSÕES E TRABALHOS FUTUROS

Nesta seção serão apresentadas as conclusões a respeito desta pesquisa e trabalhos futuros relacionados à mesma.

7.1 Conclusões

Com o crescimento das empresas que fazem uso das tecnologias de bancos de dados, os administradores destes bancos de dados criam novos esquemas a cada instante e não existe uma normalização ou formalização para que tal tarefa seja desempenhada de forma homogênea, resultando assim, em bases de dados incompatíveis, o que dificulta a troca de dados entre as mesmas (ELMASRI; NAVATHE, 2000).

Quando os Sistemas de Bancos de Dados são projetados e implementados de forma independente, ou seja, sem se levar em consideração a existência de outro sistema de dados, é provável que existam incompatibilidades entre os dados destes sistemas. Sendo estes sistemas vitais para a sobrevivência da informação, é necessário que sejam averiguadas redundâncias e integridade entre os dados.

A imprecisão na informação devido à existência de mais de uma fonte de dados, uma maior necessidade na capacidade de armazenamento e processamento, um custo maior na manutenção das bases de dados, são conseqüências de atributos especificados de forma redundante.

A descoberta de equivalência semântica de esquemas e conseqüentemente de atributos é um problema relacionado a vários domínios de aplicação em bancos de dados, como integração entre bancos de dados heterogêneos, comércio eletrônico, data warehouse, processamento de consultas semânticas.

O desenvolvimento da ferramenta SEA levou em consideração tais necessidades e a contribuição para a solução problema pode ser evidenciada pela comparação das características com ferramentas existentes. A SEA não objetiva substituir outras ferramentas e também não é a solução definitiva para o problema de descoberta de equivalência semântica entre atributos, principalmente em se considerando a questão automação, uma vez que, como a maioria das ferramentas analisadas nesta pesquisa, exige a validação dos resultados pelo usuário.

Não se pode afirmar que as redes neurais artificiais são a solução definitiva para resolver problemas como o proposto nesta pesquisa, porém fica evidenciado que até o momento é uma das técnicas que apresenta bons resultados. A escolha da rede neural do tipo SOM como sendo a primeira rede implementada na ferramenta SEA mostrou-se satisfatória e abre caminho para a implementação de outros tipos de redes e até mesmo técnicas híbridas.

A elaboração do framework para criação da ferramenta contribui para que outras ferramentas semelhantes possam ser elaboradas, utilizando outras linguagens, aplicadas a outros domínios de problemas relacionados.

A comparação dos resultados obtidos com a ferramenta SEA em relação a outras ferramentas é uma tarefa complexa. Enquanto algumas ferramentas exploram propriedades dos atributos e do esquema (CUPID, SF e COMA), outras se utilizam das instâncias dos atributos em redes Bayesianas (LSD e GLUE) ou redes neurais (SemInt). Contudo pode-se observar que a composição de técnicas é uma solução eficiente para o problema de descoberta de equivalência semântica entre atributos.

Por outro lado as redes neurais (SemInt e SEA) são pouco exploradas para a solução do problema, assim como a utilização de grandes dicionários.

A ferramenta SEA foi avaliada com bases de dados da Universidade Estadual de Londrina e foram obtidos resultados esperados. Com uso da ferramenta foram evidenciadas redundâncias de dados e problemas de integridade de dados entre as bases de dados das unidades reitoria e hospital universitário.

7.2 Trabalhos Futuros

A ferramenta SEA encontra-se em fase inicial de desenvolvimento. Portanto algumas limitações aqui citadas decorrem deste fato. No entanto existem limitações que ocorrem devido às características e a complexidade do processo de DSEA.

7.2.1 Falso agrupamento

Supondo-se que exista um atributo A_1 e um outro atributo A_2 com características parecidas, ou seja, discriminadores semelhantes, estes podem ser considerados equivalentes. Por exemplo, se em uma tabela existe um atributo cujo nome é SEXO e em outra tabela existe outro atributo com nome de SITUAÇÃO. Os dois atributos são do tipo numérico, não podem conter nulos, tem tamanho de 1 byte e seguem uma distribuição das tuplas de forma muito parecida, pois na maioria dos casos, quando o SEXO for igual a 1 a SITUAÇÃO também será igual a 1. A ferramenta identificará estes atributos como semanticamente equivalentes e para o usuário eles não são. Como trabalho futuro pode-se estudar métodos para solução deste fato.

7.2.2 Ênfase nos mesmos tipos de dados

Como a ferramenta obtém metadados de dados do tipo texto de forma diferenciada dos dados numéricos, é possível que um atributo A_1 do tipo texto possa ser considerado diferente de outro atributo A_2 do tipo data, apesar de os dois serem semanticamente equivalentes. Por exemplo, considere um atributo DATA_ANIVERSARIO e outro DT_ANI, sendo a primeira do tipo texto e a segunda do tipo data. Os dois seguem a mesma distribuição e tem valores como 01/01/1922. Seus discriminadores, em nível de dados, serão diferentes e o agrupamento dos atributos será prejudicado. Como trabalho futuro é possível o desenvolvimento de métodos para os extratores, que contemplem esta situação.

7.2.3 Definição de um limiar para agrupamento

Os atributos estão sendo agrupados com um limiar de distância muito pequeno. Como trabalho futuro este limiar poderia ser ajustado automaticamente ou com intervenção humana para realimentar o processo e formar novos agrupamentos. Com isto novos grupos surgiriam e outros seriam divididos. Hoje este limiar pode ser observado pelo especialista, mas ele não interage com este parâmetro. A forma ideal de implementação desta funcionalidade seria a automática.

7.2.4 Aperfeiçoamento do módulo extrator

O módulo extrator tem um papel relevante no processo de descoberta de equivalência semântica entre atributos. Atualmente, para atributos com tipo de dados igual a texto este módulo considera o número de caracteres nas instâncias do atributo. Para uma melhor comparação com atributos do tipo numérico é importante que estas instâncias sejam tratadas como as dos atributos numéricos ou vice-versa.

Como trabalho futuro, a implementação desta funcionalidade aumentaria o desempenho da ferramenta em relação a acurácia na indicação dos grupos.

7.2.5 Chaves estrangeiras

Um aspecto em relação às chaves estrangeiras é que é importante que esta restrição de integridade conhecida, ou seja, que esteja modelada e criada no sistema de banco de dados, seja levada em consideração na descoberta de equivalência semântica entre atributos, pois é natural que um atributo que represente uma chave estrangeira seja semanticamente equivalente ao atributo na tabela de origem. Como trabalho futuro, esta verificação poderia ser acrescida à fase de análise e aplicação do modelo.

REFERÊNCIAS

AGRAWAL, R.; SRIKANT, R. Mining Sequential Patterns. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING. ICDE, 1995, Taipei, Taiwan. **Proceedings…** Disponível em: <http://citeseer.nj.nec.com/agrawal95mining.html>. Acesso em: fev. 2002.

AGRAWAL, R. et al. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, ACM SIGMOD, 1998, Seattle, Washington. **Proceedings...** Disponível em: <http://www.cs.cobrnell.edu/johannes/papers/1998/sigmod1998-clique.pdf>. Acesso em: jan. 2002.

ALMEIDA, M. A. F. **Tutorial sobre Redes Neurais Artificiais**. Florianópolis, Santa Catarina: CPGCC da UFSC, 1995. Disponível em: <http://www.inf.ufsc.br/~mafa/rna.pdf > Acesso em: fev. 2002.

BALDI, P.; HORNIK, K. Neural networks and principal components analysis: learning from examples without local minima. **Neural Networks**, [S.l.], v. 2, p. 53-58, 1989.

BELL, S. Discovery and maintenance of functional dependencies by independencies. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY IN DATABASES, KDD, 1., 1995. **Proceedings…**
Disponível em: <http://citeseer.nj.nec.com/214033.html>. Acesso em: abr. 2004.

BERLIN, J.; MOTRO, A. Autoplex: Automated Discovery of Content for Virtual Databases. In: INTERNATIONAL CONFERENCE ON COOPERATIVE INFORMATION SYSTEMS, COOPIS, 2001, Trento, Italy. **Proceedings…** [S. l.: s.n.], 2001. p. 108-122.
Disponível em: <http://www.isse.gmu.edu/~ami/research/publications/pdf/coopis01.pdf >. Acesso em: abr. 2004.

BERLIN, J.; MOTRO, A. Database Schema Matching Using Machine Learning with Feature Selection. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, CAiSE, 2002, Toronto, Canadá. **Proceedings …**
Disponível em: <http://www.isse.gmu.edu/~ami/research/publications/pdf/caise02.pdf >. Acesso em: abr. 2004.

BENEVENTANO, D. et al. The MOMIS approach to Information Integration. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, 1., 2001, Setúbal, Portugal. **Proceedings …**
Disponível em: <http://dbgroup.unimo.it/Momis/articles.html>. Acesso em: mar. 2004.

BERGAMASCHI, S. et al. Semantic integration of heterogeneous information sources. Data & Knowledge Engineering , [S.l.], v. 36, 3, p. 215–249, 2001.
Disponível em: <http://islab.dico.unimi.it/main.php?obj=publication&mode=view &key =label%7CDKE-2001 >. Acesso em: abr. 2004.

BRAGA, A. C.; LUDERMIR T. **RedesNeurais Artificiais:** Teoria e Aplicações. Rio de Janeiro: LTC, 2000.

CARPENTER, G. A.; GROSSBERG, S.; ROSEN, D. B. ART2: Self-organization of stable category recognition codes for analog input patterns. **Neural Networks,** [S. l.], v. 4. p. 493-504, 1991.

CARPENTER, G. A.; GROSSBERG, S. The ART of adaptive pattern recognition by a self-organizing neural network. **Computer**, [S. l.], v. 21, p. 77-88, 1988. (CARPENTER; GROSSBERG, 1988)

CARPENTER, G. A.; GROSSBERG, S. ART3: Hieralchical search using chemical transmitters in self-organizing pattern recognition architectures. **Neural Networks**, [S. l.], v. 3, p. 129-152,1990.

CARRIÓN, G. A. **Integración de esquemas en bases de datos heterogéneas fuertemente acopladas**. 1999, Tesis (Maestría). Departamento de Ingeniería en Sistemas Computacionales, Universidad de las Américas-Puebla, Cholula, Puebla, México. (CARRIÓN, 1999)
Disponível em: <http://mailweb.udlap.mx/~tesis/msp/alvarez_c_g/indice.html>. Acesso em: set. 2002.

CARVALHO, J. V. de. Utilização de Técnicas de "Data Mining" para o Reconhecimento de Caracteres Manuscritos**.** In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 14., 1999, Florianópolis. **Anais …** Florianópolis: UFSCA, 1999.
Disponível em: <http://www.inf.ufsc.br/sbbd99/anais/SBBD-Completo/20.pdf >. Acesso em: fev. 2002.

CARVALHO, L. A. V. de. **Dataming:** a mineração de dados no marketing, medicina, engenharia e administração. São Paulo: Érica, 2001.

CLIFTON, C.; HOUSMAN, E.; ROSENTHAL, A. Experience with a Combined Approach to Attribute-Matching Across Heterogeneous Databases. In: THE IFIP WORKING CONFERENCE ON DATA SEMANTICS, , 1996. **Proceedings …**
Disponível em: <http://www.mitre.org/tech/itc/staffpages/arnie/pubs/ds-7.pdf >. Acesso em: abr. 2004.

COSTA, J. A. F.; NETTO, M. L. de A. **Clustering of complex shaped data sets via Kohonen maps and mathematical morphology.**
Disponível em: < www.dee.ufrn.br/alfredo/Costa_&_Netto__SPIE_DM& KD_2001 .pdf > . Acesso em: fev. 2004.

CUNHA, J. F. **Sistemas de Informação Modelação do Conhecimento e Bases de Dados**. Faculdade de Engenharia da Universidade do Porto. 2001. (CUNHA, 2001)
Disponível em: <http://paginas.fe.up.pt/~jlborges/SI.html>. Acesso em: maio 2004.

DO, H. H.; RAHM, E. COMA – A System for Flexible Combination of Schema Matching Approach. In VERY LARGE DATABASES, VLDB, 2002. **Proceedings …**
Disponível em: <http://www.vldb.org/conf/2002/S17P03.pdf >. Acesso em: abr. 2004.

DOAN, A.; DOMINGOS P.; HALEVY, A. Reconciling Schemas of Disparate Data

Sources: A Machine-Learning Approach. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD, 2001. **Proceedings …**
Disponível em: <http://www.cs.washington.edu/homes/pedrod/papers/sigmod01.pdf >. Acesso em: abr. 2004.

DOAN A. et al. Learning to Map between Ontologies on the Semantic Web. In: INTERNATIONAL WORLD WIDE WEB CONFERENCE, WWW, 2002, Honolulu, Hawaii, USA. **Proceedings …**
Disponível em: < http://www.cs.washington.edu/homes/jayant/Pubs/ GlueWWW02.pdf >. Acesso em: abr. 2004.

ELMASRI, R.; NAVATHE, S. B. **Fundamentals of database systems**. 3 rd ed. New York – NY: Addison-Wesley, 2000. p. 478-483.

ENGEL, P. M. **Descoberta de Conhecimento em Banco de Dados**. Porto Alegre: PPGC da UFRGS, 2001 . Notas de Aula.

ERWIN, E.; OBERMAYER, K.; SCHULTEN, K. I. Self-Organizing Maps**:** Stationary states, metastability and convergence rate. **Biological Cybernetics**, [S. l.], v. 67, p. 35-45, 1992.

FAUSETT L. V. **Fundamentals of Neural Networks:** Architectures, Algorithms, and Applications. Engleweed Cliffs, NJ: Prentice Hall International Editions, 1994.

FAYYAD, U. M. et al. From Data Mining to Knowledge Discovery: An Overview. In: FAYYAD, U. M. et al. **Advances in Knowledge Discovery and Data Mining**. Menlo Park: AAAI Press, 1995.

FAYYAD, U. M. et al. From Data Mining to Knowledge Discovery. In: FAYYAD, U. M. et al. **Advances in Knowledge Discovery and Data Mining** , Menlo Park: AAAI Press, 1996.

GIUNCHIGLIA F.; SHVAIKO P. Semantic matching. In: WORKSHOP ON ONTOLOGIES AND DISTRIBUTED SYSTEMS, ODS, 2003. **Proceedings …**
Disponível em: < http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS// Vol-71/ Giunchiglia.pdf >. Acesso em: abr. 2004.

HAN, J. et al. Data-Driven Discovery of Quantitative Rules in Relational Databases. **IEEE Transactions on Knowledge and Data Engineering**, [S. l.], v. 5, n. 1, p. 29 – 40, Feb. 1993.

HAYKIN, S. **Redes Neurais Princípios e Prática.** Trad. Paulo Martins Engel. 2. ed. Porto Alegre: Bookman, 2001.

HEBB, D. O. **The Organization of Behavior**. New York: Wiley, 1949.

HEUSER, C. A. **Projeto de banco de dados**. 3. ed. Porto Alegre: Sagra Luzzatto, 2000. p. 135-147

HOPFIELD J. J. **Neural networks and physical systems with emergent collective computational abilities**. **Proceedings of the National Academic of Sciences of United States of America**, Washington, v. 79, 1982.
Disponível em: <http://www.pubmedcentral.nih.gov/picrender.fcgi?artid=346238& action=stream&blobtype=pdf >. Acesso em: jan. 2002.

KIM, W.; SEO, Y. Classifying schematic and data heterogeneity in multidatabase systems. **Computer**, [S. l.], v. 24, n. 12, , p. 12-18, Dec. 1991. Disponível em: <http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=116884>. Acesso em: maio 2002.

KOHONEN, T. ***Self*-Organization *and Associate Memory*.** 3 rd ed. Berlin: Springer-Verlag, 1989.

KOHONEN, T. **The self-organizing map**. New York: IEEE, 1990. p. 1464-1480.

LEE, Y.; MOON, S. Heterogeneous schema integration method for multidatabase system. **Microprocessing and Microprogramming**, [S. l.], v.38, p. 265-272, 1993.

LI, W. S.; CLIFTON, C. Semantic Integration in Heterogeneous Databases Using Neural Networks. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 1994. **Proceedings …**
Disponível em: <http://www.vldb.org/conf/1994/P001.PDF>. Acesso em: abr. 2004.

LI, W. S.; CLIFTON, C. SemInt: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Network. **Data and Knowledge Engineering,** [S. l.], v.33, n. 1, p. 49–84, 2000.

LINSKER, R. From basic network principles to neural architecture: Emergence of orientation-selective cells. **Proceedings of the National Academic of Sciences of United States of America**, Washington, v. 83, p. 7508 – 7512, 1986. Disponível em: <http://citeseer.comp.nus.edu.sg/context/17891/0>. Acesso em: abr. 2004.

LOPES, C. H. P. **Classificação de Registros em Banco de Dados por Evolução de Regras de Associação Utilizando Algoritmos Genéticos.** 1999, Dissertação (Mestrado) - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica, Rio de Janeiro.
Disponível em: <http://www.ica.ele.puc-rio.br/publicacoes/pbl_abstract.asp? Referencia= tes_0002 >. Acesso em: mar. 2002.

LO, Z.; FUJITA, M.; BAVARIAN, B. Analysis of neighborhood interaction in Kohonen Neural networks. In: INTERNATIONAL PARALLEL PROCESSING SYMPOSIUM, IPPS , 1991. **Proceedings …**
Disponível em: <http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel2/489/4017/00153786.pdf?arnumber=153786>. Acesso mar. 2002.

MADHAVAN, J.; BERNSTEIN, P. A.; RAHM, E. Generic Schema Matching with Cupid. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 2001. **Proceedings …**
Disponível em: <http://research.microsoft.com/~philbe/CupidVLDB01.pdf >. Acesso em: abr. 2004.

MALSBURG, V. C. An automaton with brain-like properties. **International Journal of Systems & Cybernetics**, [S. l.], v. 14, p. 85 - 100, 1973.

MELNIK, S.; GARCIA-MOLINA, H.; RAHM, E. Similarity Flooding: A Versatile Graph Matching Algorithm. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 2002. **Proceedings …**
Disponível em: <http://www-db.stanford.edu/~melnik/pub/melnik_ICDE02.pdf>. Acesso em: abr. 2004.

MENDONÇA NETO, M. G. de. **Mineração de dados**. Universidade Salvador. Departamento de Ciências Exatas, Núcleo de Pesquisa em Redes de Computadores e Núcleo de Energia - Petróleo e Petroquímica. Material publicado nos Anais da 6a Escola Regional de Informática da Regional 2 do Estado de São Paulo, São Carlos, São Paulo, abril de 2001. Disponível em: <http://www.nuperc.unifacs.br/RT-NUPERC-

2001-3p.pdf>. Acesso em: jul. 2002.

MILLER, R. et al. The Clio Project: Managing Heterogeneity. **SIGMOD Record**, New York, v. 30, n.1, p. 78–83, 2001. Disponível em: <http://citeseer.ist.psu.edu/452072.html >. Acesso em: abr. 2004

MILO, T.; ZOHAR, S. Using Schema Matching to Simplify Heterogeneous Data Translation. In: VERY LARGE DATABASES, VLDB, 1998. **Proceedings …**
Disponível em: <http://www.vldb.org/conf/1998/p122.pdf >. Acesso em: abr. 2004.

MITRA, P.; WIEDERHOLD, G.; JANNINK, J. Semi-automatic Integration of Knowledge Sources. In: INT. CONF. ON INFORMATION, FUSION, 2., 1999. **Proceedings …**
Disponível em: <http://www-db.stanford.edu/SKC/publications/match.ps >. Acesso em: abr. 2004.

MURAD, M. M.; LOMBARDI, C. R. **Estudo do modelo de Rede Neural Artificial - SOM (Kohonen) "SOMA" (Self-Organizing Maps Analyze).** 2003, Projeto Final, Universidade Católica de Brasília, Brasília .
Disponível em: <http://www.ucb.br/prg/professores/rogerio/diretorios/Kohonen/Kohonen%20monografia_Final.pdf >. Acesso em: fev. 2004.

NEVES, M. C. et al. **Mineração de dados em grandes bancos de dados geográficos**. [S. l.]: INPE, 2001. Relatório Técnico.
Disponível em: <http://www.dpi.inpe.br/geopro/modelagem/relatorio_data_mining.pdf>. Acesso em: fev. 2002.

OZSU, M. T.; VALDURIEZ, P. **Principles of Distributed Database Systems.** 2 nd ed. New Jersey: Prentice-Hall, 1999.

RAHM, E.; BERNSTEIN, P.A. A Survey of Approaches to Automatic Schema Matching. **VLDB Journal**, [S. l.], v.10, n. 4, 2001.
Disponível em: <http://research.microsoft.com/~philbe/VLDBJ-Dec2001.pdf >. Acesso em: abr. 2004.

RAHM, E.; BERNSTEIN, P.A. **On Matching Schemas Automatically**. [S. l.: s. n.], 2000. (Technical Report MSR-TR-2001-17).

RITTER, H.; MARTINETZ, T. M.; SCHULTEN, K. **Neural computation and self-organizing maps:** An introduction. Reading, MA: Addison-Wesley, 1992.

ROMÃO, W.; FREITAS, A. A; PACHECO, R. C. S. Uma revisão de abordagens genético-difusas para descoberta de conhecimento em banco de dados. **Revista Acta Scientiarum**, Maringá, v.22, n.5, p. 1347 – 1359, Dec. 2000. Disponível em: <http://www.din.uem.br/~wesley/ >. Acesso em: fev. 2002.

SAVASERE, A.; OMIECINSKI, E.; NAVATHE, S. An Efficient Algorithm for Mining Association Rules in Large Database**s.** In: VERY LARGE DATA BASES, VLDB, 21., Zurich, Swizerland. **Proceedings …** San Francisco: Margan Koufmann, 1995. p. 432 – 444. Disponível em:
<http://delab.csd.auth.gr/~manolopo/datamin/P432.pdf>. Acesso em: jun. 2003.

SHETH, P.; LARSON, J. Federated database system for managing distribuited, heterogeneus, and autonomous databases**. ACM Computing Surveys**, New York, v. 22, n.3, p. 183-235, Sept. 1990.

SILVA, L. N. de C. **Análise e Síntese de Estratégias de Aprendizado para Redes Neurais Artificiais.** Campinas: Departamento de Engenharia de Computação e Automação Industrial Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, SP, Brasil, 1998.

SILVA, M. A. **Mapas Auto-Organizáveis na Análise Exploratória de Dados Geoespaciais Multivariados**. 2004. Dissertação (Mestrado) - Ministério da Ciência e Tecnologia, INPE, São José dos Pinhais.

SPACCAPETRA, S.; PARENT, C.; DUPONT, Y. Model Independent Assertions for Integration of Heterogeneous Schemas. **Very Large Database Journal**, [S.1.], v.1, n.1, p.81-126, 1992.

SRIKANT, R.; AGRAWAL, R. Mining sequential patterns: Generalizations and performance improvements. In: INTERNATIONAL CONF. ON EXTENDING DATABASE TECHNOLOGY, EDBT, 5., 1996, Avgnon, France. **Advances in Database Technology:** proceedings. Berlin: Springer – Verlag, 1996. Disponível em: <http://citeseer.nj.nec.com/51015.html>. Acesso em: fev. 2002.

SRIKANT, R.; AGRAWAL, R. Mining Quantitative Association Rules in Large Relational Tables. In: ACM-SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD, 1996, Montreal, Canada. Disponível em: <http://www.cs.uky.edu/~dekhtyar/685-Spring2003/literature/rules.html>. Acesso em: fev. 2002.

YAN, L.L. et al. Data-Driven Understanding and Refinement of Schema Mappings. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD, 2001. **Proceedings …**
Disponível em: <http://www.almaden.ibm.com/cs/people/fagin/sigmod01.pdf >. Acesso em: abr. 2004

VARGAS, E. C**. Recuperação de informação por similaridade utilizando técnicas inteligentes.** 2004. Tese (Doutorado). Instituto de Ciências Matemáticas e de Computação -ICMC/USP, Campinas, SP, BR.

VILAS BOAS, R. M. F. **XMLS+Matcher**: Um Método para Identificação de Correspondências entre Esquemas XML Schemas Semânticos**.** 2002. Dissertação (Mestrado) - Universidade Federal do Ceará, Departamento de Computação. Disponível em <www.mcc.ufc.br/disser/RenataFigueredo.pdf>. Acesso em: maio 2004.

PALOPOLI, L.; TERRACINA G.; URSINO D. Experiences using DIKE, a system for supporting cooperative information system and data warehouse design. **Informations Systems,** [S. l.], v. 28, n. 7, p. 835-865, 2003.

SUN, X. L. Automated Schema Matching Techniques: An Exploratory Study. **Lett**. **Inf. Math. Sci**, [S. l.], v.4, p. 113-136, 2003.
Disponível em: <http://iims.massey.ac.nz /research/letters/>. Acesso em: fev. 2004.

ULTSCH, **A.** Knowledge extraction from self-organizing neural networks. In: OPITZ, O.; LAUSEN, B. ; KLAR, R. (Ed.). **Information and Classification.** Berlin: Springer Verlag, 1993.
Disponível em : <www.informatik.uni-marburg.de/ ~databionics/papers/93KD.pdf>. Acesso em: mar. 2004.

Tabela 8.1: Atributos e seus metadados

Scheme	Table	Attribute	Nulable	Size	Name	Count	Average	Sum	StdDev	Variance	Min	Max	New Type	Type	Reserved	Reserved	Reserved
TESTE_ORACLE	FUNCIONARIO	CHAPA	N	8	CHAPA	11.235	56.224,9	631.687.276	32.452,1	1.053.137.063,9	112516	15	NUMBER	NUMBER	empty	empty	empty
TESTE_ORACLE	FUNCIONARIO	NOME	N	60	NOME	11.235	23,0	257.994	5,8	34,0	48	8	CHAR	VARCHAR2	empty	empty	empty
TESTE_ORACLE	FUNCIONARIO	SEXO	Y	1	SEXO	11.235	1,0	11.235	0,0	0,0	1	1	CHAR	CHAR	empty	empty	empty
TESTE_ORACLE	FUNCIONARIO	DAT_NASC	Y	7	DAT_NASC	11.235	8,0	89.880	0,0	0,0	31/12/1973	1/1/2026	DATE	DATE	empty	empty	empty
TESTE_ORACLE	FUNCIONARIO	SENHA	Y	30	SENHA	10.982	18,3	200.544	2,5	6,4	20	6	CHAR	NCHAR	empty	empty	empty
TESTE_ORACLE	FUNCIONARIO	ENDERECO	Y	0	ENDERECO	0	0,0	0	0,0	0,0	0	0	TEXT	LONG	empty	empty	empty
TESTE_ORACLE	PACIENTE	PACIENTE_ID	N	8	PACIENTE_ID	11.235	56.224,9	631.687.276	32.452,1	1.053.137.063,9	112516	15	NUMBER	NUMBER	empty	empty	empty
TESTE_ORACLE	PACIENTE	PACIENTE_NOME	N	60	PACIENTE_NOME	11.235	23,0	257.994	5,8	34,0	48	8	CHAR	VARCHAR2	empty	empty	empty
TESTE_ORACLE	PACIENTE	SEXO	Y	1	SEXO	11.235	1,0	11.235	0,0	0,0	1	1	CHAR	CHAR	empty	empty	empty
TESTE_ORACLE	PACIENTE	DAT_NASC	Y	4	DAT_NASC	11.235	1.977,5	22.217.523	31,4	985,3	2049	1950	NUMBER	NUMBER	empty	empty	empty
TESTE_ORACLE	PACIENTE	SENHA	Y	30	SENHA	10.982	18,3	200.544	2,5	6,4	20	6	CHAR	NCHAR	empty	empty	empty
TESTE_ORACLE	TESTE	PACIENTE_ID	N	8	PACIENTE_ID	10.136	56.889,8	576.635.270	32.579,1	1.061.394.689,9	112516	15	NUMBER	NUMBER	empty	empty	empty
TESTE_ORACLE	TESTE	PACIENTE_NOME	N	60	PACIENTE_NOME	10.136	23,1	234.263	5,9	34,4	48	8	CHAR	VARCHAR2	empty	empty	empty
TESTE_ORACLE	TESTE	SEXO	Y	1	SEXO	10.136	1,0	10.136	0,0	0,0	1	1	CHAR	CHAR	empty	empty	empty
TESTE_ORACLE	TESTE	DAT_NASC	Y	4	DAT_NASC	10.136	1.977,1	20.039.501	30,9	954,9	2049	1950	NUMBER	NUMBER	empty	empty	empty
TESTE_ORACLE	TESTE	SENHA	Y	30	SENHA	9.920	18,3	181.220	2,5	6,3	20	6	CHAR	NCHAR	empty	empty	empty

Tabela 8.2: Atributos com discriminadores

Scheme	Table	Attribute	Nulable	Size	Name	Count	Average	Sum	StdDev	Variance	Min	Max	Bin	Text	Char	Date	Number
TESTE_ORACLE	FUNCIONARIO	CHAPA	0,0000000000	0,0397803197	0,1501022381	1,0000000000	1,0000000000	1,0000000000	1,0000000000	1,0000000000	1,0000000000	0,0744892494	0,0000000000	0,0000000000	0,0000000000	0,0000000000	1,0000000000
TESTE_ORACLE	FUNCIONARIO	NOME	0,0000000000	0,2899483991	0,1359697048	1,0000000000	0,1137523219	1,0000000000	0,0289848099	0,1673676384	0,2343694839	0,0397803197	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000
TESTE_ORACLE	FUNCIONARIO	SEXO	1,0000000000	0,0049751244	0,1868355956	1,0000000000	0,0049751244	1,0000000000	0,0000000000	0,0000000000	0,0049751244	0,0049751244	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000
TESTE_ORACLE	FUNCIONARIO	DAT_NASC	1,0000000000	0,0348120850	0,2779417464	1,0000000000	0,0397803197	1,0000000000	0,0000000000	0,0000000000	0,0422637150	0,0323273069	0,0000000000	0,0000000000	0,0000000000	1,0000000000	0,0000000000
TESTE_ORACLE	FUNCIONARIO	SENHA	1,0000000000	0,1481564308	0,1491294788	1,0000000000	0,0906031187	1,0000000000	0,0125917602	0,0318613554	0,0991762128	0,0298421292	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000
TESTE_ORACLE	FUNCIONARIO	ENDERECO	1,0000000000	0,0000000000	0,3063794692	0,0000000000	0,0000000000	0,0000000000	0,0000000000	0,0000000000	0,0000000000	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000	0,0000000000
TESTE_ORACLE	PACIENTE	PACIENTE_ID	0,0000000000	0,0397803197	0,3527792615	1,0000000000	1,0000000000	1,0000000000	1,0000000000	1,0000000000	1,0000000000	0,0744892494	0,0000000000	0,0000000000	0,0000000000	0,0000000000	1,0000000000
TESTE_ORACLE	PACIENTE	PACIENTE_NOME	0,0000000000	0,2899483991	0,4839526114	1,0000000000	0,1137523219	1,0000000000	0,0289848099	0,1673676384	0,2343694839	0,0397803197	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000
TESTE_ORACLE	PACIENTE	SEXO	1,0000000000	0,0049751244	0,1868355956	1,0000000000	0,0049751244	1,0000000000	0,0000000000	0,0000000000	0,0049751244	0,0049751244	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000
TESTE_ORACLE	PACIENTE	DAT_NASC	1,0000000000	0,0198980350	0,2779417464	1,0000000000	0,9999999943	1,0000000000	0,1549113852	0,9998895635	0,9999999972	0,9999999925	0,0000000000	0,0000000000	0,0000000000	0,0000000000	1,0000000000
TESTE_ORACLE	PACIENTE	SENHA	1,0000000000	0,1481564308	0,1491294788	1,0000000000	0,0906031187	1,0000000000	0,0125917602	0,0318613554	0,0991762128	0,0298421292	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000
TESTE_ORACLE	TESTE	PACIENTE_ID	0,0000000000	0,0397803197	0,3527792615	1,0000000000	1,0000000000	1,0000000000	1,0000000000	1,0000000000	1,0000000000	0,0744892494	0,0000000000	0,0000000000	0,0000000000	0,0000000000	1,0000000000
TESTE_ORACLE	TESTE	PACIENTE_NOME	0,0000000000	0,2899483991	0,4839526114	1,0000000000	0,1144818034	1,0000000000	0,0291815973	0,1696055597	0,2343694839	0,0397803197	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000
TESTE_ORACLE	TESTE	SEXO	1,0000000000	0,0049751244	0,1868355956	1,0000000000	0,0049751244	1,0000000000	0,0000000000	0,0000000000	0,0049751244	0,0049751244	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000
TESTE_ORACLE	TESTE	DAT_NASC	1,0000000000	0,0198980350	0,2779417464	1,0000000000	0,9999999943	1,0000000000	0,1525384520	0,9998505144	0,9999999972	0,9999999925	0,0000000000	0,0000000000	0,0000000000	0,0000000000	1,0000000000
TESTE_ORACLE	TESTE	SENHA	1,0000000000	0,1481564308	0,1491294788	1,0000000000	0,0906376122	1,0000000000	0,0125177519	0,0314881341	0,0991762128	0,0298421292	0,0000000000	0,0000000000	1,0000000000	0,0000000000	0,0000000000

Tabela 8.3: Características das técnicas de combinação de esquemas

		SemInt	**LSD**	**SKAT**	**TranScm**	**DIKE**	**ARTEMIS**	**CUPID**
Tipo de esquema		Relacional, arquivos	XML	XML, IDL, Texto	SGML, OO	ER	Relacional, OO, ER	XML, Relacional
Representação de Metadados		Não especificado (baseado em atributos)	XML esquema	Grafo baseado em modelo de dados OO	Grafo rotulado	grafo	Híbrido relacional/ OO/ modelo de dados	ER estendido
Granularidade das combinações		Em nível de elemento: Atributo (grupos de atributos)	Em nível de elemento e estrutura	Em nível de Elemento/ Estrutura / classes	Em nível de elemento	Em nível de Elemento / Estrutura: Entidades / relacionamentos / atributos	Em nível de Elemento/ Estrutura: Entidades / relacionamentos / atributos	Em nível de elemento e estrutura
Cardinalidade das combinações		1:1	1:1	1:1 e n:1	1:1	1:1	1:1	1:1 e n:1
Combinações em nível de esquema	Baseadas em nome	-	Igualdade de nomes / sinônimos	Igualdade de nomes; sinônimos; homônimos; hipernônimos	Igualdade de nomes; sinônimos; homônimos; hipernônimos	Igualdade de nomes; sinônimos; hipernônimos	Igualdade de nomes; sinônimos; hipernônimos	Igualdade de nomes; sinônimos; homônimos; hipernônimos; abreviações
	Baseadas em restrições	Vários critérios: tipo de dados, tamanho, informações sobre chaves,...	-	É-um (inclusão); cardinalidade dos relacionamentos	É-um (inclusão); cardinalidade dos relacionamentos	Compatibilidade de Domínio	Compatibilidade de Domínio. No MOMIS, usa chaves, chaves estrangeiras, é-um, agregação	Tipo de dados e compatibilidade de domínios, constraints referenciais
Combinação em nível de instância	Orientadas a texto	-	Redes Bayesianas	-	-	-	-	-
	Orientadas a restrições	Caractere / padrões de dados numéricos, distribuição de valores, médias	Lista de domínios válidos	-	-	-	-	-
Reutilização /informação auxiliar utilizada		-	Busca de domínios válidos	Reutilização de regras de combinação genéricas	-	Provisão de alguns sinônimos + inclusão com probabilidades de similaridade	Dicionário	Dicionário, glossário
Combinação da combinadores		Combinadores híbridos	Automático: combinação ponderada de todos os combinadores; combinação de resultados	híbridos	Combinadores híbridos; Ordem fixa dos combinadores	híbridos	Híbrido de combinadores de nome e de estrutura	híbridos
Trabalho manual		Seleção do critério de combinação (opcional); seleção das combinações de atributos dos grupos de atributos	O usuário fornece combinações para o treinamento; o usuário pode especificar parâmetros e *constraints* de integridade para a grade de candidatos a combinações	Regras de similaridade /dessimilaridade + refinamento interativo	Resolver múltiplas combinações adicionando novas regras de combinações	Resolver conflitos estruturais (reprocessamento); especificação de alguns sinônimos + inclusões com probabilidade de similaridade; validação	O usuário pode ajustar pesos no calculo das combinações e escolher combinações válidas	O usuário pode ajustar o limiar dos pesos
Área de aplicação		Integração de dados	Integração de dados com um esquema global pré-definido	Composição de Ontologias para suporte a integração de dados / interoperabilidade	Tradução de dados	Integração de esquemas	Integração de esquemas	Tradução de dados, mas com o objetivo de ser genérico.
Outros		Redes Neurais	-	"Algoritmos" implicitamente representados por regras	Regras implementadas em java	Algoritmos para calcular novos sinônimos, homônimos, métricas de similaridade	Faz parte do mediador MOMIS	-