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

Regras de Associação Aplicadas aos Filtros de Mensagens e Canais de Informação do Projeto Direto

por

MICHELE FRIGHETTO

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Ciência da Computação

Prof. Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, junho de 2003.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Frighetto, Michele

Regras de Associação Aplicadas aos Filtros de Mensagens e Canais de Informação do Projeto Direto / por Michele Frighetto. – Porto Alegre: PPGC da UFRGS, 2003.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós Graduação em Computação, Porto Alegre, BR-RS, 2003. Orientador: Geyer, Cláudio Fernando Resin.

1. Descoberta de Conhecimento. 2. Mineração de Dados. 3. Regras de Associação. 4. Listas de Discussão. 5. Canais de Informação 6. Filtros de Mensagens 7. Projeto Direto I. Geyer, Cláudio Fernando Resin. II.Título.

Agradecimentos

Ao Prof. Dr. Cláudio Fernando Resin Geyer pelo indispensável auxílio fornecido durante o desenvolvimento do trabalho.

A minha família.

Aos professores, colegas e funcionários do Instituto de Informática da UFRGS.

Regras de Associação Aplicadas aos Filtros de Mensagens e Canais de Informação do Projeto Direto

Resumo

Neste trabalho é apresentado um breve estudo sobre o processo de descoberta de conhecimento em banco de dados, com enfoque na etapa de mineração de dados através de regras de associação. Propostas por Agrawal em 1993, num estudo chamado análise de cesta de mercado, as regras de associação representam que com um certo grau de suporte e confiança um conjunto de itens pode estar presente numa transação visto que outro conjunto está presente.

A necessidade de análise semelhante às realizadas por Agrawal surgiu em outros campos e estas foram estendidas a outras aplicações. Neste, são apresentadas as principais variações sobre o tema regras de associação encontradas na literatura.

É proposta a mineração de dados através de regras de associação sobre filtros de mensagens e canais de informação do software de catálogo, agenda e correio eletrônico Direto. Para as pesquisas são utilizadas três ferramentas: Intelligent Miner, CBA e Magnus Opus. Elas foram aplicadas sobre uma lista de discussão da Linguagem Java, pois o projeto Direto ainda não possui mensagens públicas. As ferramentas possuem características distintas: o Intelligent Miner permite a definição de hierarquias sobre os dados que serão minerados; o Magnus Opus trabalha com diversos filtros e com a definição de intervalos para o tratamento de campos numéricos; o CBA permite que sejam especificados suportes múltiplos para os itens.

Palavras-Chave: Mineração de Dados, Regras de Associação, Listas de Discussão, Filtros de Mensagens, Canais de Informação, Intelligent Miner, Magnus Opus e CBA.

Tittle: “Association Rules Applied to Messages Filters and Information Channel in the Direto Environment”

Abstract

This work presents a brief review about knowledge discovery in database having association rules as the data mining process. Association rules were proposed by Agrawal in 1993 in a basket data analysis. Association rules have been extended to other applications because there is a necessity for similar Agrawal’s analysis in different domains. Here are presented some variations proposed in the literature about association rules along with the main algorithms. This work proposes the use of association rules over message filters and information channels from the Direto, which is a catalog, schedule and e-mail software. Three data mining tools were used: Intelligent Miner, CBA and Magnus Opus. They were applied over a Java discussion list because Direto project does not have public messages. Each tool has distinct features: Intelligent Miner allows to define a hierarchy over the data that will be mined; Magnus Opus works with many filters over the data and permits to define ranges over numeric fields and CBA allows to specify multiple minimum support over the items.

Key-Words: Data Mining, Association Rules, Discussion List, Message Filters, Information Channel, Intelligent Miner, Magnus Opus and CBA.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

OLAP	On-Line Analytic Processing Systems
BDE	Borland Database Engine
ODBC	Open Database Connectivity
Sup	Suporte
Conf	Confiança
Cover	Coveraege
Lever	Levereage
LHS	Left Hand Side
RHS	Right Hand Side

Lista de Símbolos

+	Soma
-	Subtração
*	Multiplicação
/	Divisão
=	Igualdade
<>	Diferença
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Menor ou igual a
∩	Intersecção
Φ	Conjunto Vazio
| |	Módulo
∀	Para todo
⊂	Está Contido
U	União
∈	Pertence
¬	Negação

Sumário

1 Introdução

Segundo [FAY 96], o crescimento explosivo no volume dos bancos de dados interfere na habilidade de interpretação dos mesmos, criando a necessidade de uma nova geração de ferramentas para automatizar uma análise inteligente. As ferramentas e técnicas desenvolvidas para suprir tal necessidade originaram uma nova área denominada descoberta de conhecimento em banco de dados. O processo de descoberta de conhecimento, segundo [FAY 96], envolve diversos passos e muitas decisões que devem ser feitas pelo usuário.

No processo de descoberta de conhecimento, destacamos neste trabalho um estudo sobre a técnica de mineração de dados através de regras de associação. Na literatura foram encontrados diversos trabalhos sobre o tema aplicados a diferentes áreas. Neste, é proposta a mineração de dados sobre os filtros de mensagens e canais de informação do software de correio, agenda e catálago eletrônico Direto. Para mineração através de regras de associação são utilizadas três ferramentas que possuem funcionalidades diferentes: Intelligent Miner, Magnus Opus e CBA.

1.1 Motivação

Desde mil novecentos e noventa e três, quando Agrawal propôs o algoritmo Apriori para análise de cesta de mercado, as regras de associação ganharam uma atenção especial quando o assunto envolvido é Mineração de Dados. Um exemplo típico de regras de associação é encontrado em transações de compras (*market basket analisys*), onde a partir de uma banco de dados na qual registram-se os itens adquiridos por cliente, uma estratégia de mineração poderia gerar o seguinte exemplo: {pão, leite} → manteiga (Suporte 2%, Confiança 40%). A regra indica que um cliente que compra pão e leite, também compra manteiga. O suporte indica a proporção em que os itens (pão, leite e manteiga) estão presentes nas transações do banco de dados e a confiança indica a probabilidade que um caso satisfaça o consequente da regra, se satisfizer o antecedente.

A utilização de regras de associação em ferramentas de mineração de dados e em sistemas acadêmicos evidenciou a necessidade de se definir e oferecer aos usuários variações para este tipo de regras.

Em [AGR 99] foi proposto um estudo que utiliza além das medidas tradicionais (suporte e confiança), uma medida adicional, chamada *Improvment*, capaz de filtrar as regras de associação mais interessantes em bancos de dados muito densos. Segundo esta proposta são válidas somente as regras onde o antecedente composto de uma regra, possui confiança superior à de suas subregras. As subregras derivadas de uma regra são formadas pela decomposição da regra, fixando-se o conseqüente e combinando os itens do seu antecedente.

Em [HAN 97] são propostas regras de associação representadas através de uma estrutura hierárquica de árvore e denominadas regras taxonômicas. Nos nodos superiores são representados os níveis mais generalizados e nos inferiores os mais específicos. Se considerarmos como exemplo, produtos de um supermercado, o nível superior poderia ser representado por “leite” e o seu nodo-filho por “leite achocolatado Parmalat”. O algoritmo Apriori é aplicado e são eliminadas as regras redundantes, isto é, as que podem ser obtidas

em níveis mais altos da hierarquia, sem apresentarem grandes diferenças nas medidas de suporte e confiança.

Em [SRI 96] são apresentadas as Regras de Associação Quantitativas, onde é aplicado aos dados existentes um processamento inicial, mapeando-os a uma tabela lógica. Cada atributo origina uma coluna na nova tabela. A presença de um atributo é representada pelo valor um, e a ausência por zero. Quando a quantidade de atributos for muito grande, estes podem ser agrupados em intervalos, porém é preciso atenção ao definir-se o agrupamento de atributos, pois estes podem apresentar seus valores para suporte e confiança muito diferente do que se fossem tratados individualmente.

No estudo realizado por [SAV 98] é considerado um problema complementar, denominado regras de associação negativas. Enquanto as regras de associação convencionais identificam itens que aparecem juntos em um número significativo de transações, as regras de associação negativas representam que, com certo grau de certeza, determinados itens não ocorrem quando outros específicos estão presentes.

Grupos de pesquisa do Instituto de Informática da UFRGS têm trabalhado com o tema Regras de Associação. [CER 99] implementou um protótipo de uma ferramenta que pode ser utilizada sobre vários bancos de dados para extração de regras de associação. O protótipo permite alguma iteração com o usuário e foi validado sobre um banco de dados do SUS (Sistema Único de Saúde). [BRU 2000] realizou um estudo sobre a utilização de técnicas de mineração de dados aplicadas ao uso na Web e propôs um modelo para mineração através de regras de associação, neste ambiente.

1.2 Proposta

Com a necessidade de distribuir informações de qualidade através das instituições estaduais e o surgimento da idéia de filtros de mensagens e canais de informações para o Direto, surgiu a atual proposta que tem por objetivo melhorar ainda mais os serviços de informações disponibilizados pelo software. Através do estudo realizado sobre a aplicação de regras de associação, concluí-se que poderiam ser empregadas com sucesso no Projeto Direto auxiliando na identificação de perfis de usuários. Os usuários do Direto podem definir dois tipos de filtros: um filtro para mensagens de usuários e outro filtro para o serviço de canais de informação. A aplicação de regras de associação sobre as mensagens dos serviços poderia auxiliar na criação, manutenção e exclusão de canais de informação. Além disso poderiam ser oferecidos determinados canais a conjuntos específicos de usuários e sugerida a utilização de determinados filtros.

A presente proposta é a de minerar, através de regras de associação, a utilização de filtros de mensagens e canais de informação, através de três ferramentas distintas que implementam as regras de associação e possuem características distintas.

A primeira ferramenta escolhida é o Intelligent Miner da IBM, que além de implementar o tradicional algoritmo Apriori permite a especificação de uma taxonomia para os dados.

O CBA é a segunda ferramenta a ser utilizada. [CBA 99] foi desenvolvida e cedida pelo grupo de pesquisas de Ling Biu, da Universidade de Singapura. Uma das propostas do autor é a importância da análise de itens que ocorrem com pouca freqüência. Para isto, a ferramenta permite que se especifique diferentes valores de suporte aos itens que estão sendo analisados.

O Magnus Opus é a terceira ferramenta por apresentar um mecanismo de classificação de dados em intervalos, semelhantes aos propostos por [SRI 96]. [MAG 2001] possui alguns filtros como *Association Rule Strenght*, *Lift*, *Coverage e Leverage*, que podem ser úteis na análise dos resultados obtidos.

Para exemplificar a potencialidade e características das ferramentas aplicáveis as listas, neste trabalho é utilizado um caso específico de listas de discussão sobre a linguagem de programação JAVA. As mensagens da lista Java foram utilizadas como dados de entrada para a mineração pois, o Direto ainda não possui mensagens públicas.

1.3 Organização do Texto

O restante do texto está dividido em seis capítulos. No capítulo 2 são apresentados os conceitos sobre descoberta de conhecimento, os trabalhos que se destacaram na literatura sobre regras de associação e os trabalhos realizados no instituto de informática da UFRGS para o tema regras de associação. No capítulo 3 é apresentado o software Direto, os módulos disponibilizados ao usuário e uma proposta para filtros de mensagens e canais de informação.

No capítulo 4 é apresentada uma proposta para mineração de dados sobre os filtros de mensagens e canais de informação. São apresentadas as características das mensagens utilizadas na lista de discussão SouJava e o seu pré-processamento. São descritas as três ferramentas utilizadas: Intelligent Miner, Magnus Opus e CBA. No capítulo 5 são apresentados os resultados obtidos através das ferramentas, e finalmente no capítulo 6 a conclusão.

2 Fundamentos Teóricos

Neste capítulo apresenta-se uma revisão bibliográfica sobre a mineração de dados e o processo de descoberta de conhecimento em bancos de dados, destacando-se um estudo sobre a técnica de mineração de dados conhecida como regras de associação. Inicialmente é descrito todo o processo de descoberta de conhecimento em banco de dados, onde são citadas as técnicas mais conhecidas. Posteriormente são apresentados alguns estudos sobre os tipos de regras de associação, que são o tema principal deste trabalho.

2.1 O Processo de Descoberta de Conhecimento em Bancos de Dados

Na última década, observou-se o crescimento explosivo na capacidade de geração e coleta de dados. Avanços científicos na coleta de dados, como sensores remotos, satélites espaciais, código de barras em transações comercias e armazenamento de transações de cartões de crédito, geraram uma quantidade inestimável de dados. Tais volumes não podem ser analisados de uma maneira eficaz por métodos tradicionais de pesquisa a bancos de dados, como consultas do tipo *ad-hoc* ou planilhas de cálculo. Estes métodos podem criar relatórios para os dados, mas não podem analisar seu conteúdo focando-se no conhecimento específico. Criou-se, então, a necessidade de uma nova geração de ferramentas e técnicas para analisar de forma inteligente e automática o conteúdo destes imensos bancos de dados, originando a área de mineração de dados.

A necessidade de informações vinculadas a grandes repositórios de dados fez com que a área de mineração de dados ganhasse importância frente às organizações, além de incentivar novos estudos. Segundo [BRA 2002], a mineração de dados está cada vez mais reconhecida como a chave para analizar, digerir e compreeender a inundação de dados digitais coletados através de aplicações governamentais, de negócios e científicas.

A área da descoberta de conhecimento envolve avaliação e possíveis interpretações de padrões para decisão do que é conhecimento e do que não é. Historicamente a noção de descobrir-se padrões utilizáveis em bancos de dados recebeu diversos nomes, incluindo descoberta de conhecimento em bancos de dados, mineração de dados, extração de conhecimento, descoberta de informação e processamento de padrões de dados. Assim como em [FAY 96], adota-se neste estudo descoberta de conhecimento em bancos de dados para referenciar todo o processo e mineração de dados para referenciar os algoritmos de descoberta de conhecimento.

O processo de descoberta de conhecimento é interativo e iterativo, envolvendo diversos passos e muitas decisões que podem ser feitas pelo usuário. O processo básico é descrito brevemente em [FAY 96], através de nove etapas:

a) Desenvolver e compreender o domínio da aplicação, além de conhecer o objetivo do usuário e o problema;
b) Definir um conjunto de dados ou focar-se num subconjunto de variáveis ou exemplos de dados, os quais serão utilizados para a descoberta de conhecimento;
c) Limpar e pré-processar os dados: aplicar operações básicas de remoção de ruído, coletar informações necessárias ao modelo, decidir estratégias para campos nulos e padronizá-los;

d) Reduzir e projetar os dados: descobrir características importantes para representá-los de acordo com o objetivo, além de fazer uso de transformações para reduzir o número de variáveis;
e) Escolher o método para mineração de dados mais apropriado para o estudo em questão (classificação, regressão, clusterização);
f) Escolher o algoritmo de mineração de dados, selecionando métodos para busca de padrões;
g) Minerar os dados;
h) Interpretar os padrões minerados, possivelmente retornando aos passos a-g para novas iterações;
i) Consolidar a descoberta de conhecimento, documentando-a e reportando-a aos interessados.

2.2 Técnicas de Mineração de Dados

Na prática, a mineração de dados é utilizada para predizer e descrever os dados. A predição envolve a utilização de algumas variáveis ou campos no banco de dados, para predizer valores desconhecidos ou futuros de outras variáveis de interesse. A descrição foca-se na descoberta de padrões interpretáveis, descrevendo os dados.

Com o objetivo de suprir necessidades de diferentes domínios de aplicações, um grande número de sistemas para descoberta de conhecimento foi desenvolvido. Como resultado, pode-se identificar métodos diferentes, que devem ser aplicados dependendo do interesse do usuário. De um modo geral, cada método extrai um tipo diferente de conhecimento do banco de dados, visto que cada um deles utiliza algoritmos distintos.

Como exemplos de predição serão descritos os métodos de classificação e regressão e, para descrição regras de associação e clusterização.

2.2.1 Métodos Preditivos

2.2.1.1 Classificação

O método de classificação é descrito em [FRE 98] como o mais estudado dentre os métodos de mineração de dados. Na classificação cada tupla pertence a uma classe dentre um conjunto de classes pré-definidas, sendo que cada uma delas é definida pelo usuário. As tuplas consistem de um conjunto de atributos preditivos e de um atributo objetivo. Este último atributo é categórico, isto é, possui um valor dentre um conjunto de valores discretos chamado classe ou categoria. Ele pode receber os valores discretos sim ou não, ou ainda, códigos na linha de números inteiros. Caso este atributo receba um valor contínuo, como números valorados reais, então o método é chamado de regressão.

O objetivo do método de classificação é descobrir algum tipo de relacionamento entre os atributos preditivos e o atributo categórico. Um exemplo ilustrado pelo autor é o de uma companhia internacional que publica um livro chamado "Um guia de restaurantes franceses na Inglaterra". O livro é publicado em inglês, francês e alemão de acordo com o país onde será vendido. Supondo-se que a companhia possua uma base de dados contendo referências sobre clientes nos três países nomeados Inglaterra, França e Alemanha, seria interessante utilizar estes dados para predizer qual o tipo de cliente mais se parece com os

que supostamente comprariam o novo livro. A companhia então, poderia concentrar seus esforços para vender a estes clientes, enviando-lhes, por exemplo, material de propaganda.

Para predizermos quando um cliente irá ou não comprar o livro ao receber o material de propaganda, a empresa precisa de dados sobre os efeitos desta técnica de marketing sobre alguns de seus clientes na base de dados. A partir destes dados, um algoritmo de classificação poderia descobrir regras que predizem quando um novo cliente, que não recebeu material de propaganda, irá comprar o livro, ou não. Esta informação é então armazenada em um novo atributo do banco denominado "comprar". Este atributo torna-se o objetivo e o seu valor é preditivo de valores de outros atributos do banco de dados. No caso apresentado o objetivo é um novo atributo, mas através da classificação pode-se também predizer valores para atributos existentes no banco de dados.

Uma vez determinado o objetivo, o próximo passo é selecionar um subconjunto de atributos que serão analisados para determinar o valor do atributo preditivo. Na Tabela 2.1, são apresentados os dados de dez clientes que receberam o material de propaganda e o valor do atributo preditivo “Comprar”.

TABELA 2.1 - Dados de Entrada para Classificação

Sexo	País	Idade	Comprar
Masculino	França	25	Sim
Masculino	Inglaterra	21	Sim
Feminino	França	23	Sim
Feminino	Inglaterra	34	Sim
Feminino	França	30	Não
Masculino	Alemanha	21	Não
Masculino	Alemanha	20	Não
Feminino	Alemanha	18	Não
Feminino	França	34	Não
Masculino	França	55	Não

Fonte: tradução de [FRE 98]

TABELA 2.2 - Regras de Classificação geradas a partir da TABELA 2.1

Se (País = "Alemanha") Então (Comprar = "Não")
Se (País = "Inglaterra") Então (Comprar = "Sim")
Se (País = "França" e Idade <= 25) Então (Comprar = "Sim")
Se (País = "França" e Idade > 25) Então (Comprar = "Não")

Fonte: tradução de [FRE 98]

Um algoritmo de classificação analisa os dados da Tabela 2.1 e determina quais valores tendem a ser associados ao objetivo. Na Tabela 2.2 são apresentadas as regras geradas pelo algoritmo de classificação. Estas regras são interpretadas de forma que os atributos de predição são submetidos às condições no antecedente da regra, e o seu valor é indicado no conseqüente.

2.2.1.2 Regressão

Segundo [FRE 98], o método de regressão é similar à classificação. A maior diferença entre eles é que o atributo preditivo possui um valor contínuo.

Em [FAY 96] são citados alguns exemplos de aplicação destacando-se a predição da quantidade de biomassa presente em uma floresta, com base em medidas remotamente tomadas, ou ainda, a probabilidade de morte de um paciente segundo um conjunto de diagnósticos.

2.2.2 Métodos Descritivos

2.2.2.1 Clustering

Segundo [BER 97] clustering é um método utilizado para segmentar populações heterogêneas em subgrupos mais homogêneos chamados clusters. O que distingue classificação de clustering é que nesta técnica não existem classes pré-definidas. Na classificação a população é subdividida em classes preestabelecidas, enquanto que em clustering os registros são agrupados por alguma similaridade entre as classes.

[FRE 98] classifica clustering como o método mais complexo, tendo em vista que, após a geração das classes alguns algoritmos de classificação e sumarização são aplicados para geração das regras que deverão ser validadas.

Em [FAY 96] são citados alguns exemplos de aplicação onde destacamos a descoberta de subgrupos homogêneos de clientes a fim de orientarmos o marketing, no desenvolvimento e oferta de novos produtos.

2.2.2.2 Regras de Associação

Regras de associação são padrões que representam a probabilidade de que um conjunto de itens ocorra em conseqüência de outro conjunto. Como o objetivo deste trabalho é avaliar os diferentes tipos de regras de associação aplicados a um conjunto de dados específicos, nesta seção será apresentado somente um resumo da proposta inicial.

[FRE 98] descreve brevemente o método de descoberta de regras de associação. Segundo o autor, em sua forma original este método é definido como um tipo de dados especiais, chamado de cesta de mercado, onde uma tupla consiste de um conjunto de atributos binários chamados itens. Cada tupla corresponde a uma transação de cliente onde, dado um item, este é valorado como verdadeiro ou falso, dependendo dos itens que o cliente comprou. Este tipo de dados é normalmente coletado através de tecnologia de códigos de barras, onde o exemplo típico é os produtos vendidos no supermercado.

Uma regra de associação é um relacionamento na forma $X \rightarrow Y$, onde X e Y são conjuntos de itens e $X \cap Y = \varnothing$. Para cada regra de associação são determinados dois filtros: suporte e confiança. O suporte é definido como a razão do número de registros satisfazendo X e Y sobre o número total de tuplas, isto é, $SUP = |X \cup Y| / N$, onde N é o número de tuplas e |A| representa o número de tuplas contendo todos os itens do conjunto A. A confiança é a razão entre o número de tuplas satisfazendo ambos X e Y e o número de tuplas que satisfaz X, isto é, $CONF = |X \cup Y| / |X|$.

A tarefa de descoberta de regras de associação consiste em extrair do banco de dados todas as regras com suporte e confiança maiores ou iguais a uma medida de suporte e

confiança mínimos especificados pelo usuário. A descoberta de regras de associação é desenvolvida em dois passos. Primeiramente, um algoritmo determina todos os conjuntos de itens tendo suporte e confiança maiores ou iguais aos estabelecidos pelo usuário, os quais denominamos itens freqüentes. Posteriormente, para cada item freqüente, todas as possibilidades de regras candidatas são geradas e testadas, através da confiança. Uma regra candidata é gerada tendo um subconjunto de itens dos itens freqüentes como antecedente da regra, e os demais itens do conjunto de itens freqüentes como conseqüentes da regra. Somente regras candidatas tendo confiança maior ou igual à especificada pelo usuário são apresentadas pelo algoritmo.

Para ilustrar a descoberta de regras de associação, considerar o exemplo simples da cesta de mercado apresentado na Tabela 2.3. A primeira coluna na tabela apresenta o identificador da transação, e as outras colunas indicam quais itens foram comprados em cada transação. No exemplo ilustrativo os itens são apresentados num alto nível de abstração, em análises mais aprofundadas, normalmente considerarmos os itens num baixo nível de abstração incluindo subcategorias como a marca do produto.

TABELA 2.3 - Dados de Entrada para Descoberta de Regras de Associação

Tr_Id	Leite	Café	Cerveja	Pão	Manteiga	Arroz	Feijão
1	Não	Sim	Não	Sim	Sim	Não	Não
2	Sim	Não	Sim	Sim	Sim	Não	Não
3	Não	Sim	Não	Sim	Sim	Não	Não
4	Sim	Sim	Não	Sim	Sim	Não	Não
5	Não	Não	Sim	Não	Não	Não	Não
6	Não	Não	Não	Não	Sim	Não	Não
7	Não	Não	Não	Sim	Não	Não	Não
8	Não	Não	Não	Não	Não	Não	Sim
9	Não	Não	Não	Não	Não	Sim	Sim
10	Não	Não	Não	Não	Não	Sim	Não

Fonte: tradução de [FRE 98]

Supondo que o usuário tenha especificado como parâmetros *SUP = 0.3* e *CONF = 0.8*, a Tabela 2.4 apresenta as regras de associação que deveriam ser descobertas sobre os dados da Tabela 2.3, de acordo com os valores especificados para suporte e confiança. Na Tabela 2.4 as regras de associação estão agrupadas por itens freqüentes conforme foram geradas. Nela, apresentam-se somente itens freqüentes com dois ou mais itens. Itens freqüentes com apenas um item (por exemplo, café) não são apresentados, pois produzem regras desinteressantes, onde o antecedente ou conseqüente estariam vazios. A Tabela 2.4 apresenta o suporte de cada item freqüente e a confiança de cada regra descoberta.

TABELA 2.4 - Regras de Associação(Suporte= 0.3, Confidencia=0.8)

Itens Freqüentes: Café, Pão. Suporte=0.3
Regra: Se (Café) Então (Pão). Confiança =1.
Itens Freqüentes: Café, Manteiga. Suporte=0.3
Regra: Se (Café) Então (Manteiga). Confiança =1.
Itens Freqüentes: Pão, Manteiga. Suporte=0.4
Regra: Se (Pão) Então (Manteiga). Confiança = 0.8.
Regra: Se (Manteiga) Então (Pão). Confiança = 0.8.
Itens Freqüentes: Café, Pão, Manteiga. Suporte=0.3
Regra: Se (Café e Pão) Então (Manteiga). Confiança = 1.
Regra: Se (Café e Manteiga) Então (Pão). Confiança = 1.
Regra: Se (Café) Então (Pão e Manteiga). Confiança = 1.

Fonte: tradução de [FRE 98]

Recentemente as regras de associação têm sido estendidas a outras aplicações e vários estudos surgiram aprimorando a proposta inicial de Agrawal.

2.3 Regras de Associação

Como um dos objetivos deste trabalho é fazer um comparativo de técnicas de mineração de dados através de regras de associação utilizando ferramentas que implementam variações da proposta inicial, nesta seção são apresentados estudos encontrados na literatura sobre variações para o tema mineração de dados através de regras de associação.

As regras de associação são um problema que atraem um interesse considerável, pois oferecem padrões concisos de informações utilizáveis que as tornam facilmente entendidas pelo usuário final. Na literatura de banco de dados, o foco é desenvolver algoritmos de regras de associação que identifiquem todas as regras encontradas, através de especificações de usuários, para filtros de suporte e confiança mínimos.

2.3.1 Mineração em Bancos de Dados Muito Densos

Nesta seção é apresentada a proposta desenvolvida por [AGR 99], em cujo trabalho descreve um algoritmo para minerar regras de associação em bancos de dados muito densos. Ele utiliza um algoritmo que explora diretamente as *constraints* tradicionais da literatura de regras de associação (suporte e confiança mínimos), e propõe um novo filtro, ou *constraint*, denominado *Improvment*. Segundo o autor, este filtro assegura uma predição mais avançada sobre as então ditas simplificações da proposta inicial descritas em [AGR 93, AGR 94]. O autor afirma ainda que, neste algoritmo a eficiência é mantida até mesmo para suportes baixos, onde os dados são muito densos.

O algoritmo de regras de associação foi inicialmente desenvolvido para análise de dados de cesta de mercado, porém, recentemente houve interesse de aplicações em áreas como telecomunicações e censo demográfico. Diferente da análise de cesta de mercado, estes conjuntos de dados tendem a ser ainda mais densos e possuem as propriedades abaixo citadas:

- Presença de itens que aparecem com muita freqüência nas transações (ex. sexo masculino);
- Fortes correlações entre os itens;
- Muitos itens em cada registro.

Os conjuntos de dados que possuem as características acima mencionadas podem causar um estouro exponencial no consumo de recursos, se citarmos o algoritmo Apriori e suas variações. Os itens mais freqüentes e os que possuem fortes correlações aparecem freqüentemente associados à maioria das regras encontradas. Aplicando-se somente filtros de suporte e confiança mínimos durante o processamento, um número exorbitante de regras poderia ser gerado. Embora outros filtros pudessem ser empregados para filtrar as regras mais interessantes, eles estariam sendo empregados apenas na etapa de pós-processamento.

[AGR 99] considera que a atual proposta explora mínimos durante a mineração com muita eficiência. Além disso, as regras que satisfazem estas *constraints* em um banco de dados muito denso apresentam um resultado facilmente compreendido pelo usuário final. Para remediar o problema da quantidade de regras geradas, o algoritmo explora uma *constraint* que elimina as regras que não são interessantes, pois segundo ele, tais regras não possuem condições que contribuam para predição. Para ilustrar o conceito, seguem os exemplos:

Pão e Manteiga → Leite (Confiança 80%)

A regra tem confiança de 80%, o que significa dizer que 80% das pessoas que compram pão e manteiga, também compram o item conseqüente da regra, que é leite. Em conseqüência da alta confiança, poder-se-ia acreditar que esta regra é uma descoberta interessante, caso o objetivo fosse compreender a população apreciadora de leite. Entretanto, se for considerado que 85% da população examinada compra leite, esta regra é na realidade, desinteressante para o propósito de caracterizar a população que regularmente consome o produto, já que está abaixo da média vendida. Existe uma medida denominada *lift* capaz de eliminar este tipo de regra. Esta medida considera a proporção entre a confiança da regra e a de seu conseqüente. Neste caso o *Lift* da regra seria negativo, pois 0.80/0.85 <1 e como conseqüência à regra seria eliminada do resultado da pesquisa.

Ovos e Cereais → Leite (Confiança = 95%)
Ovos → Leite (Confiança =80%)
Cereais → Leite (Confiança = 99%)

Como conseqüência da alta confiança da regra (95%) e da freqüência com o qual, o leite é comprado (85%), esta regra pode parecer interessante. Entretanto, se considerarmos que cereais comprados sozinhos implicariam a compra de leite com 99% de confiança, temos então, na verdade, uma regra que representa um significante decréscimo na habilidade preditiva, visto que existe uma regra mais concisa e aplicável.

Para endereçar este problema, o algoritmo proposto em [AGR 99] permite que o usuário especifique um *Improvment* mínimo. A ideia é minerar somente regras nas quais a confiança é pelo menos *minimp* maior que a confiança de qualquer sub-regra conveniente onde, uma sub-regra conveniente é uma simplificação da regra, formada pela remoção de

uma ou mais condições do seu antecedente. Qualquer conjunto positivo ao *minimp* irá prevenir o resultado de regras desinteressantes como os exemplos acima. Dado o estado da regra em que cereal implica leite com 99% de confiança, pode haver centenas de regras na forma abaixo com confiança entre 99% e 99,1%.

Cereal & Item1 & Item2 & Item3...&ItemN → Leite

Para que as regras possam ser comparadas como descrito acima, é preciso que estas tenham conseqüentes equivalentes.

Uma transação é um conjunto de um ou mais itens obtidos a partir de um domínio finito de itens e um *data-set* é uma coleção de transações. Um conjunto de itens será referenciado mais sucintamente como um *itemset*. O suporte de um *itemset I*, denotado *sup(I)*, é o número de transações no *data-set* que contém *I*. Uma regra de associação consiste de um *itemset* chamado antecedente, e de um *itemset* desconexo do antecedente chamado conseqüente. Uma regra será apresentada na forma $A \rightarrow C$ onde *A* é o antecedente e *C* o conseqüente. O suporte da regra de associação é o suporte do *itemset* formado pela união do antecedente e conseqüente *(A U C)*. A confiança da regra é a probabilidade com a qual o item no antecedente *A* apareça junto aos itens do conseqüente *C* num dado *data-set*.

$$Conf(A \rightarrow C) = \frac{Sup(A \cup C)}{Sup(A)}$$

A introdução da *constraint* de *Improvment* por [AGR 99] tem como objetivo minimizar o problema das milhares de regras retornadas por algoritmos tradicionais, quando submetidos a banco de dados densos. O *Improvment* pode ser definido como a diferença mínima entre a confiança da regra em questão e a confiança de qualquer sub-regra gerada a partir da regra proposta, sendo que ambas, devem possuir o mesmo conseqüente.

Formalmente, dada uma regra $A \rightarrow C$:

$Imp\ (A \rightarrow C) = Min(\forall A' \subset A, conf(A \rightarrow C) - conf(A' \rightarrow C))$

Se o *Improvement* de uma regra for positivo, então removendo qualquer combinação não vazia de itens de seu antecedente sua confiança é pelo menos igual ao *Improvement*. Assim, cada item e cada combinação de itens presentes no antecedente de uma regra com *Improvment* maior do que o estabelecido, torna-se um importante contribuidor para predição de regras aceitas. Uma regra com *Improvement* negativo é desinteressante, pois a própria regra pode ser simplificada para produzir uma sub-regra que é mais preditiva. Um *Improvement* maior do que zero é uma *constraint* desejável, na maioria das aplicações de mineração por regras de associação.

2.3.1.1 Formalização do Problema

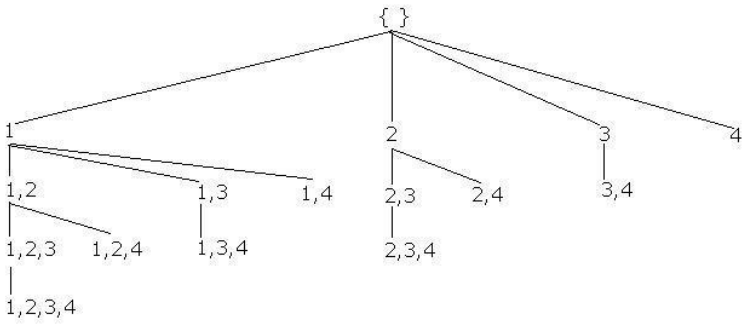
O algoritmo proposto foi desenvolvido para minerar todas as regras de associação com um conseqüente de acordo com mínimos para suporte, confiança e *Improvment*

especificados pelo usuário. Os parâmetros do algoritmo que especificam a confiança mínima, suporte mínimo e *Improvment* mínimo são conhecidos com *mininconf*, *minsup e minimp*, respectivamente.

Uma regra pode ser dita com grande confiança quando sua confiança for pelo menos igual ao *minimp*, e freqüente se o suporte for pelo menos igual ao *minsup*. Uma regra é dita com grande *Improvment* quando este é pelo menos igual ao *minimp*.

Nesta seção uma regra será representada utilizando-se somente seu antecedente desde que o conseqüente seja assumido como fixo no *itemset C*. Representamos *U* como o conjunto de todos os itens presentes no banco de dados exceto aqueles que estiverem no conseqüente *C*. O problema de mineração é então buscar através de *U*, regras que satisfazem o suporte, confiança e *Improvment* mínimos. Uma estrutura de árvore é utilizada para representar o problema de busca de subconjunto, tornando-o um problema de busca em árvore e permitindo que regras possam ser cortadas como se estivéssemos podando uma árvore. Além disso, a estrutura permite que o espaço dos subconjuntos (regras) possa ser reduzido. A idéia é primeiramente impor uma ordenação no conjunto de itens e então enumerá-los de acordo com a Figura 2.1.

FIGURA 2.1 - Árvore sobre U={1,2,3,4}, ordenada lexicamente



[AGR 99]

Como dito anteriormente, estruturou-se o problema de minerar máximos *itemset* freqüentes do banco de dados como um problema de busca de conjuntos enumerados em árvore. Cada nodo na árvore é representado por dois *itemset* chamados grupos. O primeiro *itemset*, chamado *head*, é simplesmente o *itemset* (regra) enumerado. O segundo chamado *tail* é um conjunto ordenado, e consiste dos itens os quais podem ser juntados a *head* para formarem regras viáveis enumeradas por um sub-nodo. Por exemplo, na raiz da árvore, o conjunto de itens *head* é um conjunto vazio, enquanto o conjunto do *tail* consiste de todos os itens em *U*.

O *head* e *tail* de um grupo *g* serão representados por *h(g)* e *t(g)* respectivamente. A ordem na qual os itens do *tail* aparecem, possui uma significância, pois refletem como seus nodos filhos estão dispostos para serem expandidos. Cada nodo filho g_c do grupo será expandido, tomando-se um item $i \in t(g)$ e apendando a *h(g)* para formar $h(g_c)$. Os nodos são ordenados e mantendo-se este policiamento para expansão dos nodos filhos, ao considerar que não houve nenhum corte, a árvore enumera todo e qualquer subconjunto de *U*, exatamente uma vez.

2.3.1.2 Descrição de Alto Nível do Algoritmo

Nesta seção é descrito o algoritmo *Dense-Miner* [AGR 99] em alto nível. O corpo do algoritmo apresentado abaixo implementa a busca de abertura do conjunto da árvore gerando os grupos iniciais.

DENSE-MINER (Conjunto de Transações T)
;; Retorna os itens que satisfazem os filtros de suporte,
;; confiança e improvment
Conjunto de Regras ← ϕ
Conjunto de Grupos G ← GENERATE-INITIAL-GROUPS (T, R)
While G não-vazio **do**
    Examina T para processar todos os grupos em G
    PRUNE-GROUPS(G, R)
    G ← GENERATE-NEXT-LEVEL(G)
    R ← R ∪ EXTRACT-RULES(G)
    PRUNE-GROUPS(G, R)
Return POST-PROCESS(R, T)

Os grupos gerados representam um nível inteiro da árvore os quais são um passe sobre o conjunto de dados. A geração dos grupos iniciais poderia simplesmente produzir o nodo raiz, que é vazio. Entretanto, a implementação produz a busca no segundo nível da árvore depois da fase de otimização que rapidamente computa o suporte de todos itens 1 e 2.

A geração do próximo nível utiliza o algoritmo descrito abaixo e gera os grupos que compreendem o próximo nível da árvore.

GENERATE-NEXT-LEVEL (Conjunto de Grupos G)
;; Retorna um conjunto de grupos representando o próximo nível da árvore
Conjunto de Regras ← ϕ
Conjunto de Grupos G ← GENERATE-INITIAL-GROUPS(T, R)
For each grupo g em G_c **do**
    Reordenar os itens em t(g)
    For each item i em t(g) **do**
        Considerar g_c sendo um novo grupo
            com $h(g_c) = h(g_c) \cup \{i\}$ e
            $t(g_c) = \{j/j$ seguido de i ordenadamente$\}$
        $G_c \leftarrow G_c \cup \{g_c\}$
Return G_c

Pelo algoritmo percebe-se que os itens do *tail* de um grupo são reordenados depois dos nodos-filhos terem sido expandidos. Este passo de reorganização é uma otimização desenvolvida para maximizar a eficiência do corte. Após expandir os nodos filhos, qualquer regra representada pelo *head* de um grupo é colocada em *R* pela função *Extract-rules* se for freqüente e possuir confiança igual ou superior a estabelecida. A informação de suporte

necessária para verificar se o *head* do grupo *g* representa uma regra freqüente e/ou com a confiança igual ou superior ao limiar é fornecida pelo nodo pai de g no conjunto enumerado da árvore, pois *h(g) e h(g) U C* são membros de seus conjuntos candidatos. Como resultado, este passo pode ser executado após *g* ter sido processado.

A função *Prune-Groups* é utilizada pelo *Dense-Miner* para cortar ambos os grupos: processados e não processados. No primeiro algoritmo descrito, os grupo são cortados sobre a expansão da árvore bem como imediatamente após seu processamento. Como grupos não são processados seguindo a expansão da árvore, com o objetivo de determinar se serão cortados, *Dense-Miner* utiliza informações de suporte apanhadas em passes prévios sobre o banco de dados.

```
PRUNE-GROUPS (Conjunto de Grupos G, Conjunto de Regras R)
;; Corta grupos e tails dos grupos de G
;; G e R são passados por referência
For each grupo g em G
   do
     try_again ← falso
     if IS-PRUNENABLE(g)
       then remover g de G
       else for each i ∈ t(g) do
                  considerar g' um grupo
                     com h(g') = h(g) ∪ {i}
                     e t(g') = t(g) - {i}
                  if IS-PRUNABLE (g')
                    then remover i de t(g)
                         incluir h(g) ∪ {i} em R se
                         a regra for freqüente e possuir confiança superior ao limiar
                         try-again ← verdadeiro
  while try-again = verdadeiro
```

Dense-Miner aplica múltiplas estratégias para cortar nodos da árvore. Estas estratégias determinam quando um grupo *g* pode ser ou não cortado. A função de corte verifica se ao invés de cortar grupos, apenas alguns itens *i* em *t(g)*, deveriam ser cortados. Cortando-se *tails* reduz-se o número de filhos gerados do nodo, e desse modo reduz-se o espaço de busca. Um benefício adicional de cortar-se os *tails* é o incremento na eficiência da estratégia usada para cortar o grupo.

A importância deste fato é que dado um grupo *g* e item *i* de *t(g)* como condição, evita-se enumerar muitas regras as quais não satisfazem a *constraint* pela simples remoção de *i* de *t(g)*. A implementação de grupos de corte está descrita através do algoritmo *Prune-Groups*, acima apresentado. A estratégia de corte de grupo é aplicada com o auxilio de uma função chamada *Is-Prunable* que é descrita abaixo.

A função *Is-Prunable* computa os seguintes valores para um dado grupo g:

- Um valor superior ao limite da confiança de qualquer regra derivada de *g*, chamado *uconf(g)*;

- Um valor superior ao limite do *improvment* de qualquer regra derivada de *g*, chamado *uimp(g)*;
- Um valor superior ao suporte de qualquer regra derivada de *g*, chamado *usup(g)*.

Um grupo *g* pode ser cortado sem afetar a busca, se algum valor superior falhar de acordo com o mínimo estabelecido pelo usuário para *minconf*, *minimp*, *minsup* respectivamente. A dificuldade para implementar o corte não é somente em termos de como computar os valores, mas deve-se levar em consideração o tempo de processamento, considerando-se um grande banco de dados. Os valores são computados, utilizando-se somente o suporte fornecido pelo conjunto candidato, isto é, os nodos superiores.

O objetivo de reordenar os itens durante a função de *General_Next-Level*, utilizada pelo *Dense-Miner*, é não forçar regras não promissoras para a mesma porção da árvore de busca. A razão desta estratégia é importante, pois ao ordenar um grupo para ser podado, todos os seus sub-nodos devem representar uma regra que falha, ao ser testada através das *constraints* estabelecidas.
Diversos experimentos foram conduzidos pelo autor e este concluiu que a melhor política de ordenamento que poderia ser utilizada pelo *Dense-Miner* seria ordenar em ordem decrescente de $sup(h(g) \cup \{\neg i, \neg c\})$, onde $\neg c$, que contêm somente as transações que não contém o item conseqüente *C* e $\neg i$, que contêm apenas as transações que não contém *i*.

2.3.1.3 Pós-processamento

O pós-processamento identifica algumas regras que não têm um *Improvment* superior ao limiar estabelecido, pela simples comparação delas com as outras regras no conjunto de regras mineradas *R*. Ele compara cada regra $r1 \in R$ para cada regra *r2* tal que $r2 \in R$ e $r2 \subset r1$. Se não for encontrada *conf* (r1) - *conf* (r2) < *minimp*, então a regra *r1* é removida, pois seu *Improvment* é pequeno demais.

O passo de pós-processamento não necessita acesso ao banco de dados, e remove quase todas as regras que não possuem um *Improvment* suficiente grande. Além disso, inclui uma extensão sobre a ordenação de medidas para serem apresentadas ao usuário, incluindo a sub-regra correta responsável pelo *Improvment* de uma regra.

2.3.2 Regras de Associação Taxonômicas

Na proposta tradicional de regras de associação, um exemplo típico é o fato de que 80% dos clientes que compram leite, também compram pão. Como exemplo para mineração em diferentes níveis de abstração, cita-se o fato de que 75% dos clientes que compram pão de trigo compram leite de uma determinada marca. É interessante apresentar ao usuário a expansão da informação, ou ainda agrupá-la num nível menos detalhado para que esta possa estar presente nos resultados da pesquisa.

[SRI 95] introduziu o problema de minerar regras de associação, onde os itens pudessem ser generalizados através de uma taxonomia. A idéia é a de que itens com baixo suporte pudessem estar presentes no resultado da mineração, caso estes pudessem ser apresentados de uma maneira menos detalhada. Trabalhos anteriores a este, envolvendo regras de associação, não faziam referência a taxonomias e restringiam a construção de

regras apenas nos nodos-folha da hierarquia. A proposta do autor é de descobrir regras de associação onde os itens pudessem pertencer a qualquer nível da hierarquia.

Segundo [HAN 97], para explorar mineração em múltiplos níveis é necessário que os dados sejam dispostos em diferentes níveis de abstração, além de aplicar métodos eficientes para este propósito. A primeira necessidade pode ser satisfeita contanto que o conceito de taxonomia seja aplicado desde os níveis mais primitivos até os mais altos. Com o recente desenvolvimento do *datewarehousing* e tecnologia OLAP (*Analytic Processing Systems*), distribuir os dados em níveis de abstração tem se tornado uma prática comum.

A segunda necessidade pode ser explorada de diversas maneiras. Uma alternativa é aplicar diretamente métodos existentes para níveis únicos a múltiplos níveis. Um exemplo é a aplicação do algoritmo Apriori para examinar itens de dados em múltiplos níveis de abstração sobre os mesmos limites de suporte e confiança mínimos. Esta estratégia é simples, mas ocasiona alguns resultados desinteressantes. Primeiramente, é encontrado suporte superior ao determinado, nos mais altos níveis de abstração. Caso fosse necessário descobrir fortes associações nos menores níveis, o suporte mínimo deveria ser reduzido e como conseqüência seriam geradas regras desinteressantes nos níveis intermediários e superiores. Segundo [HAN 97], desde que seja improvável descobrir muitas regras de associação com suporte esperado no nível primitivo de conceitos, mineração através de associações pode ser desenvolvida antes num alto nível, o que é na verdade o objeto de estudo de diversos trabalhos. Entretanto, a mineração de regras de associação pode freqüentemente gerar regras correspondentes a expectativas e conhecimento prévio, como é o caso de leite→ pão, que é senso comum, ou ainda, gerar combinações desinteressantes, se o suporte mínimo for extremamente baixo, como é o caso de brinquedos→ leite.

As observações acima descritas nos levam a examinar outra proposta: aplicar limites de suporte mínimo diferente, e possivelmente diferentes limites para confiança mínima, nos diferentes níveis de abstração. Se explorarmos o caso de reduzir progressivamente os limites de suporte mínimo nos níveis mais baixos de abstração, encontraríamos regras de associação interessantes nos múltiplos níveis de conceito. Neste caso seriam descobertas não somente regras de associação nos diferentes níveis, mas regras de associação não triviais, informativas, pois há flexibilidade para diferentes conjuntos de dados.

Neste estudo um método é desenvolvido pela extensão do Apriori para mineração de regras de associação. O método busca primeiramente itens freqüentes nos níveis mais altos e então progressivamente aprofunda o processo de mineração aos níveis mais baixos dos descendentes dos itens freqüentes.

Um fato importante que deve ser assumido neste estudo é a exploração de somente descendentes dos itens freqüentes, pois se considerarmos que um item ocorre raramente, então seus descendentes irão ocorrer mais raramente ainda.

O autor relata que a necessidade de mineração de regras de associação em múltiplos níveis ou utilização de informações de taxonomias foi observada em outros trabalhos. O diferencial entre os outros estudos e o [HAN 97], é que utilizam o mesmo limite de suporte através dos diferentes níveis de abstração. Utilizando-se o mesmo suporte para diferentes níveis é preciso muito esforço para identificar e remover as regras redundantes.

2.3.2.1 Definição Formal do Problema

Assumimos que o banco de dados contenha:

1) Um conjunto de itens *I*, que contém a descrição de cada item em *I* na forma < A_i, descrição>, onde $A_i \in I$;
2) Um conjunto de dados de transações, *T*, as quais consistem de um conjunto de transações $<Ti,\{Ap,....,Aq\}>$, onde T_i é uma transação identificável por $A_i \in I$ (para $i = p,...q$).

Um padrão, ou um *itemset*, é um item A_i, ou um conjunto de itens $A_i \wedge .... \wedge A_j \in I$. O suporte de um padrão *A* em um conjunto *S*, $\sigma (A/S)$, é o número de transações em *S* as quais contém *A* versus o total de número de transações em *S*. A confiança de $A \rightarrow B$ em *S*, $\varphi (A \rightarrow B/S)$, é a proporção de $\varphi (A \wedge B/S)$ versus $\varphi (A/S)$, isto é, a probabilidade que o padrão *B* ocorra em *S* quando o padrão *A* ocorre.

Para descobrirmos padrões freqüentes e razoáveis implicações de regras fortes, o usuário precisa especificar dois limites: suporte mínimo e confiança mínima. Para a descoberta de regras de associação de múltiplos níveis, diferentes valores para o suporte mínimo e confiança devem ser especificados nos diferentes níveis.

Um padrão *A* é freqüente em um conjunto *S* no nível *l* se o suporte de *A* não for menor que o suporte mínimo correspondente. Uma regra $A \rightarrow B/S$ é dita forte se, para um conjunto *S*, cada ancestral (i.e., o nível superior correspondente) de cada item em *A* e *B* for freqüente, e a confiança de $A \rightarrow B/S$ for maior que o limite mínimo de confiança no nível corrente.

A definição acima implica um processo de filtragem, o qual limita o padrão para examinar o menor nível. Somente os itens que possuem suporte maior ao especificado no seu nível superior são examinados. Baseados nesta definição, a idéia de minerar múltiplos níveis de associação é ilustrada abaixo.

Exemplo: A consulta consiste em descobrirmos fortes associações em múltiplos níveis no banco de dados da Tabela 2.5, relacionados à categoria, conteúdo e marca de alimentos, os quais podem ser armazenados por menos de três semanas.

A parte relevante da descrição dos itens vendidos está relacionada na Tabela 2.6. Na Tabela 2.7, cada registro representa a generalização de itens os quais possuem mesmo valor para os atributos de interesse.

TABELA 2.5 - Tabela de Transações de Compras

Transação	Conjunto de Código de Barras
351428	{17325, 92108, 55349, 88157,....}
....	{....,....}

Fonte: tradução de [HAN97]

TABELA 2.6 - Descrição de Itens

Cód Barra	Categoria	Marca	Conteúdo	Tamanho	Dias Armazenados	Preço
17325	Leite	Foremost	2%	1(ga.)	14 dias	$3,89
...	...	....	...	...	...	...

Fonte: tradução de [HAN97]

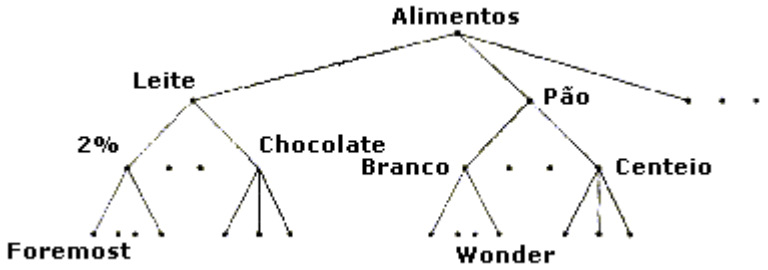
TABELA 2.7 - Itens de Venda Generalizados

GID	Conjunto de Código de Barras	Categoria	Conteúdo	Marca
112	{17325, 31414, 91265}	Leite	2%	Foremost
...	...	...	...	...

Fonte: tradução de [HAN97]

A informação de taxonomia é provida implicitamente na Tabela 2.7. A categoria leite representa o primeiro nível de conceito, o conteúdo 2%, é o segundo nível, e a marca o terceiro. A tabela implica uma árvore como a apresentada na Figura 2.2.

FIGURA 2.2 - Taxonomia dos Itens Relevantes



[HAN 97]

O processo primeiramente descobre padrões freqüentes e fortes para regras de associação no mais alto nível de conceito. Considerando o suporte mínimo para este nível igual a 5% e a confiança mínima igual a 50%, uma descoberta seria: um conjunto de itens freqüentes (1-*itemset*, com suporte entre parênteses): {pão (25%), vegetais (30%)...}, um conjunto de combinações em pares de itens freqüentes (2-*itemset*): {vegetais, pão (19%)),...}, e um conjunto de fortes regras de associação, tais como pão→ vegetais (76%).

No segundo nível, considerando suporte mínimo igual a 2% e confiança mínima igual a 40%, uma descoberta seria: 1-*itemset* freqüentes: "alface (10%), pão de trigo (15%)..." e 2-*itemset* freqüentes:"<alface, pão de trigo (6%)>",..., e um conjunto de regras de associação fortes: "alface→ pão de trigo (60%)". Os níveis serão expandidos até que houver itens com suporte e confiança suficientes para gerar um nodo filho.

2.3.2.2 O Algoritmo

Nesta seção é apresentada mineração de regras de associação em múltiplos níveis, utilizando-se como exemplo a informação de hierarquia representada na Tabela 2.8. Uma consulta de mineração de dados releva somente algumas porções das transações no banco de dados, motivo pelo qual nem todos os itens estão sendo representados.

TABELA 2.8 - Tabela Decodificada T1

TID	Items
T1	{111, 121, 211, 221}
T2	{111, 211, 222, 323 }
T3	{112, 122, 221, 411}
T4	{111, 121}
T5	{111, 122, 211, 221, 411}
T6	{211, 323, 524}
T7	{323, 411, 524, 713}

Fonte: tradução de [HAN97]

A codificação é feita de forma que a informação de taxonomia para cada item no exemplo seja decodificada como uma seqüência de dígitos na tabela de transações representadas através da Tabela 2.8. Por exemplo, o item "2% de leite *Foremost*" é codificado como '112' no qual o primeiro dígito 1 representa leite no primeiro nível-1 , o segundo '1' representa "2%(leite)", no nível-2 e o terceiro dígito representa a marca "*Foremost*" no nível-3.

A derivação do conjunto de itens freqüente no nível 1 é apresentada a seguir: Considerar o suporte mínimo igual a quatro transações *(i.e., minsup[1]=4)*[1]. O nível 1 freqüente na tabela 1-*itemset* L[1,1] pode ser derivado pela busca de T[1], registrando o suporte para cada item generalizado. Um exemplo de representação é 1**,...., 4**, caso estes itens estejam contidos na transação. O filtro de suporte acumulado é utilizado para filtrar:

1) Qualquer item que não é freqüente na transação;
2) A transação em T[1] que contem somente itens freqüentes.

O resultado das transações filtradas é apresentado através da Figura 2.3. Caso haja apenas duas entradas em L[1,1], o nível-1 freqüente 2-*itemset* na tabela L[1,2] poderá conter apenas um item candidato{1**,2**}, que é suportado por quatro transações na tabela T[2].

FIGURA 2.3 - Tabela de Transações T[2]

Nível –1 MinSup = 4
Nível –1 1-Itemset- freqüentes: L[1,1]

Itemset	Suporte
{1**}	5
{2**}	5

Nível –1 2-itemset freqüentes: L[1,2]

Itemset	Suporte
{1**, 2**}	4

Tabela Transações Filtradas T[2]

TID	Itens
T1	{111, 121, 211, 221}
T2	{111, 211, 222}
T3	{112, 122, 221}
T4	{111, 121}
T5	{111, 122, 211, 221}

[HAN97]

De acordo com a definição de regras de associação taxonômicas, somente os descendentes de itens freqüentes no nível-1 (i.e., L[1,1]) são considerados como candidatos

ao nível-2 freqüentes 1-*itemset*. Considerando-se *minsup*[2]=3, o nível-2 freqüente 1-*itemset* l[2,1] pode ser derivado da tabela de transações filtradas T[2], através da acumulação do contador de suporte e remoção dos itens nos quais o suporte é inferior ao estabelecido, o que resultará em L[2,1]. O *itemset*-3 freqüente da tabela L[2,3] é formado pela combinação das entradas L[2,2].

Finalmente, L[3,1] e L[3,2] no nível 3 são computados num processo similar, com o resultado apresentado na Figura 2.4. A computação termina quando não existe nenhum nível mais profundo necessário a consulta, ou quando um 1-*itemset* freqüente vazio é gerado em qualquer nível.

FIGURA 2.4 - Itens freqüentes para os níveis 2 e 3

Nível 2 Minsup=3
Nivel 2 frequentes 1-itemset:L[2,1]

Itemset	Suporte
{11*}	5
{12*}	4
{21*}	4
{22*}	4

Nível 2 frequentes 2-itemset: L[2,2]

Itemset	Suporte
{11*, 12*}	4
{11*, 21*}	3
{11*, 22*}	4
{12*, 22*}	3
{21*, 22*}	3

Nível 2 freqüente 3-itemset: L[2,3]

Itemset	Suporte
{11*, 12*, 22*}	3
{11*, 21*, 22*}	3

Nível 3 Minsup=3
Nível 3 frequentes 1-itemset L[3,1]

Itemset	Suporte
{111}	4
{211}	4
{221}	3

Nivel 3 frequentes 2-itemset L[3,2]

Itemset	Suporte
{111, 222}	3

[HAN 97]

2.3.2.3 Algoritmo ML_T2L1

Nesta seção apresenta-se um dos algoritmos mencionados por [HAN 97] chamado ML_T2L1, cujo objetivo é descobrir conjuntos de itens freqüentes, através dos múltiplos níveis e minerar fortes regras de associação.

Ele inicia no nível 1, derivando para cada nível l, o conjunto k-items freqüentes, $\mathcal{L}$ [l, k], para cada k, e conjunto de itens freqüentes, $\mathcal{L}\mathcal{L}$ [l] (para todos k's) como a seguir:

```
for  (l := 1; ℒ[1,1] <> ∅ and l < max_level; l ++) do  {
  if l = 1 then {
     ℒ[1,1] : = get_large_l_itemset (τ[1]; Λ);
     τ[2] : = get_filtered_table ( τ[1], ℒ[1]);
  }
```

```
  else ℒ[1,1] : = get_large_l_itemsets  ( τ [2], l );
  for (k: = 2;  ℒ [l, k -1] <> ∅; k ++ ) do {
       Ck : = get_candidate_set (ℒ [1,k -1] );
       foreach transaction t ∈ τ [2] do {
          Ct : = get_subsets (Ck, t);
          foreach candidate c ∈ Ct do c.support ++;
       }
      ℒ [1,k] : = { c ∈ Ck | c.support >= minsup [l] }
  }
ℒℒ [1] : = Uk ℒ [1,k];
}
```

Entrada para o Algoritmo:

1) T[1], que é a informação hierárquica do banco de dados, no formato de *<TID, Itemset>*.
2) Informação de suporte mínimo (*minsup[l]*) para cada nível conceitual *l*.

Saída do Algoritmo: Um conjunto de *Itemset* freqüentes para os múltiplos níveis.

Após terem sido descobertos os itens freqüentes, o conjunto de regras de associação para cada nível pode ser derivada através do conjunto de itens freqüentes baseados na confiança mínima aplicada ao nível.

Método: Um processo top-down progressivo para a profundidade, o qual coleta itens freqüentes em diferentes níveis de conceito similar ao exemplo acima apresentado.

2.3.2.4 Pós-processamento

Nem todas as regras que passam pelos filtros de suporte mínimo e confiança mínima são interessantes para serem apresentados ao usuário. Duas medidas interessantes foram propostas para filtrar regras redundantes e desnecessárias.

Uma regra é considerada redundante caso possa ser derivada ou computada num nível mais alto. Por exemplo, supondo-se que exista uma regra leite→ pão (suporte 12% e confiança 85%) e outra regra leite com chocolate → pão (suporte 1% e confiança 84%), pode não ser interessante dependendo da percentagem de leite com chocolate frente ao item leite.

Se possuirmos uma regra "80% das pessoas que compram pão, compram leite" e outra regra "80% das pessoas que compram pão e manteiga, compram leite", a primeira regra deve ser desconsiderada, tendo em vista que a segunda fornece um nível mais detalhado de informações.

2.3.3 Mineração de Regras de Associação Quantitativas

A mineração de regras de associação pode ser vista como a descoberta de associações entre valores "1", numa tabela onde todos os atributos são lógicos. Esta tabela

possui um atributo correspondente para cada item e registro de cada transação. O valor para um dado atributo é "1", caso este esteja presente na transação e "0" caso contrário. É preciso verificar os casos onde o problema de regras de associação pode ser mapeado para o problema de regras de associação lógicas. O mapeamento é simples no caso de todos os atributos serem categóricos ou quantitativos. Conceitualmente, ao invés de existir apenas um campo na tabela para cada tipo de atributo (Idade, Número de Carros), é preciso criar muitos campos, de acordo com os atributos lógicos. O valor para um campo lógico corresponde <atributo1, campo1>, onde possui valor 1 se o atributo1 possui o valor 1 no registro original, e 0 caso contrário. Se o domínio de valores for grande e quantitativo, estes podem ser particionados em intervalos e cada um mapeado na forma <atributo, intervalo>. Podemos então utilizar qualquer algoritmo para descobrirmos regras de associação quantitativas.

Tabelas relacionais, na maioria dos negócios e domínio científico possuem vários tipos de atributos. Em [SRI 96], os atributos são definidos como quantitativos (idade, rendimento) ou categóricos (marca de carro) e os atributos lógicos são considerados um caso especial de atributos categóricos.

Na Tabela 2.10 apresenta-se o mapeamento dos atributos da tabela pessoa apresentada na Tabela 2.9. A idade é particionada em dois intervalos 20. 29 e 30..39. O atributo categórico estado civil possui dois atributos lógicos: "Casado: Sim e Casado: Não". Como o número de valores para número de carros é pequeno, então este não é particionado em intervalos e cada valor é mapeado para um campo lógico.

TABELA 2.9 - Pessoas

Id	Idade	Casado	NumCarros
100	23	Não	1
200	25	Sim	1
300	29	Não	0
400	34	Sim	2
500	38	Sim	2

Fonte[SRI 96]

TABELA 2.10 - Mapeamento Lógico de Atributos

Id	Idade: 20.. 29	Idade: 30..39	Casado: Sim	Casado: Não	NumCar: 0	NumCar:1	NumCar:0
100	1	0	0	1	0	1	0
200	1	0	1	0	0	1	0
300	1	0	0	1	1	0	0
400	0	1	1	0	0	0	1
500	0	1	1	0	0	0	1

Fonte[SRI 96]

Existem dois problemas quando aplicamos atributos quantitativos:

– *MinSup*: Se o número de intervalos for muito grande para atributos quantitativos (caso não estejam particionados), o suporte para cada intervalo pode ser menor do que o suporte especificado. Por isso, ao utilizar-se uma quantidade grande de

intervalos, algumas regras podem não ser descobertas, pois ficam abaixo do suporte mínimo estabelecido;

– *MinConf*: Existem algumas informações perdidas quando particionamos valores em intervalos. Algumas regras podem apresentar confiança mínima acima do limite quando o item estiver sozinho no antecedente (ou existir um pequeno intervalo).

Como exemplo, cita-se o fato da regra descoberta a partir da Tabela 2.11: "<Numero de Carros: 0→ Casado: Não" ter confiança de 100%". Se um intervalo para o número de carros (0..1), então para a regra:" "Número de Carros:0→ Casado: Não", terímos confiança de 66%.

TABELA 2.11 - Regras com Suporte Mínimo=40%, Confiança Mínima=50%

Regra	Suporte	Confiança
<Idade: 30.. 39> e <Casado: Sim> → <NumCarros:2>	40%	100%
<NumCarros:2> → <Casado: Não>	40%	60%

Fonte[SRI 96]

Cria-se uma situação a partir dos dois problemas: caso o intervalo seja muito longo, algumas regras não terão confiança mínima; se for muito pequeno, algumas regras não terão suporte mínimo.

Para identificar o problema é preciso considerar todas as variações contínuas sobre os valores dos atributos quantitativos, ou sobre intervalos particionados. O problema do *Minsup* desaparece desde que seja possível combinar valores adjacentes nos intervalos. O problema da *Minconf* ainda está presente, mas pode ser reduzido pelo incremento no número de intervalos encontrados sem o problema de *Minsup*. Entretanto, aumentando o número de intervalos enquanto simultaneamente combina-se os intervalos adjacentes faz surgir dois novos problemas em termos de tempo de execução e quantidade de regras geradas.

Na proposta do autor são considerados variações de valores (intervalos) adjacnetes de atributos quantitativos para evitar o problema do *Minsup*. Para atenuar o tempo de execução, é restrita a extenção de como valores adjacentes podem ser combinados, pela introdução de um parâmetro chamadao suporte máximo *Maxsup*.

[CHA 97] descreve uma técnica chamada APAC52. A utilização desta técnica, aplicada a um banco de dados de zoológico e censo demográfico, possui a vantagem de não necessitar intervenção do usuário. A proposta é discretizar qualquer atributo quantitativo em intervalos *equal-width* onde cada intervalo possui o mesmo número de registros. A técnica permite que regras positivas e negativas (descritas na próxima seção 1.3.4) sejam geradas. Como exemplo o autor cita as regras abaixo para os dados do zoológico:

Plumas → Pássaros (+)

Leite → Mamíferos (-)

[WEB 98] faz uma referência a construção de intervalos fazendo uma analogia às regras de associação taxonômicas. Segundo ela, as regras quantitativas não precisam ser combinadas de forma que os itens tenham um nodo-pai comum. A utilização deste critério

poderia gerar combinações que continuariam vazias, ou ainda que seriam equivalentes aos seus descendentes, numa hierarquia.

2.3.3.1 Definição Formal do Problema

O problema de descobrirmos regras de associação quantitativas pode ser dividido em cinco etapas, segundo a proposta de [SRI 96]:

1) Determinar o número de partições de cada atributo quantitativo;

2) Para atributos categóricos, mapear os valores dos atributos para um conjunto de inteiros consecutivos. Para atributos quantitativos que não são particionados em intervalos, os valores são mapeados em inteiros consecutivos conforme a ordem apresentada. Se um atributo quantitativo for particionado em intervalos, os intervalos são mapeados para inteiros consecutivos, e a sua ordem é preservada. O algoritmo enxerga somente valores de atributos quantitativos, e os valores que representam intervalos tornam-se transparentes ao algoritmo;

3) Descobrir o suporte para cada valor, para ambos atributos (categóricos e quantitativos). Adicionalmente, para os atributos quantitativos, valores adjacentes são combinados e serão tão grandes quanto seus suportes permaneçam inferiores ao suporte especificado pelo usuário. Conhecidas às variações e valores com suporte mínimo para cada atributo quantitativo e categórico, é formado o conjunto de itens freqüentes;

4) Utilizar os *itemsets* freqüentes para geração de regras;

5) Determinar as regras interessantes na saída.

Exemplo: Considerar a Tabela 2.9 de Pessoas, onde existem dois atributos quantitativos denominados idade e número de carros. Assumindo-se que no passo 1, a idade é particionada em quatro intervalos, como é apresentado na Tabela 2.12. Depois de mapearmos os intervalos em inteiros consecutivos, utilizando as Tabelas 2.12 e 2.13, a tabela se parece com a Tabela 2.14. Assumindo-se suporte mínimo de 40% e confiança mínima de 50%, a Tabela 2.15 apresenta alguns dos *itemsets* freqüentes, e a Tabela 2.16 algumas regras.

TABELA 2.12 - Particionamento da Idade e Mapeamento

Intervalos	Inteiro
20..24	1
25..29	2
30..34	3
35..39	4

Fonte[SRI 96]

TABELA 2.13 - Mapeamento Casado

Valor	Inteiro
Sim	1
Não	2

Fonte[SRI 96]

TABELA 2.14 - Mapeamento de Atributos

RecordId	Idade	Casado	NumCarros
100	1	2	1
200	2	1	1
300	2	2	0
400	3	1	2
500	4	1	2

Fonte[SRI 96]

TABELA 2.15 - Itemsets Freqüentes

Itemset	Suporte
{<Idade: 20..29>}	3
{<Idade: 30..39>}	2
{<Casado: Sim>}	3
{<Casado: Não>}	2
{<NumCarros: 0..1>}	3
{<Idade:30 ..39>, <Casado: Sim>}	2

Fonte[SRI 96]

TABELA 2.16 - Regras

Regras	Suporte	Confiança
<Idade:30..39> e <Casado:Sim>→<NumCarros:2>	40%	100%
<Idade:20..29>→<NumCarros:1>	60%	66.6%

Fonte[SRI 96]

2.3.3.2 Algoritmo

Neste estágio, os atributos quantitativos já estão particionados, e já estão criadas as combinações de intervalos de atributos quantitativos que possuem suporte mínimo. Estas possuem suporte mínimo formado pelos itens freqüentes.

São gerados todos os *itemsets* freqüentes utilizando-se um algoritmo baseado no Apriori, onde se descobrem regras de associação. O algoritmo proposto estende o procedimento de geração de itens candidatos para adicionar corte utilizando medidas de interesse, e utilizando uma estrutura de dados diferente para contar candidatos.

Representando *k-itemset* como um *itemset* de *K* itens. L_k representa o conjunto de *k-itens* freqüente, e C_k o conjunto de candidatos *k-itemsets*. O algoritmo faz múltiplos passes sobre o banco de dados, sendo que cada passe consiste de duas fases. Na primeira o conjunto de todos os freqüentes *(k-1)-itemsets*, L_{K-1}, descobertos em (*k-1*) passe, são utilizados para gerar *itemsets* candidatos C_k. O procedimento de geração dos candidatos considera *Ck* um *superset* dos conjuntos de todos os *k-itemsets* freqüentes. O algoritmo

então procura no banco de dados, para cada registro, os quais determinam quais os candidatos em *Ck* estão contidos no registro e incrementa o contador de seu suporte. No final do passe, *Ck* é examinado para determinar quais dos candidatos são freqüentes. O algoritmo termina quando *Lk* se tornar vazio.

2.3.3.3 Pós-Processamento

Um problema surgido pela combinação de intervalos para atributos quantitativos é que o número de regras descobertas pode ser muito grande. Medidas subjetivas de interesse sugerem que um padrão é interessante se for inesperado e ou acionável (o usuário pode executar uma ação).

Apresenta-se uma medida de interesse para identificar as regras interessantes na saída, chamada "greater-than-expected-value". Esta medida procura por regras especializadas e generalizadas para identificar regras interessantes. Considerando-se as regras descritas a seguir, onde cerca de um quarto das pessoas com idade entre 20..30 estão no grupo de idades 20..25.

<Idade: 20..30> → <Carros 1..2> (8% sup, 70% conf)
<Idade: 20..25> → <Carros 1..2> (2% sup, 70% conf)

A Segunda regra pode ser considerada redundante, pois é menos generalizada e não adiciona informações. Dada a primeira regra, esperamos que a segunda tenha a mesma confiança e suporte iguais a um quarto da primeira. Se o valor de confiança da segunda fosse 68% ou 73%, ela não transmitiria informações adicionais. A partir desta idéia, a noção de interesse, passa a ser a busca por regras onde o suporte e/ou a confiança são maiores do que o esperado.

2.3.4 Regras de Associação Negativas

No estudo realizado por [SAV 98] é considerado um problema complementar, denominado regras de associação negativas. Enquanto as regras de associação convencionais identificam itens que aparecem juntos em um número significativo de transações, as regras de associação negativas representam que, com certo grau de certeza, determinados itens não ocorrem quando outros específicos estão presentes nas transações. Um exemplo de regras de associação negativas é que "60% dos clientes que compram batata frita não compram água mineral com gás".

2.3.4.1 O Problema

Descobrir regras de associação negativas não é uma tarefa fácil, pois em grandes bancos de dados, existem milhões de itens. Caso estes milhões de transações sejam consideradas, muitas das combinações de itens não devem aparecer regularmente. Se a ausência de certa combinação, significar uma regra de associação negativa, pode-se então, gerar milhões de regras, dentre as quais muitas delas não são interessantes.

O foco de interesse das regras de associação negativas é descobrir conjuntos de itens, que possuem uma probabilidade baixa de serem comprados juntos.

Surgem dois problemas a partir da busca por regras de associação negativas:

- Na ausência de qualquer conhecimento sobre preferências de compras, espera-se que a compra de itens seja independente uma da outra. Uma regra onde os dois itens são negativos não traz informação relevante.

- Caso sejam procuradas combinações de itens que possuem suporte baixo, haverá um grande número de transações satisfazendo a condição dada, possuindo suporte próximo a zero.

Para compreender melhor o problema, o autor propõe o agrupamento de itens similares, através de taxonomia. Uma das suposições feitas neste trabalho é que itens pertencentes ao mesmo pai na taxonomia são esperados apresentando tipos de associações parecidas. Desde que sejam considerados somente os casos onde um suporte esperado possa ser computado, baseado na taxonomia, a classe de regras negativas gerada será restrita a um conjunto menor.

2.3.4.2 Definição Formal do Problema

Considerando $I = \{i1, i2, ..., i_m\}$ como um conjunto de *m* literais distintas chamadas itens. *T* é a taxonomia em *I*. Considerando $L \subseteq I$, sendo o conjunto de itens folhas em *T*, e $C \subset I$, sendo o conjunto de nodos internos chamados de categoria. *D* é o conjunto de transações de tamanho variável sobre *L*. Cada transação *T* contém um conjunto de itens $i_1, ...i_K \subset L$. Uma transação também possui um identificador chamado *TID*. No geral, um conjunto de itens é chamado *itemset*. O número de itens de um *itemset* é denominado de tamanho do *itemset*. *Itemsets* de tamanho *k* são chamados *k-itemset*. Para um *itemset X*.*Y*, se X é um *m-itemset* então *Y* é chamado de *m-extensão* de *X*. Um *itemset* $X \subset I$, possui suporte(*X*)=*s*, caso a fração da transação em *D* contiver *X* igual a *s*. Uma regra de associação negativa é uma implicação na forma *X-/-> Y*, onde *X*, $Y \subset I$, e $X \cap Y = 0$ é chamado de antecedente e *Y* é chamado de conseqüente da regra. Toda regra também possui uma medida de interesse. Definimos a medida de interesse *RI* de uma regra negativa a seguir:

$$RI = \frac{E[Suporte(X \cup Y)] - Suporte(X \cup Y)}{Suporte(X)}$$

Onde, *E [suporte(X)]*: é o suporte esperado de um *itemset* X;

O problema de descobrirmos regras negativas é descrito a seguir. Dado um banco de dados de transações de clientes *D* e uma taxonomia *T* no conjunto de itens, descobrir todas as regras *X-/-> Y* tais que:

a) Suporte (*x*) e suporte de (*y*) sejam maiores que o suporte mínimo *Minsup*;
b) A medida de interesse seja maior que *MinRI*, onde *Minsup* e *MinRI* são especificados pelo usuário.

A primeira condição é importante para assegurar que sejam geradas regras estatisticamente significantes. O problema pode ser decomposto em duas sub-tarefas:

1) Descobrir *itemsets* os quais suporte desvia-se de *Minsup* X *MinRI*, que são denominados itens negativos;
2) Gerar as regras negativas depois dos itens negativos serem descobertos.

Para descobrir os *itemset* negativos é preciso primeiramente, descobrir todas os *itemsets* generalizados nos dados, isto é, os *itemsets* em todos os níveis da taxonomia onde o suporte é maior que o suporte mínimo especificado pelo usuário. Posteriormente é preciso identificar *itemsets* negativos baseados em grandes *itemsets* da taxonomia e assegurar o suporte esperado. Finalmente, contar o suporte atual para os *itemsets* candidatos e manter somente *itemset* negativos.

2.3.4.3 O Algoritmo

No algoritmo *Naive*, um dos apresentados por [SAV 98], a iteração é realizada em duas fases. Na primeira fase de iteração *k*, são computados os *itemsets* generalizados de tamanho *k*. Na Segunda fase, são gerados primeiramente os *itemsets* candidatos negativos de tamanho *k*. Posteriormente, o suporte é contado fazendo-se um passe sobre os dados.

O algoritmo necessita de dois passes sobre o banco de dados durante a iteração totalizando *2 X N* passes, onde *N* é o número total de iterações.

Em outro algoritmo apresentado pelo autor, chamado *Improved*, são feitas duas otimizações sobre o algoritmo *Naive*. Primeiro todos os pequenos *1-itemset* são excluídos da taxonomia; segundo ao invés de gerar os *itemsets* negativos durante cada iteração, são gerados num único passe após a geração dos *itemsets* freqüentes. A primeira otimização reduz o número de candidatos negativos. A Segunda reduz o número de passes sobre os dados de *2 X N* para *N+1*.

O trabalho proposto por [SAV 98] é um trabalho complementar no contexto de mineração de dados através de regras de associação. As regras negativas parecem ser aplicáveis a qualquer conjunto de dados, porém a sua complexidade é muito grande, no que diz respeito à identificação do que realmente pode ser considerado uma regra de associação negativa.

2.3.5 Mineração de Regras de Associação com Suporte Mínimo Múltiplo

[BIN 99] levanta uma abordagem sobre a aplicação de suportes diferentes para itens que aparecem com menor freqüência entre os dados. Se somente um valor para suporte mínimo for utilizado para todo banco de dados, o modelo implícito assume que todos os itens nos dados são da mesma natureza e/ou possuem freqüências similares. Entretanto, o caso citado não é verdadeiro para aplicações reais. Na maioria das aplicações, alguns itens

aparecem freqüentemente, enquanto outros aparecem raramente. Se o suporte mínimo for muito alto, as regras que envolvem itens raros não aparecem. Para descobrir-se regras que envolvam itens freqüentes e raros o suporte deveria ser decrementado, causando conseqüentemente um número exorbitante de regras geradas. Este dilema é chamado por Bing Liu de "problema dos itens raros". Para contonar o problema ele propõe uma técnica que permite que o usuário especifique suportes mínimos múltiplos de acordo com a reflexão feita sobre a natureza de cada item e sua freqüência no banco de dados.

Em transações de supermercado, caso o objetivo fosse o de descobrir regras que envolvem itens pouco freqüentes como assadeira e processador de alimentos, seria necessario atribuir um suporte baixo para descobrir tal regra:

processador de alimentos → assadeira [sup=0,5, conf=60%]

Entretanto, este suporte pode causar regras sem sentido como a citada:

Pão, queijo, leite → cerveja [sup=0,5, conf=60%]

Saber que os clientes compram estes quatro itens juntos, com suporte de 0.5, não é interessante, pois todos os itens são comprados freqüentemente no supermercado. Para esta regra ser útil o suporte deveria ser mais alto. Quando este problema é encontrado, alguns autores sugerem a divisão dos dados em pequenos blocos (regras quantitativas) de acordo com a freqüência dos itens. Entretanto, segundo [BIN 99] quando este tipo de regra é gerada, são encontradas apenas aproximações.

No modelo proposto por [BIN 99] são mantidos os conceitos apresentados para regras de associação, com exceção do suporte. Neste modelo, o suporte mínimo de uma regra é expresso em termos do suporte mínimo de seus itens (MIS). Cada item no banco de dados pode ter um suporte mínimo especificado pelo usuário. Fornecendo valores diferentes de MIS para os diferentes itens, o usuário estará expressando suportes diferentes para cada regra..

Representando MIS(I) como valor de MIS para um item, o suporte mínimo de uma regra é o menor valor MIS dentre os itens que compõe a regra. Este fato permite que tenhamos suportes altos para regras que envolvem somente itens freqüentes e suportes baixos para regras que envolvem itens raros.

Exemplo: O usuário especifica os seguintes valores de MIS:

MIS (pão) = 2%
MIS (roupas) = 0.2%
MIS (sapatos) = 0.1%
A regra (i) não satisfaz o filtro de suporte mínimo, enquanto que a regra (ii) satisfaz.

(i) roupas → pão [sup=0.15, conf=70%]
(ii) roupas → sapatos [sup=0.15, conf=70%]

Algoritmos eficientes para busca de itens freqüentes estão baseados na busca por níveis. Considerando-se k-*itemset* um conjunto de itens com k itens, no nível-1 , todos os

conjuntos de itens freqüentes 1-*itemset* são gerados, no nível-2 todos os conjuntos de itens freqüentes 2-*itemset* são gerados. Para este modelo, caso seja utilizada esta técnica, algumas regras importantes podem ser perdidas, como no exemplo abaixo.

Exemplo: Considerando quatro items 1, 2, 3 e 4 no banco de dados, seus suportes mínimos para os itens são:

MIS (1)=10%
MIS (2)=20%
MIS (3)=5%
MIS (4)=6%

Caso seja descoberto que os itens {1,2} possuem 9% de suporte no nível 2, então eles não satisfariam nem MIS(1), nem MIS(2). Utilizando um algortimo existente com o Apriori, os itens seriam descartados e os itens freqüentes {1,2,3} e {1,2,4} não seriam geradas no nível 3. É proposto um algoritmo para gerar conjuntos de itens freqüentes que satisfazem uma propriedade de ordenação ascendente dos itens de acordo com seus MIS, denominado MSApriori, evitando que itens sejam eliminados.

2.3.6 Trabalhos Realizados pelos Grupos de Pesquisa

Regras de associação é um tema freqüentemente abordado em trabalhos realizados no instituto de informática da UFRGS.

[CER 99] desenvolveu um protótipo, encapsulando as funções do algoritmo Apriori, que possibilita o acesso a bancos de dados relacionais de forma genérica. É possível acessar todos os bancos registrados na máquina local em que a ferramenta esta sendo executada, através de driver ODBC ou BDE. A ferramenta foi projetada para acesso genérico para que possa ser utilizada em bancos de dados com formatos diferentes e com qualquer SGBD. A ferramenta projetada foi validada sobre um conjunto de dados do SUS (Sistema Único de Saúde).

O processo de mineração de dados pode ser aplicado na análise da utilização da Web de forma produtiva uma vez que, em virtude das características do ambiente em questão, simples estatísticas não são suficientes para a real compreensão do comportamento do usuário. [BRU 2000] apresenta a proposta de um novo modelo de processo para a obtenção de regras de associação a partir do registro de acesso de usuários às páginas de um site, chamado *Access Miner*. A medida que um usuário interage com um site são coletados dados sobre ele e sobre como ele interage com o conteudo oferecido. Como exemplo pode-se descobrir de onde vieram, quais as paginas que visitaram e quanto tempo visitiram o mesmo site. [BRU 2000] propõe um modelo onde aplica o algoritmo Apriori sobre os dados coletados de logs de servidores. Na etapa de pós-processamento ele propõe como vantagem, a eliminação dos caminhos redundantes das paginas dos sites.

[CAM 2002] realizou um estudo sobre a aplicação de regras de associação sobre dados do comércio varegista e propõe um algoritmo para o estudo em questão que tem um bom desempenho em termos de tempo de processamento. [CAM 2002] cita o fato das transações de bancos de dados do comércio varegista possuirem um número médio de itens inferior a dois. Outros estudos realizados por [AGR 94] e [SRI 96] utilizam dados de testes que contenham de cinco a vinte itens por transação. Segundo o autor a primeira parte do

processamento é mais onerosa e nesta fase, o algoritmo executa múltiplas passagens pelo banco de dados e uma das principais caracterísitcas dos algoritmos estudados é que no iníco de cada passagem são gerados todos os possíveis conjuntos de itens candidatos. [CAM 2002] propõe o algoritmo Mirabit que não gera o conjunto de itens candidatos e que apresenta um desempenho superior em bancos que possuem poucos itens por transações, como é o caso do comércio varegista.

2.4 Conclusão

Neste capítulo foi apresentada uma revisão bibliográfica do processo de descoberta de conhecimento, dando enfoque as regras de associação. Na literatura foram encontrados diversos trabalhos interessantes sobre regras de associação como mineração através de regras de associação em bancos de dados muito densos, regras de associação taxonômicas, regras de associação quantitativas, regras de associação negativas e regras de associação com suportes múltiplos. Foram também, citados os trabalhos mais recentes desenvolvidos no Instituto de Informática da UFRGS sobre o tema mineração de dados através de regras de associação.

3 Direto

Este capítulo apresenta uma descrição sobre o software de correio, agenda e catálago eletrônico Direto [DIR 2002] que está sendo implantado nos órgãos do governo do Estado do Rio Grande do Sul. Neste capítulo é descrito inicialmente o histórico do correio eletrônico na PROCERGS, posteriormente são apresentados os módulos já disponíveis do software Direto e uma proposta para os filtros de mensagens [BAL 2002], assistente de feedback [MEL 2002] e canais de informação [BAL 2002].

3.1 Considerações Iniciais

O Direto é uma ferramenta desenvolvida pela PROCERGS, que inclui os serviços de correio eletrônico, agenda e catálogo coorporativo e que se destina a integrar todos os órgãos do governo estadual. O governo do Estado pode ser visto como uma organização distribuída capaz de gerar inúmeras informações de diversas naturezas. A divulgação das informações geradas é uma tarefa complexa, pois existe a dificuldade de localizar o público realmente interessado, sem que haja a necessidade de distribuí-la de uma forma generalizada.

Em [AVI 2000] é citado o fato de que o correio eletrônico é bem mais do que um meio de comunicação digital: é uma fonte rica de informações, onde o usuário necessita de facilidades para armazenar, organizar e recuperar informações dentre grandes volumes.

A popularização do correio eletrônico pode fazer com que a quantidade de informações geradas cause uma sobrecarga na caixa postal do usuário e como conseqüência a perda de informações ou a dificuldade para identificá-las. O tempo necessário gasto para classificar a informação, também aumenta de acordo com a quantidade de mensagens disponíveis.

Um serviço de filtro de correio eletrônico capaz de classificar e rejeitar mensagens pode contornar o problema da sobrecarga de informações nas caixas postais. [BAL 2002] propõe um sistema para filtros de mensagens de correio eletrônico para o software Direto, com o objetivo de aumentar a produtividade do usuário. Além disso, [BAL 2002] propõe um serviço de canais de informação com a finalidade de divulgar informações através de assinaturas de canais.

3.2 Histórico do Direto

Segundo [BAL 2002], a PROCERGS atua como responsável pela área de informática sobre os órgãos estaduais desde 1972. Já em 1991 a empresa buscou uma ferramenta para suprir a necessidade de comunicação entre os órgãos estaduais. Nesta época, realizou-se um projeto de avaliação das ferramentas disponíveis no mercado com serviços de mensagens, agenda e quadros de aviso e a solução encontrada foi a implantação do Memo da Eurosoft. A ferramenta implantada possui interface caracter e algumas limitações. Diante das novas tecnologias em 1996, a PROCERGS fez uma nova aquisição e implantou a ferramenta de correio eletrônico Lotus Notes. Nos anos seguintes, o Estado manteve as duas soluções em uso. Em 1999, surgiu a idéia de padronizar o serviço de correio eletrônico para os órgãos estaduais. Os pontos críticos que motivaram a demanda por uma nova solução foram:

- A falta de uma ferramenta padrão de correio eletrônico para o governo do Estado que facilitasse a comunicação interna e externa;
- Usuários convivendo com duas ferramentas de correio eletrônico;
- Montagem de uma infra-estrutura de hardaware e software para o Estado que deveria depender do orçamento disponível;
- Familiaridade dos usuários do Estado com ferramentas de correio eletrônico como Exchange, Outlook e Eudora;
- Falta de integração entre agendas do Memo e do Lotus Notes.

A solução encontrada foi o desenvolvimento de uma ferramenta que pudesse satisfazer as necessidades, e que utilizasse os recursos do ambiente. O Direto, que é a solução adotada foi desenvolvido pela PROCERGS e está sendo aprimorado com parcerias entre o governo do estado e universidades.

3.3 O Projeto Direto

Segundo [DIR 2002], o Direto é um produto de correio, agenda e catálogo com acesso pela Internet, fundamentado na idéia de software livre, com código aberto (*free software & open source*), independência de plataforma e flexibilidade de aperfeiçoamento. Desenvolvido no Brasil pela PROCERGS (Cia. de Processamento de Dados do Estado do Rio Grande do Sul), o produto tem como objetivo principal atender a demanda de um software de comunicação de baixo custo, que interligue os diversos órgãos do Estado. Além disso, o Direto poderá ser utilizado por empresas e entidades que quiserem valer-se do conceito de software livre para implementar esta solução em seus estabelecimentos.

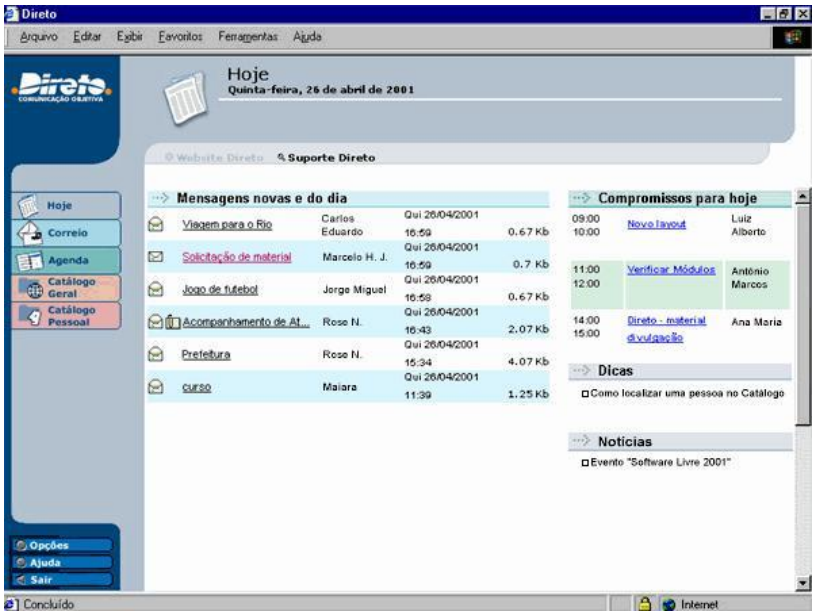
O usuário cadastrado no Direto e conectado à Internet pode acessar seus compromissos e mensagens de qualquer lugar, sem precisar fazer nenhum download ou instalar um programa específico. Além disso, pela sua estrutura modular e por ter seu código fonte disponível, o DiretoGNU possui grande facilidade de adaptação aos mais diversos ambientes, permitindo sua total customização conforme as necessidades do usuário, permitindo, inclusive, que novos módulos sejam livremente desenvolvidos e incorporados ao produto.

O processamento do Direto é centralizado no servidor e o usuário precisa estar on-line para acessar os dados. A segurança dos dados pode ser garantida através da criptografia fornecida pelo protocolo SSL, suportado pelos principais *browsers*.

Os módulos atualmente disponíveis para o usuário são:

Módulo Hoje: É a página de abertura do Direto, apresentada na Figura 3.1. Através dela o usuário tem a visão geral das mensagens recebidas e a listagem dos compromissos do dia, além do serviço de notícias e da seção de dicas sobre o produto.

FIGURA 3.1 - Interface Módulo Hoje

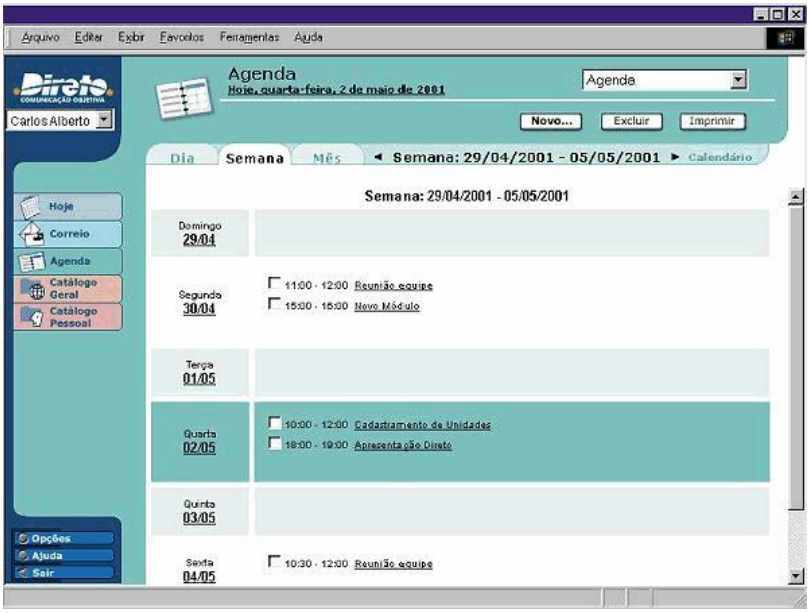


Módulo Correio - Funciona como um serviço de correio eletrônico normal, com as facilidades da interface para Internet, como é apresentado na Figura 3.2. Aqui o usuário pode ler, enviar, receber e encaminhar mensagens para qualquer contato, dentro ou fora da empresa.

FIGURA 3.2 - Interface do Módulo Correio

Módulo Agenda – Neste módulo o usuário tem a oportunidade de agendar e acompanhar seus compromissos diários, sendo possível marcar compromissos com antecedência e repetí-los em outros dias de maneira simples e prática. O Módulo Agenda é apresentado na **Error! Reference source not found.**.

FIGURA 3.3 - Interface do Módulo Agenda

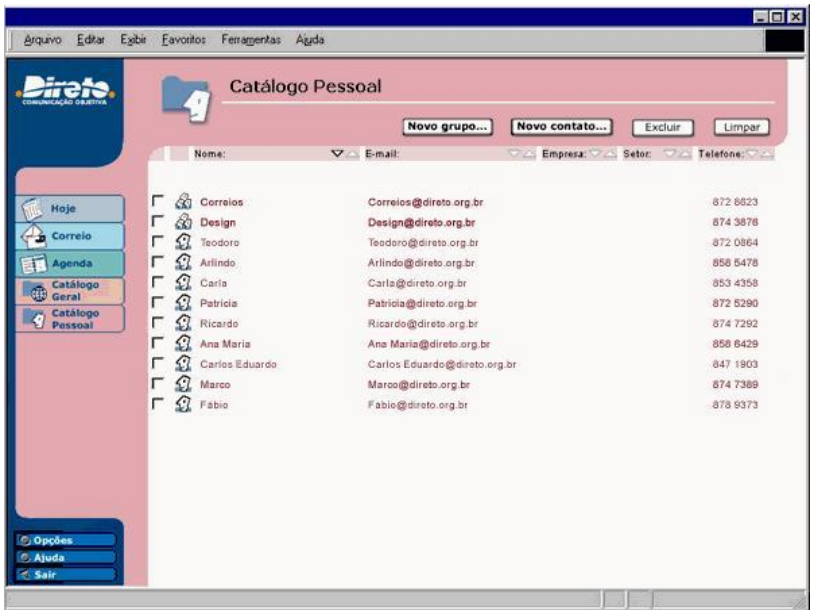


Módulo Catálogo Geral - É o catálogo de contatos da empresa, onde é possível localizar os usuários previamente cadastrados.

FIGURA 3.4 - Interface do Catálogo Geral de Usuários

Módulo Catálogo Pessoal - É o catálogo de contatos pessoais do usuário. Com ele é possível consultar informações de usuários previamente cadastrados, bem como incluir e excluir contatos ou grupo de contatos pessoais.

FIGURA 3.5 - Interface do Catálogo Pessoal



O Direto possui interface Web para ser acessado a partir de um *browser*. A interface Web possui várias vantagens em relação a um cliente específico instalado na máquina do usuário. O sistema pode ser acessado de qualquer ponto da Web sem necessidade de instalação de um *browser* ou de determinada configuração específica. O cliente é independente de sistema operacional, necessitando apenas de um *browser* Web. Os usuários já possuem experiência com este tipo de interface. A criptografia dos dados é garantida pelo protocolo SSL, suportado pelos principais *browsers* Web. O único inconveniente é que nesta solução todo processamento fica centralizado no servidor e o usuário precisa estar on-line para acessar seus dados.

A solução teve como principais requisitos tecnológicos os seguintes itens:

- Seguir protocolos padrão Internet;
- Independência de plataforma no cliente e no servidor;
- Prover independência de Gerenciador de Banco de Dados;
- Prover independência de Servidor Web;
- Ser modular, ou seja, permitir que novas funcionalidades pudessem ser adicionadas no futuro;
- Possuir baixo custo.

A solução baseou-se em protocolos padrão Internet. O Direto está estruturado de forma modular utilizando protocolos vCalendar para agenda, SMTP e IMAP para correio e LDAP para catálogo.

3.4 Distribuição de informações através do Direto

Como um dos princípios do Direto é padronizar serviços aos órgãos estaduais e distribuir informação de maneira adequada aos servidores, surgiu a preocupação com respeito à quantidade de informação que possa ser gerada e o interesse do usuário em receber tais informações.

Prevendo a necessidade de direcionar ao usuário apenas informações de interesse, foram propostos no escopo do projeto Direto, filtros de mensagens de usuários, canais de informações para o Direto e listas de discussão. Os dois primeiros itens citados já possuem um escopo definido e serão apresentados nas próximas subseções.

3.4.1 Filtros de Mensagens para o Direto

Atualmente o correio eletrônico está altamente difundido entre os usuários da Internet, sendo um dos serviços mais utilizados. A popularização do serviço de correio eletrônico aliado às mensagens de correntes, propagandas e listas de discussões dificultam o reconhecimento da informação útil e sua classificação por parte do usuário.

Uma mensagem de correio eletrônico possui uma estrutura definida. Uma mensagem é composta de informações estruturadas e semi-estruturadas. Uma mensagem normalmente contém um corpo que é a parte semi-estruturada e contém diversos cabeçalhos que identificam as informações estruturadas como remetente, data de envio, assunto e prioridade da mensagem.

[BAL 2002] apresenta uma solução para o problema de sobrecarga de informações nas caixas postais dos usuários do software Direto, através de um filtro de mensagens. A solução apresentada está baseada na linguagem Sieve. O processo de filtragem inicia com a definição das regras para as mensagens por parte do usuário, através de uma interface web. As regras são condições testadas sobre as mensagens que podem ou não gerar alguma ação. O filtro de mensagens é composto por um conjunto de regras definidas pelo usuário. Cada usuário possui permissão para criar, editar e listar as regras referentes a sua caixa postal. As regras são armazenadas ordenadamente e cada condição verdadeira gera uma ação. As condições propostas para os filtros através das regras são aplicadas sobre algum cabeçalho das mensagens. Os seguintes cabeçalhos são avaliados na proposta de [BAL 2002]:

To
Cc
Subject
From

A interface web para criação de regras para os filtros do Direto possui as seguintes características:

Regra X

Quando o cabeçalho <*escolher cabeçalho*>
Contiver o texto <*definir texto*>
<*Ação*>

Além da tela de criação de filtros é disponibilizada ao usuário, uma segunda tela que exibe a lista de todas as regras ordenadamente. Esta tela possui as funcionalidades de excluir, editar, incluir e alterar a ordem em que as regras são executadas.

As ações que podem ser executadas sobre uma mensagem são as seguintes:

- *Fileinto*: armazena uma mensagem em uma pasta específica, diferente da caixa de entrada do usuário;
- *Reject*: recusa a entrega da mensagem. A mensagem é retornada ao remetente com uma justificativa pelo não recebimento pelo destinatário;
- *Redirect*: redireciona a mensagem para outro endereço eletrônico sem deixar cópia na caixa postal do usuário;
- *Keep*: não realiza ação sobre a mensagem. Esta é entregue normalmente na caixa postal do usuário.

[BAL 2002] apresenta uma comparação entre o sistema de filtro de mensagem do Direto e o MS Outlook Express, onde é possível enxergarmos resumidamente as principais características dos filtros de mensagens para o Direto.

TABELA 3.1 - Comparação Filtros Direto (Sieve) X Outlook Express

	MS Outlook Express	Sieve
Roda no cliente	Sim	Não
Roda no Servidor	Não	Sim
Executa filtro imediatamente quando a mensagem chega no servidor	Não	Sim
Avalia cabeçalho To, Cc, Subject	Sim	Sim
Avalia prioridade	Sim	Sim
Avalia tamanho	Sim	Sim
Avalia Corpo de mensagem	Sim	Não
Verifica se existem anexos	Sim	Não
Move ou copia para pasta	Sim	Sim
Rejeita mensagem com justificativa	Não	Sim
Exclui mensagem	Sim	Sim
Realça com cor ou sinaliza	Sim	Não
Marca como lida	Sim	Não
Responde com mensagem	Sim	Sim
Redireciona para outro endereço	Sim	Sim

[BAL 2002] relata que a linguagem Sieve pode ser executada no servidor ou no cliente de e-mail. Na comparação apresentada, o script Sieve estava sendo executado no servidor. O fato de estar sendo executado no servidor pode apresentar algumas vantagens. Em ações como redirecionar mensagens, rejeitar com justificativa a resposta gerada é mais rápida. Em outros tipos de ações como mover para a pasta, o fato de executar no servidor ou no cliente não faz diferença para o usuário. A solução inicial de filtros de mensagens para o Direto possivelmente será aperfeiçoada e novas funcionalidades serão incorporadas à interface e à estrutura de armazenamento. Como complemento ao sistema de filtros de

mensagens do Direto existe um trabalho de criação de um assistente de feedback que permite a visualização das estatísticas das preferências do usuário.

3.4.1.1 Assistente de Feedback para o Serviço de Filtragem do Software Direto

[MEL 2002] propõe um modelo com a finalidade de assistir o usuário a fim de interpretar a sua preferência, através do seu comportamento com as mensagens trafegadas no Direto. O modelo proposto é dividido em três etapas com funções distintas:

Coleta dos termos e eventos das mensagens: Nesta etapa são observadas as ações dos usuários sobre as mensagens vindas do serviço de filtragem do Direto. A mensagem pode conter diversos termos (economia, saúde, informática, educação). Os eventos são as ações que o usuário executou sobre a mensagem, durante a utilização do correio (ler, excluir, encaminhar). O produto desta etapa será um histórico que deverá conter as identificações das mensagens com seus termos e os eventos executados pelo usuário. No histórico as mensagens serão representadas por registros contendo os termos mais relevantes e valores binários representando se os eventos foram executados ou não.

Classificação: A função desta etapa é a de classificar as preferências do usuário sobre os termos das mensagens, contidos no histórico, através dos eventos que representam as ações do usuário. Cada evento relacionado a uma mensagem é um indicador do interesse do usuário com relação à mensagem. Esse interesse é representado por probabilidades condicionais associando os eventos às classes de preferência dos usuário. As classes são representadas pelas letras A, B, C, D, E que determinam o percentual de preferência do usuário por um determinado termo.

Feedback: esta etapa tem a função de interpretar as classes de preferência dos termos, representadas pelos pesos, assinaladas pelo classificador na segunda etapa. Esta interpretação cria retornos de sugestões para o usuário, a fim de permitir ajustes dos parâmetros dos filtros de mensagens no ambiente do Direto.

3.4.2 Canais de Informação para o Direto

O serviço de canais de informação para o software Direto tem por objetivo distribuir informações de forma eficiente, atingindo somente os usuários nela interessados, no domínio do Direto no Estado do Rio Grande do Sul. O serviço deve ser composto de diversos canais, onde cada um divulga informações de um determinado assunto. As informações a serem distribuídas pelos canais de informação são de diferentes naturezas: leis, acontecimentos, eventos, palestras, artigos, saúde, educação, esporte, transporte, entre outras.

Divulgar a todos os interessados todas as informações geradas pelo Estado sobre um determinado assunto é um processo complicado. Os canais de informação poderiam criar uma teia de divulgação e distribuição de informações em todo Estado. Através de um canal, a informação seria distribuída para quem realmente possui interesse na mesma, com um custo baixíssimo, através da assinatura dos canais de interesse.

O serviço poderia ser composto por filtros definidos pelo usuário, sobre conteúdos de textos. Para obter resultados mais precisos e eficientes na filtragem, o sistema estimula,

através de facilidades, o usuário a informar novos termos na definição dos filtros. Como conseqüência da criação de canais de informação teríamos o início do processo de criação de perfis dos usuários do sistema, já que cada usuário identifica suas áreas de interesse através da inscrição em canais de informação. Isso possibilita a identificação de assuntos de preferência do usuário.

3.4.2.1 Descrição dos Canais de Informação

Os canais de informação serão utilizados para distribuir informações por tópicos, de forma eficiente e rápida, para todos usuários que tiverem interesse. A idéia dos canais de informação é que eles sejam utilizados de forma unidirecional para divulgação e distribuição de informações, diferente de uma lista de discussão. Cada canal irá distribuir informações de determinado assunto. Inicialmente os assuntos serão pouco abrangentes, e de acordo com o interesse serão refinados e divididos em outros canais até atingirem a granularidade ideal.

Dependendo do contexto em que serão empregados, os canais de informação podem apresentar critérios diversos:

- Armazenamento do histórico: o histórico do canal pode servir de base para a aplicação de técnicas que auxiliem o administrador na divisão do canal;
- Moderador do canal: cada canal pode possuir ou não moderadores;
- Com ou sem filtro: definição de canais que permitam ou não a criação de filtros por parte do usuário;
- Organizações autorizadas ou domínio de assinantes do canal: esta opção serve para definir a abrangência de um canal de informação;
- Usuários cadastrados: lista de todos usuários cadastrados no canal.

Para os canais criados existem diferentes níveis de permissões para os usuários. Os administradores criam e mantém os canais e definem domínios de abrangência dos canais. Os aprovadores de canais aprovam ou rejeitam o texto da mensagem a ser divulgada e definem seu horário de envio. Assinantes podem listar os canais em que estão inscritos e assinar novos canais.

3.4.2.2 Inclusão do texto para envio

Após preparar a informação a ser divulgada é preciso que o usuário escolha um ou mais canais para enviá-la. Depois de escolhidos os canais e informado o texto, o sistema realiza a sumarização do texto, onde é gerado um vetor contendo os termos que melhor representam o documento e os seus respectivos pesos. A geração deste vetor de termos inclui a eliminação de *stopwords*, a identificação de radicais e geração de sinônimos.

O sumário do texto é retornado ao usuário que está divulgando a informação. Os termos resultantes e seus pesos podem ser editados, excluídos e novos termos podem ser inseridos.

Após a inclusão do texto e sumarização, este é enviado ao moderador do canal, se for o caso. A aprovação do texto depende da opinião do moderador e caso haja rejeição o texto retorna com uma justificativa de reprovação. Caso aprovado, o texto é enviado aos assistentes do canal e armazenado juntamente com seu sumário, identificação do remetente e do aprovador, além da data e da hora.

3.4.2.3 Entrega do texto e criação do filtro pelo usuário

No momento da chegada de uma mensagem na caixa postal do usuário, é verificado se o sumário, que vem anexado a mensagem, coincide com os termos do filtro do usuário. O vetor do texto é comparado ao filtro do usuário para o canal específico, caso o usuário tenha filtro para o canal em questão. Cada usuário pode possuir um filtro específico para cada canal assinado. Se o texto estiver com a similaridade desejada pelo usuário com relação ao seu filtro, o texto é entregue na sua caixa postal, caso contrário é descartado.

Na especificação dos filtros, os usuários geralmente tendem a colocar dois ou três termos para representar o assunto da informação que eles desejam receber. No sistema de canais de informação, que está baseado no Modelo do Espaço Vetorial [SAL 75], poucos termos dificilmente representam bem um texto ou uma área do conhecimento. A forma tradicional de criação de filtros por parte do usuário poderia ser a informação de termos de interesse e seus respectivos pesos. Uma maneira encontrada para estimular a informação de novos termos foi à inclusão de uma funcionalidade, onde é possível que o usuário informe um texto de seu interesse para o filtro. A partir do texto informado, são coletados termos representativos e adicionados ao filtro do usuário.

3.5 Considerações Finais

Neste capítulo é apresentado o histórico da utilização do correio eletrônico pelos órgãos estaduais e o software de correio e agenda Direto. O Direto está sendo implantado nos órgãos do governo estadual para padronizar e substituir as ferramentas de correio eletrônico Memo e Lotus Notes. São apresentados os módulos já disponibilizados ao usuário e uma proposta para filtros de mensagens e canais de informação para o Projeto Direto.

4 Regras de Associação Aplicadas a Mensagens

Neste capítulo é apresentada uma proposta para mineração de dados para o projeto Direto, capaz de auxiliar na identificação dos perfis de seus usuários nos filtros de mensagens, canais de informação e possíveis listas de discussão do Direto. Os filtros de mensagens realizam ações sobre as mensagens como classificar ou descartar, evitando que usuários percam informações importantes decorrentes de sobrecargas nas caixas postais. Os canais de informação atuam na distribuição unidirecional da informação através da assinatura realizada pelo usuário em canais de seu interesse.

Os filtros de mensagens, os canais de informação e as listas de discussão ainda não foram disponibilizados aos usuários e como consequência o Direto não possui mensagens públicas, por isso para os experimentos serão utilizados dados obtidos a partir de uma lista de discussão da linguagem Java. Neste capítulo é descrita a etapa de pré-processamento e as ferramentas utilizadas durante a mineração.

4.1 Considerações iniciais

Com base na qualidade da informação que deve ser distribuída através das instituições estaduais e a definição da proposta de filtros de mensagens e canais de informação para o Direto, surgiu a atual proposta que tem por objetivo melhorar ainda mais os serviços disponibilizados pelo software, através de um estudo realizado sobre o histórico das mensagens de cada usuário. Como mencionado no capítulo anterior, cada usuário do Direto deve possuir dois tipos de filtros: um filtro para mensagens de usuários e outro filtro para o serviço de canais de informação.

A presente proposta é a de minerar, através de regras de associação, os históricos dos usuários na utilização de filtros de mensagens e canais de informação. Para isso serão utilizadas três ferramentas que implementam as regras de associação e apresentam alguma peculiaridade distinta.

A primeira ferramenta escolhida é o Intelligent Miner da IBM, onde é possível descobrirmos regras utilizando o algortimo Apriori, além de permitir a criação de uma taxonomia para os dados. A segunda ferramenta é o Magnus Opus que permite a criação de intervalos entre os itens e conseqüentemente a simulação de resultados para regras quantitativas, além de propor dois filtros preditivos para regras de associação. A terceira ferramenta é o CBA que permite a especificação de suportes diferentes para os diversos itens, proporcionando que itens que ocorrem com pouca freqüência possam ser estudados, associados a itens que aparecem mais freqüentemente nos resultados.

4.2 Mineração de Dados nos Canais de Informação do Direto

O serviço de canais de informação para o Direto tem por objetivo a distribuição de informações no domínio do Direto no estado do Rio Grande do Sul. As informações devem ser distribuídas de forma eficiente, atingindo todos e somente os interessados. O serviço é composto de diversos canais, e cada um deve suprir as informações a serem divulgadas em determinados assuntos. Cada usuário do Direto pode definir filtros para os canais de informação baseando-se no conteúdo de texto, isto é, além de escolher os canais que deseja assinar, o usuário pode definir termos que estejam presentes no corpo das mensagens de

interesse, ou ainda informar textos de seu interesse de onde são extraídos os termos principais com seus repectivos pesos.

O filtro para o serviço de canais de informação armazenará no LDAP os termos dos filtros do usuário com seus respectivos pesos. Os dados do filtro referente às opções de cada usuário serão lidos do LDAP para que as carasterísticas individuais possam ser interpretadas. O resultado de uma consulta ao LDAP poderá ser uma string separando por "/" os termos referentes a cada usuário, com seus respectivos pesos. Um exemplo da leitura do LDAP sobre os termos presentes nos filtros dos canais de informação de um dos usuários é apresentado a seguir, no formato termo = peso:

java=12.5/j2ee=6.8/ejb=7.0/javabeans=4.3/design=3.2/pattern=7.3
voleibol=10.5/surf=1.0/futebol=15.0
saúde=12.0/medicina=7.0/posto de saúde=5.0/atendimento=3.0/consulta=4.0
educação=8.0/biologia=3.0/

Pelos termos especificados pelo usuário poderiam ser sugeridos canais relativos a linguagens de programação, por exemplo, e quando houvesse o particionamneto deste canal poderia ser sugerida a assinatura de um novo canal relativo a Linguagem Java.

Os canais que o usuário assina também poderão ser obtidos por uma consulta ao LDAP. O resultado será o nome de cada canal, que poderá vir na mesma estrutura que a consulta do filtro.

"canal informática/canal esportes/canal saúde/canal educação"
“canal cultura/canal música/canal educação”

Com o objetivo de identificar o perfil dos usuários dos canais de informação é proposta a mineração de dados utilizando informações de preferência de canais e termos a eles associados. As ferramentas disponíveis no mercado não possuem o formato de saída do LDAP como entrada para os dados fontes. Seria necessário aplicarmos um pré-processamento nos dados coletados no LDAP. Uma sugestão seria a geração de tabelas num banco de dados relacional, com um formato semelhante à Tabela 4.1 apresentada a seguir:

TABELA 4.1 - Tabela de Termos e Pesos para os Canais de Informação

Usuário	Canal de Informação	Termos	Peso
Pedro	Saúde	Hospitais	10.0
Pedro	Saúde	Postos de Saúde	10.0
Pedro	Saúde	Consultas	7.0
Pedro	Saúde	Leitos	6.0
Pedro	Saúde	Cardiologia	5.0
Pedro	Saúde	Pediatria	4.0
Roberto	Educação	Simpósio	10.0
Roberto	Educação	Educação Física	5.0
Roberto	Esportes	Voleibol	10.0
Roberto	Esportes	Campeonato	7.0

Sugere-se ainda, a integração dos dados resultantes do pré-processamento dos canais de informação ao catálogo geral do Direto, onde é possível identificarmos a instituição, departamento e outros dados pessoais do usuário. Uma nova tabela poderia ser gerada que, associada à tabela anterior, contribuiria para uma visão mais detalhada das características pessoais dos usuários do Direto.

TABELA 4.2 - Tabela do Catálogo Geral de Endereços do Direto

Usuário	Instituição	Canal de Informação
Pedro	Secretaria da Saúde	Saúde
Pedro	Secretaria da Saúde	Informática
Ipê	Secretaria da Saúde	Saúde
Roberto	Educação	Educação
Nereu	IPÊ	Saúde

Como exemplo de regras de associação geradas a partir dos canais de informação poderia-se citar:

(i) Secretaria da Saúde, Ipê → Hospitais (Suporte 15%, Confiança 70%)
(ii) Educação, Saúde→ Simpósio (Suporte 20%, Confiança 60%)

Ambas as regras poderiam sugerir que fossem criados canais específicos para os assuntos mencionados, onde poder-se-ia restringir o acesso dos usuários aos órgãos envolvidos. Neste caso, a primeira regra sugere que seja criado um canal denominado Hospital, onde os usuários do Ipê e da Secretaria da Saúde pudessem divulgar suas informações.

4.3 Mineração dos Filtros de Mensagens no Direto

Os filtros de mensagens foram propostos com o objetivo de evitar uma sobrecarga de informações na caixa postal dos usuários. Cada usuário do Direto é reponsável pela criação e manutenção dos seus filtros. O processo de filtragem de mensagens inicia com a definição das regras por parte do usuário. Este possui permissões para criação, edição, listagem e ordenamento de regras para execução. Os filtros para as mensagens de cada usuário são armazenados através de um script Sieve.

Um exemplo de Script Sieve gerado a partir das regras definidas pelo usuário é apresentado a seguir:

```
require "fileinto";
        if header :contains "from" "spam" {
          discard;
        } elsif header :contains ["subject"] ["$$$"] {
          discard;
        } else {
          fileinto "INBOX";
        }

require ["fileinto", "reject"];
```

```
if size :over 1M
{
reject text:
Por favor não me envie mensagens grandes.
Obrigado.;
stop;
}
if header :is "From" "listas@egroups.com"
{
fileinto "listas";  # move para a pasta "listas"
}

elsif address :domain :is ["From", "To"] "procergs.rs.gov.br"
{
keep; # mensagens são entregues na caixa de entrada
}
elsif anyof (not address :all :contains
["To", "Cc", "Bcc"] "spammer@spam.com",
header :matches "subject"
["*make*money*fast*", "*free*course*"])
{
fileinto "spam";   # move para a pasta "spam"
}
else
{
fileinto "pessoais";
}
```

Para que fosse possível minerar os dados do script Sieve, seria necessário interpretar o texto e gerar uma tabela capaz de descrever as ações do usuário sobre os cabeçalhos. Para o exemplo acima apresentado sugere-se a criação de uma tabela similar à Tabela 4.3:

TABELA 4.3 - Ações e Termos para os Filtros de Mensagens do Direto

Usuário	Ação	Cabeçalho	Pal-Chave	Pasta Destino
Marco	Descartar	From	Spam	
Marco	Descartar	From	Subject	
Marco	Descartar	From	$$$	
Patrícia	Rejeitar	Tamanho	1M	
Roberta	Mover	From	listas@egroups.com	Lista
Roberta	Manter	From	Procergs.rs.gov.br	Caixa de Entrada
Roberta	Manter	To	Procergs.rs.gov.br	Caixa de Entrada
Roberta	Mover	Subject	Make, money, fast, free, course	Spam

A integração com o catálogo geral de endereços do Direto também é desejável neste caso, visto que contribui para o detalhamento do perfil do usuário.

Além dos canais de informação e filtros de mensagens, está prevista a criação de listas de discussão sobre temas diversificados, permitindo que usuários de instituições distintas interajam e troquem informações.

O trabalho de mineração de dados nos canais de informação, filtros de mensagem e listas de discussão do software Direto tem como objetivo melhorar ainda mais a qualidade da informação distribuída. Através da análise das regras geradas, poderia-se oferecer novos filtros e novos canais de informação aos usuários com perfis semelhantes. Além disso, poderia-se observar os assuntos mais abordados por usuários que trabalham nas mesmas instituições, ou ainda verificar quais as informações trocadas pelos usuários de instituições distintas, através das listas de discussão. A análise destas informações contribuiria para a manutenção dos canais de informação, contribuindo para a criação de novos canais e exclusão ou maior divulgação de canais pouco utilizados.

4.4 Mineração de Dados das Listas SouJava

O trabalho proposto de mineração de dados no projeto Direto não pode ser realizado, pois, os canais de informação, os filtros de mensagens e as listas de discussão ainda não foram disponibilizadas para os usuários e não há dados para a mineração. Para contornar o problema, é proposta a mineração de dados através das três ferramentas citadas anteriormente, sobre mensagens de uma lista de discussão sobre a linguagem Java.

4.5 Etapa de Pré-processamento

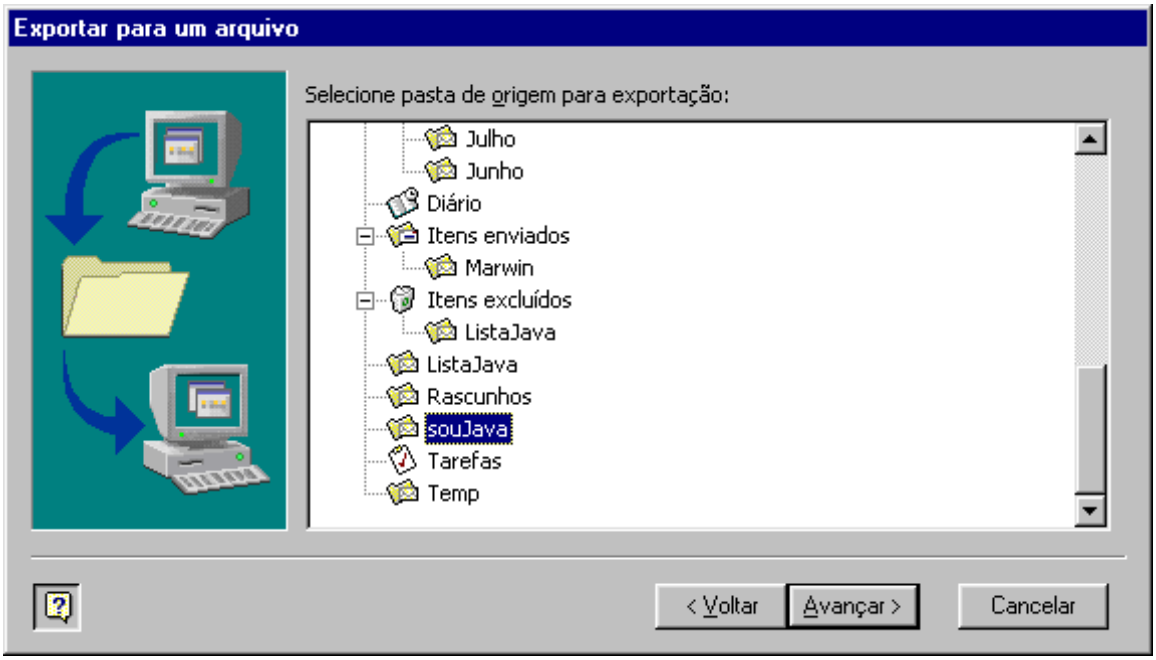
A etapa de pré-processamento pode ser resumida em três etapas. Ela inicia com a coleta dos dados, posteriormente é realizada a remoção de ruído, e finalmente os dados são consolidados de forma que possam ser utilizados nas ferramentas de mineração.

4.5.1 Coleta dos Dados

Para o estudo de caso das listas de discussão SouJava, foram utilizadas mensagens de dezembro de 2001 a setembro de 2002. A quantidade de mensagens está na ordem de 9.500 mensagens. Estas foram coletadas através de um usuário das listas de discussão SouJava e estavam disponíveis na ferramenta de correio eletrônico Netscape.

Como as mensagens precisam ser pré-processadas para três ferramentas distintas, estas foram exportadas para um banco de dados relacional do Microsoft Access, de forma que as informações ficassem centralizadas. O processo de coleta dos dados inicia com a exportação das mensagens do Netscape para o Outlook Express e posteriormente para o MS Outlook, que então habilita a opção de salvamento em arquivos externos. No caso do banco de dados Access, a exportação é feita de forma que cada pasta de mensagens no MS Outlook origina uma tabela no Access.

FIGURA 4.1 - Exportação de Mensagem para um Arquivo Externo



Durante a exportação, foi possível disponibilizar as seguintes informações relativas às mensagens: Assunto, Corpo, DeNome, DeEndereço, DeTipo, ParaNome, ParaEndereço, ParaTipo, CcNome, CCTipo, CcoNome, CcoEndereço, CcoTipo, Categorias, Importância, Informações para cobrança e Quilometragem.

4.5.2 Remoção de Ruído

Na primeira análise dos dados constatou-se que muitos dos campos que foram exportados apresentavam baixíssima incidência de informação e conseqüentemente poderiam ser eliminados. Como exemplo poderíamos citar os campos CCNome, CCTipo, CcoNome, CcoEndereço, CcoTipo, Categorias, Informações para Cobrança e Quilometragem.

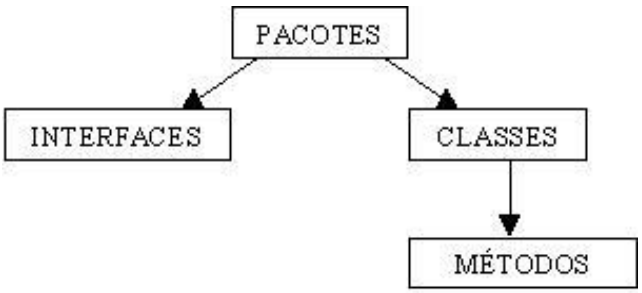
Alguns outros campos não apresentavam variações que pudessem ser consideradas como é o caso de DeTipo e ParaTipo, cujo conteúdo é sempre SMTP e o campo Importância que na grande maioria dos casos é NORMAL. Estes campos também foram eliminados.

Além dos campos já citados, foram eliminados os campos DeNome e ParaNome, que de certa forma são redundantes aos campos DeEndereço e ParaEndereço. O campo ParaEndereço também foi eliminado já que esta sendo tratada neste estudo uma única lista de discussão e o destino das mensagens é sempre o mesmo endereço: java-list@soujava.org.br.

Neste fase tem-se disponível o identificador de cada mensagem, criado no Banco Access, e os campos Assunto, Corpo e DeEndereço das mensagens. Para o pré-processamento dos campos Assunto e Corpo foi utilizado um programa em Java, cedido gentilmente por [MEL 2002], de busca de palavras-chave nos campos assunto e corpo das mensagens. As palavras-chave utilizadas foram capturadas previamente em [SUN 02], e

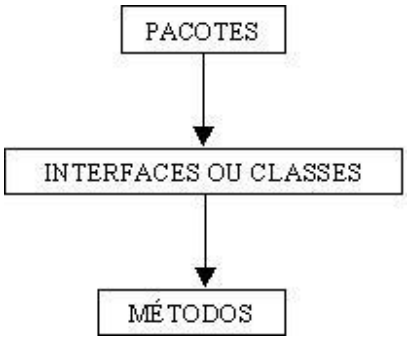
compreendem pacotes, classes, interfaces e métodos da linguagem Java. Estas palavras-chave não foram escolhidas ao acaso, tendo em vista que formam uma hierarquia:

FIGURA 4.2 - Hierarquia para as Palavras-Chave



Para a busca das palavras foi utilizada a comparação de strings e não de substrings ou radicais. O motivo para tal é que o conjunto de palavras-chave é bem definido, além de podermos encontrar palavras-chave com substrings iguais em níveis diferentes da hierarquia, como é o caso de Java.applet, que é um pacote e de Applet que é uma classe. Um problema encontrado é que algumas classes e interfaces possuem o mesmo nome, e através do mecanismo utilizado não foi possível identificar a qual grupo pertencem, por isso a hierarquia foi modificada.

FIGURA 4.3 - Hierarquia Aplicada as Palavras-chave



Em complemento à busca automática das palavras-chave nas mensagens, foi realizada uma busca por mensagens que possuem em seu corpo os strings “{“, “()”, “}”, com o objetivo de identificar as mensagens que possuem código em Java explícito. Esta busca é feita para verificar se alguma informação poderia ter sido perdida em decorrência da não utilização de substrings, pois é possível que as palavras-chave apareçam no corpo da mensagem acopladas em linhas de código, como é o caso de “JOptionPane.showMessageDialog (null, “Impossível”, “AcessoBanco.conecta”, “OptionPane.ERROR_MESSAGE”). A separação das palavras-chave no corpo destas mensagens foi feita manualmente.

A saída do programa Java de busca de palavras-chave, é um arquivo texto, contendo o identificador de cada mensagem e as pavras-chave relacionadas. Este arquivo é importado para o banco de dados Access e origina uma nova tabela chamada Palavras-Chave.

Nesta fase temos duas tabelas no banco de dados semelhantes às apresentadas abaixo:

TABELA 4.4 - Tabela Original das Listas de Discussão

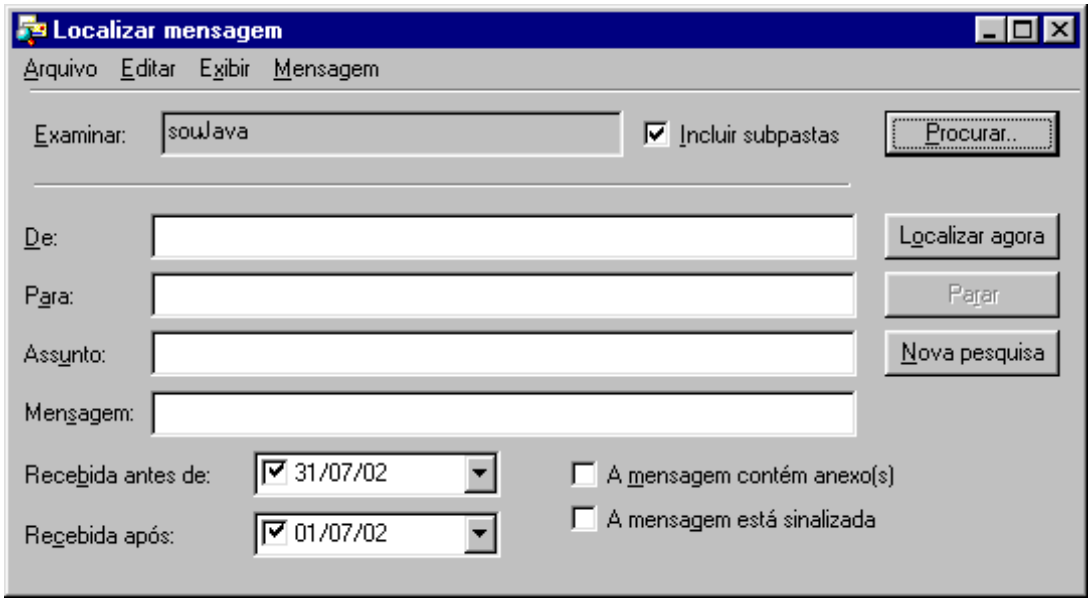
Id	DeEndereço	Assunto	Corpo
218	marcel.sakamoto@poli.usp.br	Sobre o Visual Age	Como iniciante,...
315	liandro@londrina.unimed.com.br	Conexão Persistente	SDK padrão da Sun..
420	dorivalac@ig.com.br	Email + Html	Estou com dificuldade

TABELA 4.5 - Tabela Palavras-Chave

Id	Pal-Chave1	Pal-Chave2	Pal-Chave3	Pal-ChaveN
477	List	Applet	java.awt	array()
1113	Java.io	Writer	button()	delete()

A data de envio de mensagens, que é uma informação muito valiosa, não fica habilitada durante a exportação de mensagens no MS Outlook para arquivos externos. Para contornar o problema, foi preciso voltar ao Outlook Express e utilizar um recurso de localização e posteriormente movimentação e criação de pastas para que as mensagens pudessem ser classificadas. Foi utilizado como critério, intervalos de datas, onde pastas foram criadas com as mensagens correspondentes aos meses de dezembro de 2001 a setembro de 2002. Estas pastas de mensagens foram exportadas ao MS Outlook e a partir dele originaram dez novas tabelas no banco de dados Access, cada uma contendo as mensagens de cada mês.

FIGURA 4.4 - Localização de Mensagens por Intervalo de Datas



4.5.3 Consolidação dos Dados

Durante a etapa de consolidação dos dados foi criada uma tabela integrando as informações necessárias à mineração de dados. Para tal, foi realizado um busca entre a tabela Original e Palavras-Chave pelo Msg_Id e um *look up* entre a tabela Original e as tabelas mensais através dos campos Assunto e DeEndereço. Esta última busca, dá origem a um novo campo na tabela final chamado Mês. Foram eliminados os registros que não possuíam palavras-chave, pois em todas as análises elas são consideradas.

TABELA 4.6 - Tabela Original das Listas de Discussão

Msg_Id	DeEndereço	Assunto	Corpo
218	marcel.sakamoto@poli.usp.br	Sobre o Visual Age	Como iniciante,...
315	liandro@londrina.unimed.com.br	Conexao Persistente	SDK padrão da Sun..
420	dorivalac@ig.com.br	Email + Html	Estou com dificuldade

TABELA 4.7 - Tabela Palavras-Chave

Id	Pal-Chave1	Pal-Chave2	Pal-Chave3	Pal-ChaveN
218	List	Applet	java.awt	array()
315	java.io	Writer	button()	delete()

TABELA 4.8 - Tabela Dezembro 2001

DeEndereço	Assunto
Marcel.sakamoto@poli.usp.br	Sobre o Visual Age
Liandro@londrina.unimed.com.br	Conexão Persistente
dorivalac@ig.com.br	Email + Html

TABELA 4.9 - Tabela Final

Id	Pal-Chave1	Pal-ChaveN	DeNome	Mês
218	List	array()	marcel.sakamoto@poli.usp.br	Dezembro 2001
315	java.io	delete()	liandro@londrina.unimed.com.br	Dezembro 2001

4.6 Ferramentas de Mineração

A pesquisa de dados é um processo que revela, em grandes volumes de dados, informações válidas, antes desconhecidas, de uma forma totalmente compreensível. As informações extraídas podem identificar semelhanças entre os registros do banco de dados. O resultado das pesquisas pode auxiliam o usuário a tomar decisões mais eficazes.

4.6.1 Intelligent Miner

O IBM DB2 Intelligent Miner for Data, denominado Intelligent Miner, é uma suíte de funções de pesquisas, estatísticas e de pré-processamento destinadas à análise de grandes

bancos de dados. O software oferece recursos de visualização cuja finalidade é exibir e interpretar resultados de pesquisas. O software do servidor é executado nos sistemas operacionais AIX, AS/400, OS/390, Sun Solaris e Windows NT. Os clientes podem ser AIX, OS/2 e Windows [IBM 99].

A pesquisa de dados é um processo que consiste de diversas etapas onde pode-se destacar a seleção dos dados de entrada, a sua transformação, a execução de uma função de pesquisa e a interpretação dos resultados obtidos. O Intelligent Miner oferece recursos para que as etapas citadas sejam executadas. A ferramenta oferece diversas funções que podem ser empregadas independentemente ou de forma combinada. As funções de pesquisa tem por objetivo descobrir os padrões ocultos nos dados. Depois de executadas é possível interpretar os resultados do processo de pesquisa, além de permitir que filtros sejam aplicados sobre os dados.

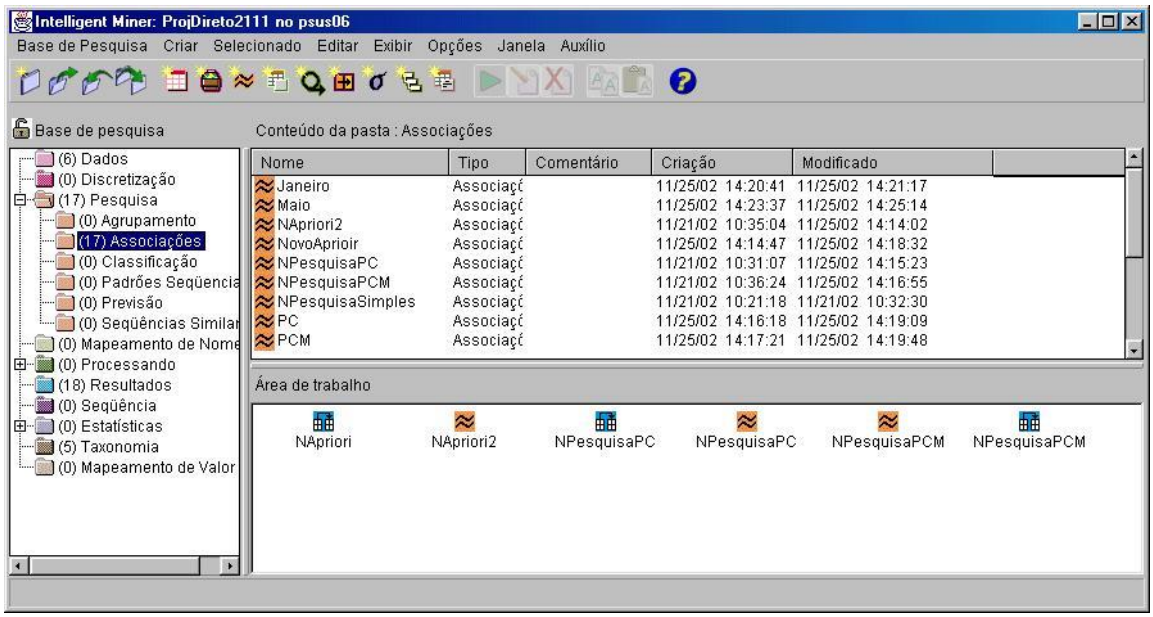
O primeiro passo da pesquisa de dados é especificar os dados de entrada a serem pesquisados e analisados. O Intelligent Miner aceita dados de tabelas de bancos de dados, *views* ou ainda em arquivos planos. Depois de especificados os dados de entrada, estes podem ser transformados por meio de funções de pré-processamento. Estas funções tais como agrupamentos e somas ajudam a melhorar a qualidade dos dados para que eles possam ser minerados de forma correta. Os dados transformados são, em seguida pesquisados por meio de uma ou mais funções de pesquisa. O Intelligent Miner possui os seguintes tipos de funções de pesquisa: associações, classificação neural, classificação em árvore, agrupamento demográfico, agrupamento neural, padrões seqüenciais, seqüências similares, previsão neural, previsão RBF (Função de Base Radial).

Os resultados do processo de pesquisa podem ser analisados através de ferramentas de visualização, que permitem a exibição dos resultados e a identificação das informações importantes reveladas pelo processo. Os resultados podem ser exportados para que possam ser exibidos em outro local. A pesquisa de dados é um processo iterativo, pois ao ver o resultado o usuário poderá refinar seus parâmetros e filtros.

A pesquisa de dados no Intelligent Miner é realizada por meio da criação de objetos inter-relacionados. Estes objetos são mostrados através de ícones e representam as definições dos dados, as funções de pré-processamento, definições de hierarquia, pesquisa e resultados que podem ser gerados através do Intelligent Miner. As definições do usuário são consideradas objetos como por exemplo a definição dos dados de entrada cria um objeto dados, a criação de uma pesquisa com determinados parâmetros cria um objeto de pesquisa e o seu resultado origina o objeto resultado.

A figura a seguir representa a janela principal do Intelligent Miner, que é descrita em [IBM 99].

FIGURA 4.5 - Janela Principal do Intelligent Miner



A janela principal pode ser descrita através de três áreas:

Repositório da Base de Pesquisa: o repositório da base de pesquisa, que fica do lado esquerdo da janela principal, apresenta os objetos armazenados na base de pesquisa. A quantidade de objetos que há em cada pasta é indicada pelo número entre parênteses que fica à frente da pasta.

Repositório de Conteúdo: O repositório de conteúdo, que fica na parte direita e superior da tela, apresenta os objetos de definições de um determinado tipo no repositório de conteúdos. Por exemplo, através do repositório de conteúdo pode-se visualizar o conteúdo da pasta Associações, que é uma subpasta de pesquisa. O conteúdo da pasta associações são os objetos de pesquisa associação definidos pelo usuário.

Área de Trabalho: A área de trabalho, que fica na parte direita inferior da tela, é a área que dá flexibilidade para que o usuário trabalhe com seus objetos com definições de tipos diferentes.

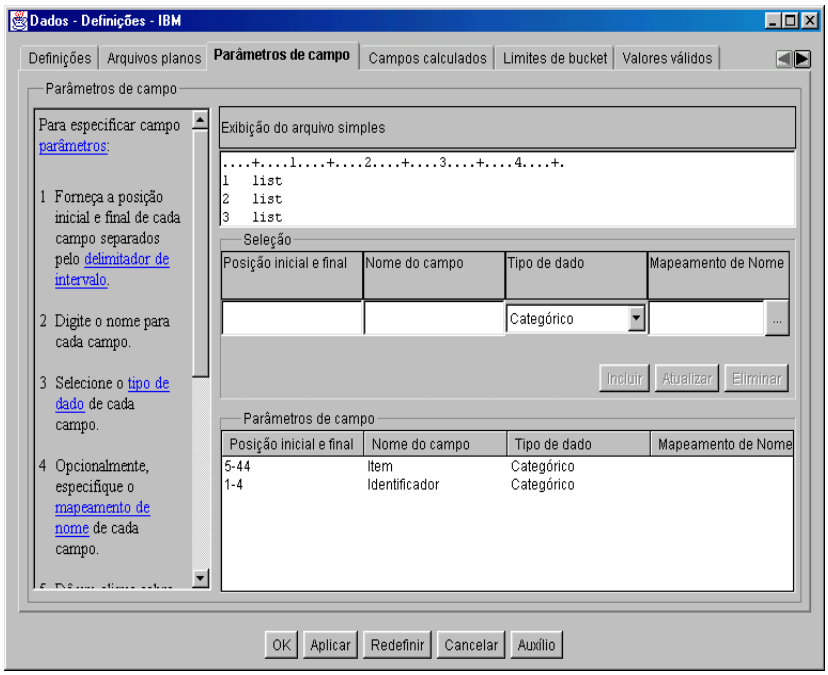
4.6.1.1 Pesquisa através de Regras de Associação no Intelligent Miner

O processo de pesquisa no Intelligent Miner inicia com a criação de uma base de pesquisa, que deverá conter todos os objetos relacionados ao estudo em questão. A base de pesquisa criada para este estudo é chamada de ProjetoDireto e contém todas as informações descritivas sobre os objetos utilizados. A base de pesquisa não contém os dados que deverão ser analisados, propriamente ditos. Estes ficam armazenados em diretórios do servidor.

Após a criação da base de pesquisa, o próximo passo é a criação de objetos de dados. Estes são as descrições lógicas dos arquivos planos ou banco de dados. Quando o

objeto de dados é criado é preciso especificar a localização dos dados e sua formatação para que possam ser utilizados de forma correta.

FIGURA 4.6 - Definição dos Dados de Entrada no Intelligent Miner



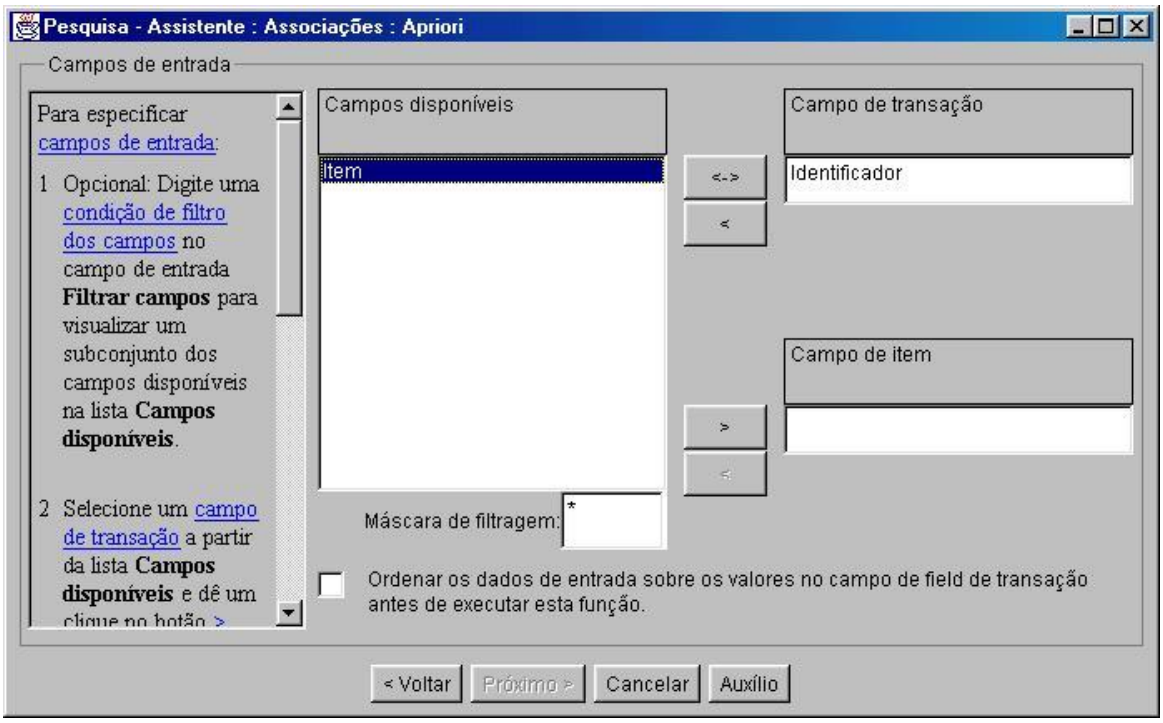
Para a pesquisa foram utilizados arquivos planos como dados de entrada e o objeto Associações que implementa regras de associação. As funções de pré-processamento disponíveis na ferramenta não foram utilizadas, pois os dados precisavam ser preparados para análise em três ferramentas distintas e foram pré-processados anteriormente. Foi realizado, entretanto uma adequação dos dados ao formato exigido pela ferramenta para Associações. Uma transação pode ser composta de diversos registros no arquivo de entrada. A tabela abaixo ilustra a afirmação:

TABELA 4.10 - Formato dos Dados de Entrada no Intelligent Miner

Identificador	Item
33	list
33	applet
37	java.awt
41	subject

A pesquisa Associações permite que se denomine um campo de transação e um campo de item para que as regras possam ser geradas. Estes dados devem ser escolhidos dentre as colunas dos dados de entrada disponíveis.

FIGURA 4.7 - Definição dos Campos para Consulta Associações



A tabela a seguir resume os parâmetros básicos e avançados da função de pesquisa Associações disponíveis no assistente de pesquisa do Intelligent Miner:

TABELA 4.11 - Parâmetros Disponíveis para Pesquisa Associações

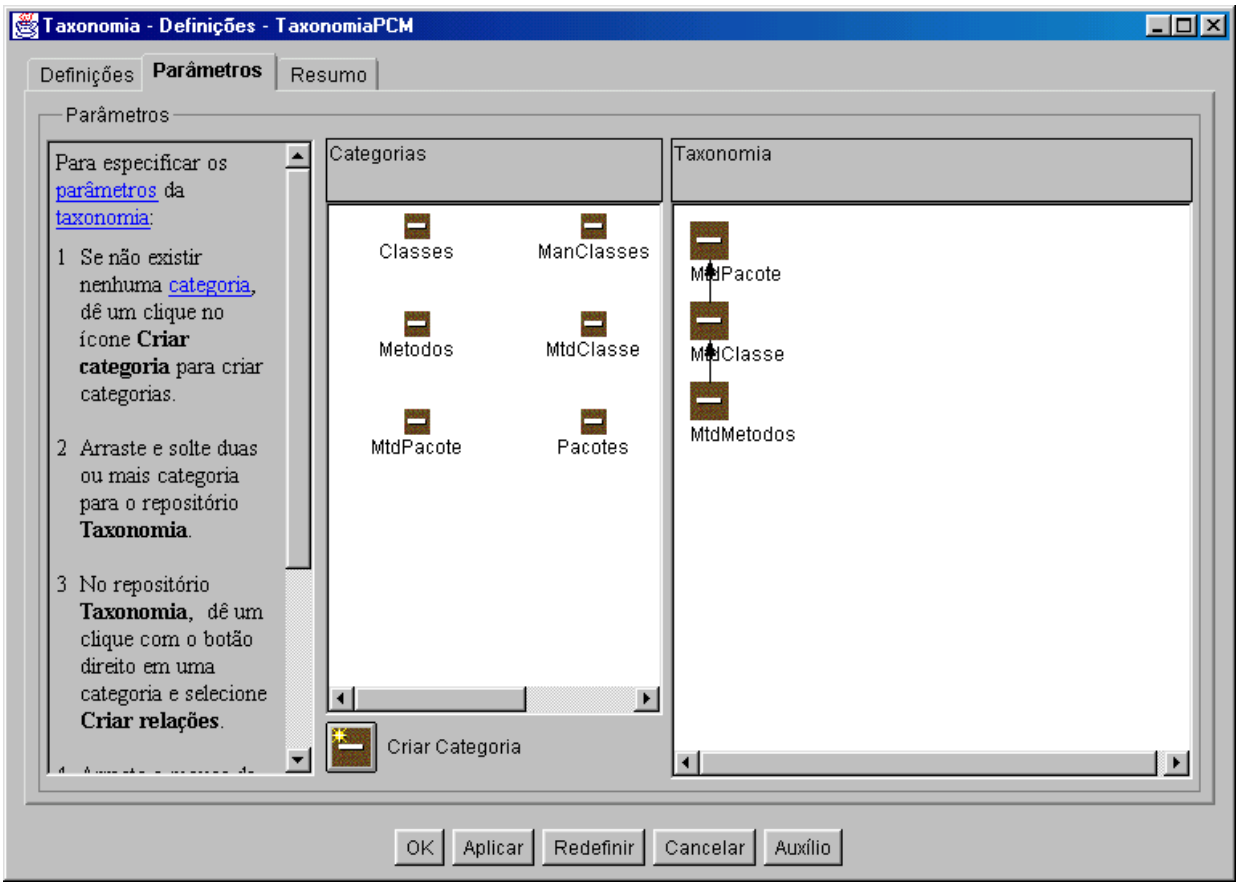
Página do Assistente	**Parâmetro**	**Valor**
Definições	Nome	Nome atribuído à consulta
	Comentário	Comentário
	Mostrar Páginas e Controles Avançados	Sim/Não
Dados de Entrada	Dados de Entrada	Tabelas disponíveis através da importação de dados
	Otimizar execução de pesquisa para:	Tempo/Espaço em Disco
	Filtrar Registros:	Expressão
	Opções de Potência	

Campos de Entrada	Campo de Transação	Disponível a partir da tabela de entrada selecionada em dados de Entrada. (utiliza-se normalmente o identificador ou a data)
	Campo de Item	Disponível a partir da tabela de entrada selecionada em dados de Entrada. (Itens)
	Ordenar os dados de entrada dos valores no campo ID de transações antes de executar esta função	Verdadeiro/Falso
Parâmetros	Suporte Mínimo	Utilizar suporte atribuido pelo sistema ou especificar
	Confiança Mínima	Utilizar confiança atribuído pelo sistema ou especificar
	Comprimento Máximo da Regra	Especificar o comprimento máximo da regra.
	Limites do item	
Parâmetros Paralelos	Executar o modo paralelo desta função	
Taxonomia	Nome da Taxonomia	Associar um objeto Taxonomia previamente definido.
Resultados	Nome dos Resultados	Atribuir um nome ao objeto resultados
	Comentários	Comentários
	Se existir resultado com esse nome, substitua-o.	Verdadeiro/ Falso

Como o principal propósito da utilização do Inteligent Miner para este trabalho é a facilidade de definir-se uma hierarquia, o processo de definição e criação de um objeto taxonomia será descrito mais detalhadamente.

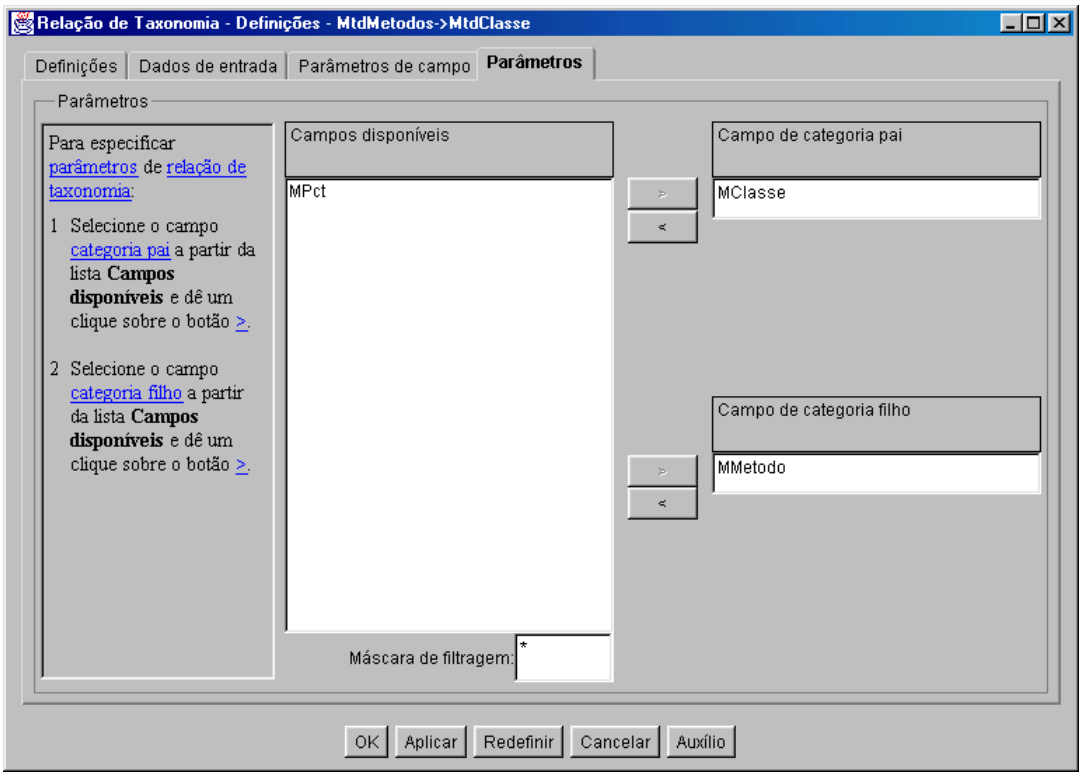
O processo de criação de uma taxonomia inicia com a criação de categorias que irão representar os nodos da árvore. A figura a seguir apresenta a tela de criação de categorias: O quadro Categorias contém as categorias definidas pelo usuário. Após a definição das mesmas, as categorias que serão utilizadas devem ser arrastadas para o quadro Taxonomia. Deve-se criar relações entre as categorias da taxonomia partindo-se sempre dos nodos filhos. Através dos *links* criados deve-se especificar as entradas de dados para a hierarquia.

FIGURA 4.8 - Definição de Taxonomia no Intelligent Miner



O Intelligent Miner permite que apenas os nodos folha contenham os itens propriamente ditos. Os níveis superiores são apenas nodos lógicos, de forma que agrupam as informações dos nodos folha por algum critério. Os dados que compõem uma hierarquia podem ser definidos manualmente ou utilizando-se uma tabela construída especialmente para este propósito. A figura abaixo apresenta a tela de definição de hierarquia utilizando um arquivo plano como entrada, que já foi previamente definido.

FIGURA 4.9 - Janela para Escolha dos Campos da Hierarquia

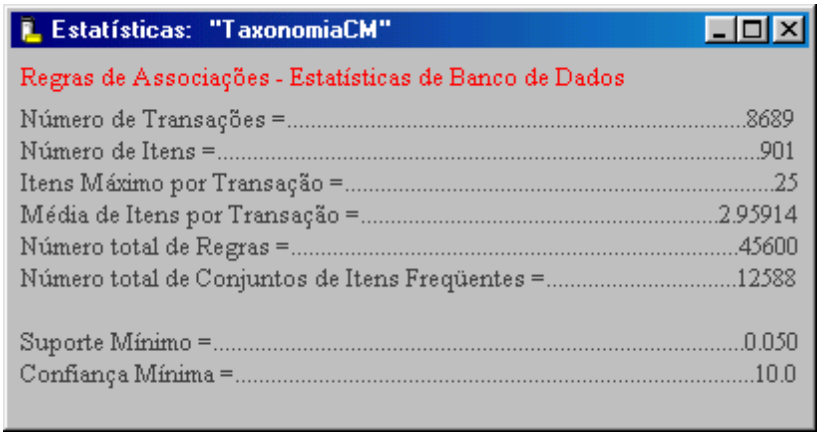


O objeto taxonomia não produz resultados. Ele deve ser associado durante a definição de uma pesquisa para produzir algum resultado.

Após a consulta Associações ser definida, ela deve ser executada e um objeto resultado correspondente é criado. O objeto resultado cria duas janelas, uma delas contém as estatísticas dos dados minerados e a outra as regras propriamente ditas.

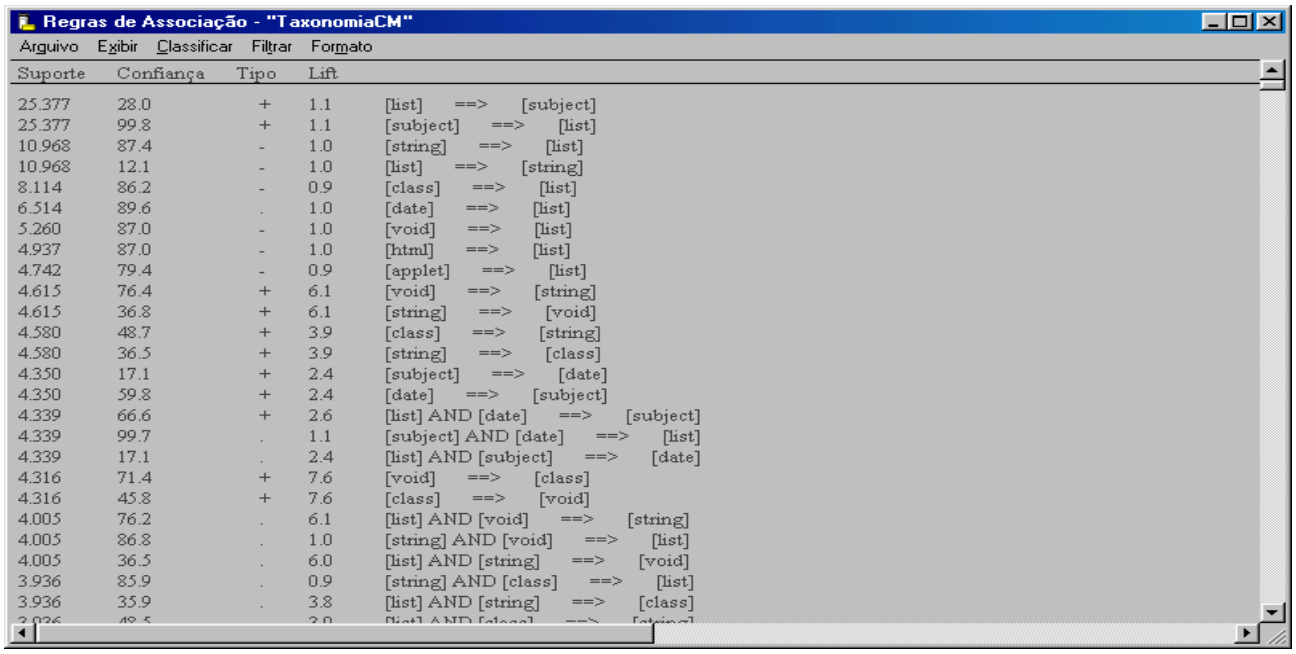
Na janela Estatísticas é possível visualizar informações como o número de transações do banco de dados, número de ítens, itens máximo por transação, média de itens por transação, número total de regras, número total de conjuntos de itens freqüentes, suporte mínimo e suporte máximo.

FIGURA 4.10 - Janela de Estatísticas no Intelligent Miner



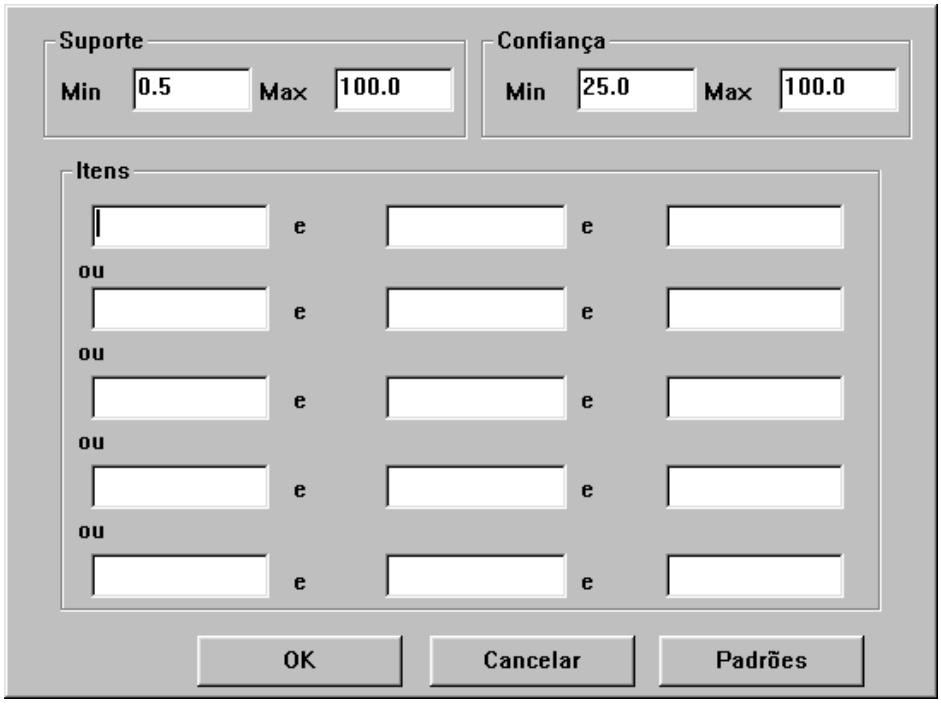
Na janela regras de associação, podemos visualizar as regras obtidas pela consulta, ordená-las de acordo com critérios como suporte, confiança e lift , além de aplicar filtros sobre os dados e parâmetros informados.

FIGURA 4.11 - Janela de Resultados do Intelligent Miner



Os critérios para filtrar regras de associação são apresentados através da Figura 4.12, onde é possível especificar intervalos para os parâmetros suporte e confiança, e informar conjuntos de itens que poderiam estar presentes nas regras.

FIGURA 4.12 - Filtros Disponíveis para as Regras de Associação



4.7 Magnus Opus

Magnum Opus é uma ferramenta desenvolvida especifcamente para mineração de regras de associação. Ela não apresenta funções para o pré-processamento dos dados, nem outros métodos de pesquisa. Como vantagem ela permite a especificação de diversas medidas para os dados e de intervalos, o que possibilita o estudo de regras quantitativas.

A ferramenta permite que se trabalhe com dois tipos de arquivos para os dados de entrada:

Identifier-Item-File (itf): os dados devem estar dispostos no formato Identificador, Item.

Exemplo de arquivo itf:

ID001, Aplet
ID001, Class
ID005, Aplet

Attribute-value data (data): os dados devem estar dispostos no formato Item 1, Item 2, Item N para cada transação. Utiliza um arquivo auxiliar denominado Mês, que possui a definição de cada coluna.

Exemplo de arquivo data:

```
1, C, f
7, C, f
12, A, t
10, B, f
```

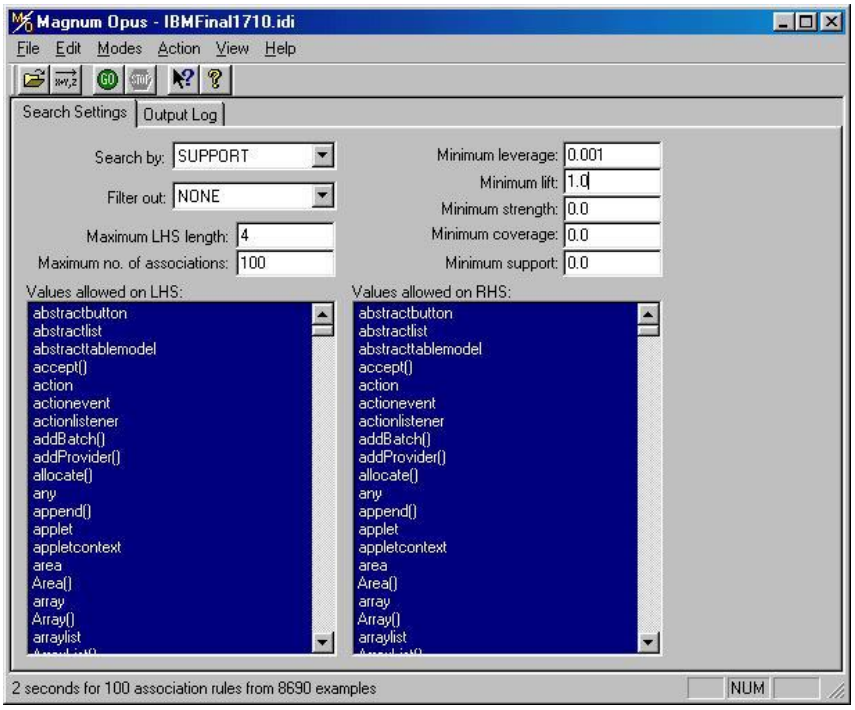
Exemplo de arquivo name:

Mês: numeric 3
GrupoSócioEconômico: categorical
Promoção: t, f

O atributo numérico foi definido como `numeric 3`, o que indica que os valores deste campo devem ser divididos em três intervalos, com aproximadamente o mesmo número de casos. Os últimos dois atributos são categóricos. O primeiro definido apenas como *categorical* pode assumir qualquer valor que esteja presente nos registros. O último assume apenas os valores listados. Se algum outro valor aparecer para a coluna, uma mensagem de erro será apresentada e as regras não são geradas.

A ferramenta é composta basicamente de duas janelas principais. Na primeira, especifica-se os critérios para a pesquisa e na segunda os dados são visualizados. Dentre os critérios de pesquisa cita-se o critério de busca, filtro, especificação de medidas, comprimento da regra e quantidade máxima de regras para serem vistas no resultado.

FIGURA 4.13 - Janela Principal do Intelligent Miner



Os parâmetros de medida que podem ser especificados pela ferramenta são *coverage*, *lift*, *strength*, *leveraege* e suporte.

Coverage: é a proporção de casos em que o antecedente de uma regra (LHS – *Left Hand Side*) aparece no total de dados. Como exemplo, supondo que exista um arquivo com 1000 casos e o antecedente está presente em 200 casos, a medida correspondente ao *coverage* seria 200/1000 = 0.2. O número total de casos de ocorrência do LHS é apresentado entre parênteses no resultado da pesquisa.

Strength: é denominação dada pelo Magnus Opus à confiança da regra. A medida indica a probabilidade que um caso satisfaça o consequente se satisfizer o antecedente. Supondo-se que o antecedente esteja presente em 200 examplos e o consequente RHS (*Right Hand Side*) em 50 exemplos dos presentes no antecedente, o *strength* da regra seria 50/200 = 0.25.

Lift: é uma medida preditiva de regras de associação determinada pela proporção entre o strengh (confiança) e os casos em que o RHS é encontrado no banco de dados.

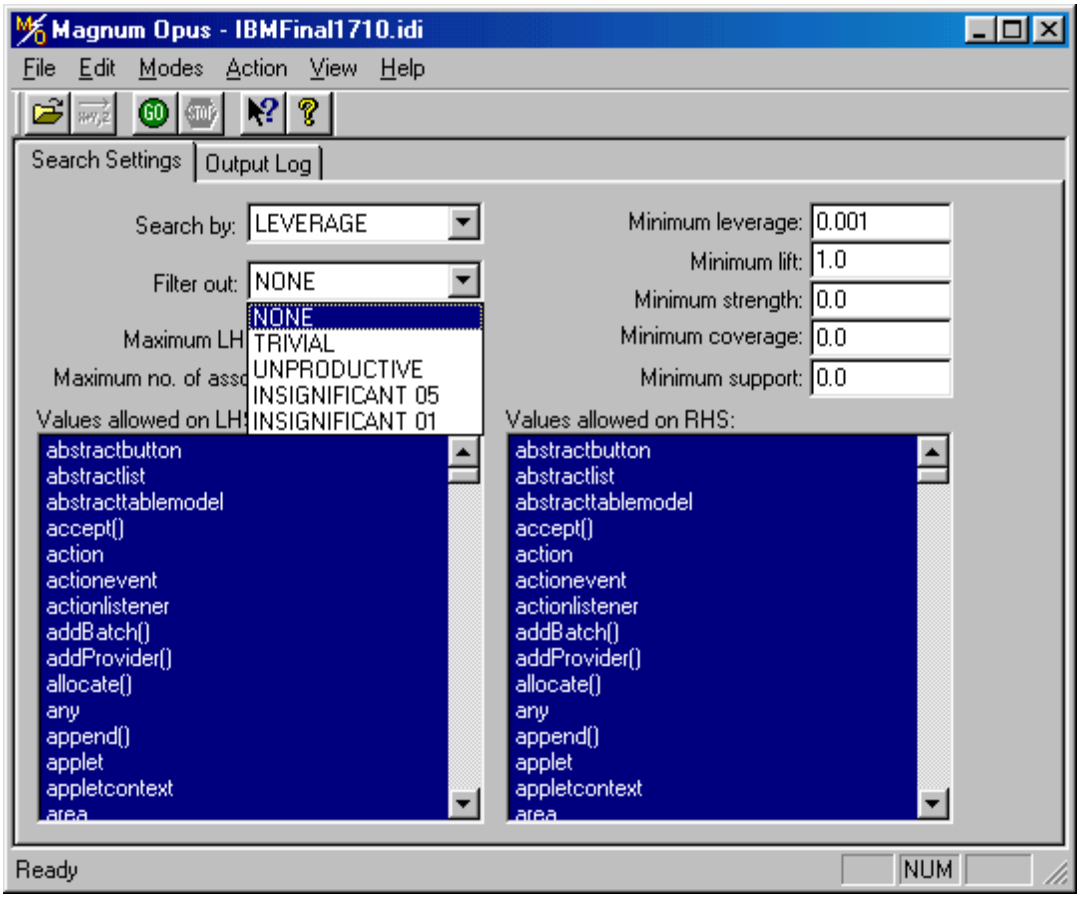
Por exemplo, supondo-se que existam 1000 casos, onde o LHS esta presente em 200 casos, o RHS está presente em 100 casos, e o RHS está presente em 50 dos 200 casos em que o LHS está. O strength seria determinado por 50/200 = 0.25. A proporção de todos os casos em que o RHS está presente seria 100/1000 = 0.1 e o *lift*, 0.25/0.1 = 2.5. Regras com

Lift negativo podem não ser muito interessantes como cita [AGR 99] através exemplo abordado na seção 2.3.1.

Leverage: é a proporção de casos adicionais que estão presentes em LHS e RHS sobre o esperado caso LHS e RHS caso estes fossem independentes um do outro. Esta medida é capaz de refletir o *strenght* e *coverage* da regra. Supondo-se que existam 1000 casos, se o LHS estivesse presente em 200 casos, o RHS em 100 e dos 200 em que o LHS estivesse presente, o RHS também estivesse, a proporção de casos em que LHS e RHS estarem presentes seria 50/1000 = 0.05. A proporção de casos esperados caso fossem independentes um do outro seria (200/1000) * (100/1000) = 0.02. O *leverage* seria 0.05 - 0.02 = 0.03. Neste exemplo, o número total de casos que ele representa seria 30.

Existe ainda uma opção denominda *Search by* que permite que seja definida qual a medida de maior relevância para a busca, dentre as citadas a cima. Durante a pesquisa de regras de associação, o Magnum Opus pode automaticamente eliminar regras de pouco interesse, caso seja utilizada a opção de filtros. Existem cinco opções de filtros: *Filter-out None*, *Filter-out Trivial*, *Filter-out Unproductive*, *Filter-out Insignificant 05* e *Filter-out Insignificant 01*.

FIGURA 4.14 - Filtros Disponíveis no Magnus Opus



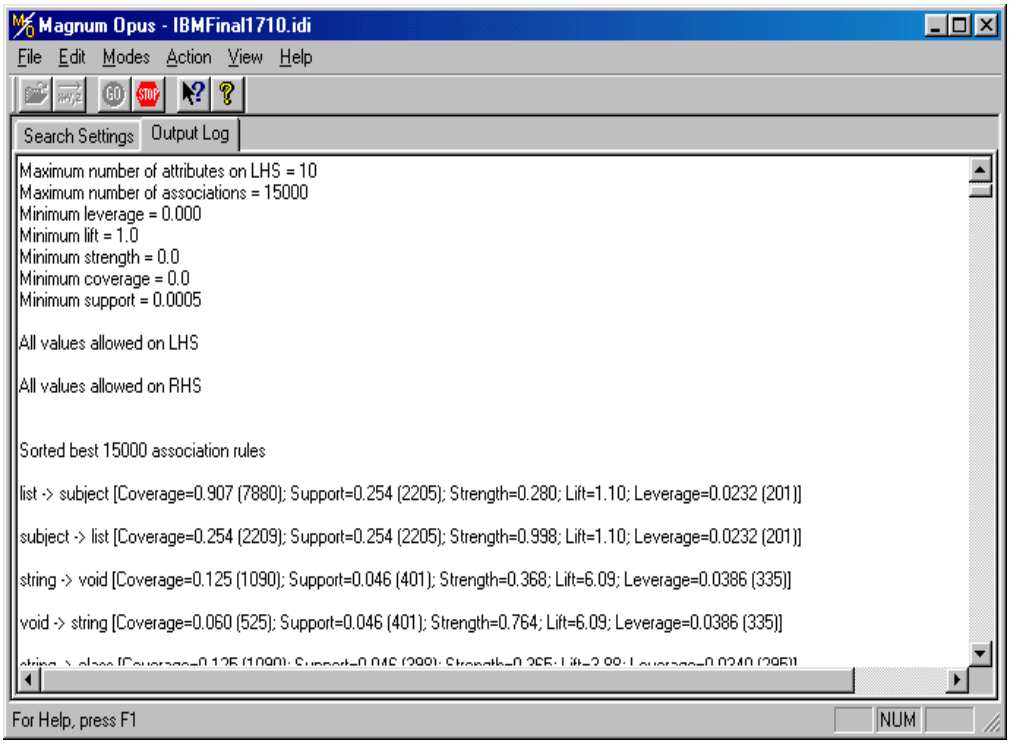
Regra Trivial: Uma regra de associação é trivial se existir uma sub-regra com o mesmo RHS que possui exatamente o mesmo número de casos (Coverage).

Regras Improdutivas: Uma regra de associação é improdutiva se existir outra associação, que seja uma subregra, com o mesmo RHS e que possua confiança igual ou superior a regra em questão. Esse filtro é similar ao *Improvment* proposto por [AGR 99], com a diferença de que não é possível estipular um valor de *Improvment* mínimo. Para regras improdutivas, o *Improvment* é sempre zero.

Regras Insignificantes: Uma associação é insignificante se existir outra associação com o mesmo conseqüente e um subconjunto do antecedente para o qual a confiança da associação formada não for significantemente maior que a confiança da primeira. Para a opção de Filter Out Insignificant 0.05 uma associação é rejeitada caso a diferenca entre o suporte ou *coveraege* das regras em questão for inferior a 0.05. Para o caso de Insignificant01, para que as regras sejam eliminadas a diferença deve ser inferior a 0.1.

Após os parâmetros e filtros serem informados a consulta é executada e as regras são geradas. Estas podem ser visulalizadas através da janela de resultados do Magnus Opus, que apresenta um cabeçalho descrevendo as medidas utilizadas e apresentando as regras. Não existem filtros ou qualquer outro tipo de facilidade para analisar as regras após terem sido geradas.

FIGURA 4.15 - Resultados Gerados pelo Magnus Opus



4.8 CBA

O CBA é uma ferramenta desenvolvida pela Universidade de Singapura, de mineração de dados através de regras de associação e classificação. A principal atração

oferecida pela ferramenta é a pesquisa através de regras de associação com suportes múltiplos para os itens.

Os dados de entrada devem estar no formato especificado pelo exemplo:

array, boolean, list

subject, html

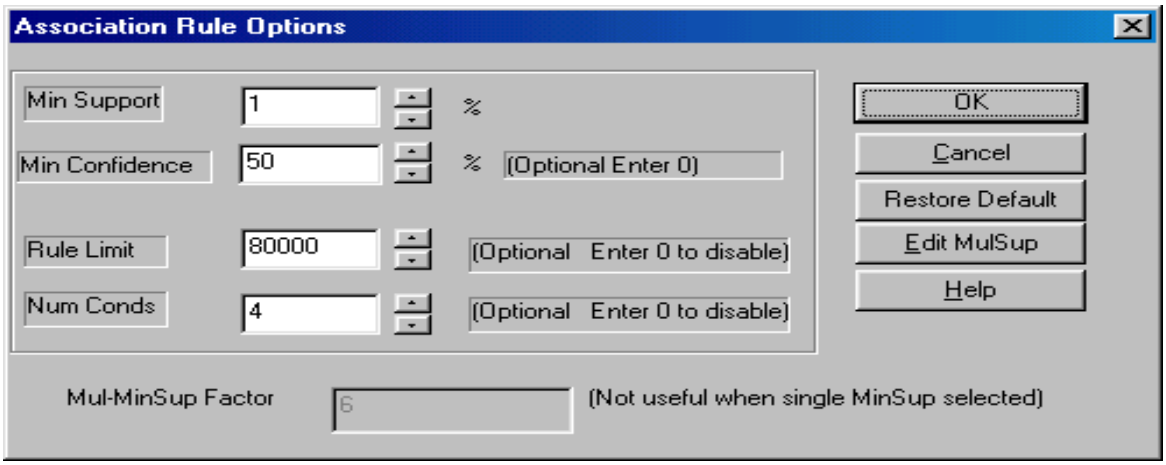
A janela principal do CBA permite que se escolha o tipo de técnica a ser utilizada. A partir da escolha de mineração por regras de associação ficam disponíveis algumas opções: a verificação da integridade do arquivo de entrada, escolha entre mineração com suportes múltiplos e suporte único, e modo de vizualização dos resultados.

FIGURA 4.16 - Janela Principal do CBA

Segundo [BIN 99], caso seja escolhida mineração de regras de associação com suporte único o algoritmo é reduzido ao algoritmo Apriori.

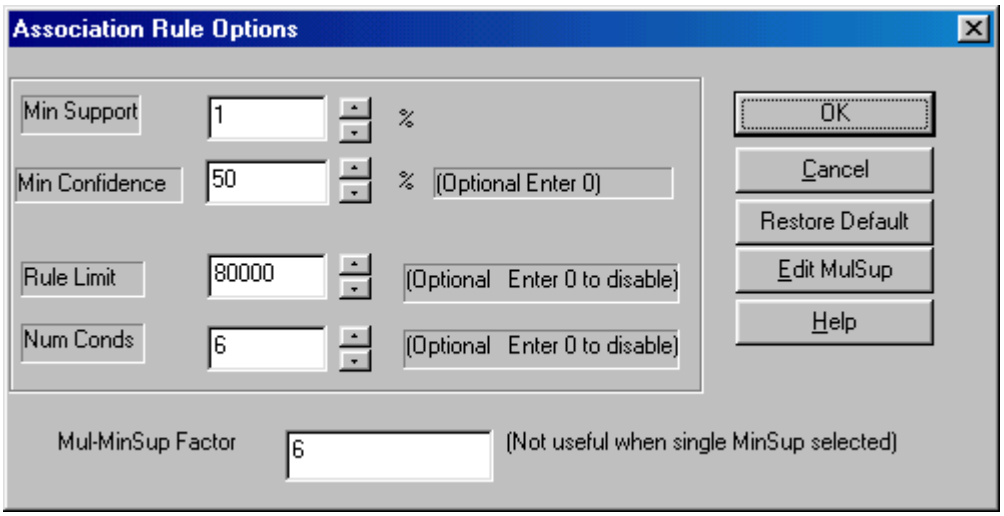
É permitido especificar alguns parâmetros para este tipo de pesquisa como suporte mínimo, confiança mínima, limite de memória e comprimento da regra.

FIGURA 4.17 - Janela de Suporte Único no CBA



Na janela de regras de associação múltiplas pode-se especificar os mesmos parâmetros utilizados para regras de associação com suporte único, além de uma medida denominada fator Mul-MinSup, onde Mul-MinSup= f * freq (item i).

FIGURA 4.18 - Janela de Suportes Múltiplos para o CBA



Em [BIN 99] é citado um exemplo para a utilização de suportes múltiplos, onde são utilizadas as seguintes fórmulas:

MIS (i) = M(i) se M(i) >LS
LS caso contrário

M(i) = f * f(i)

Onde:
f (i) = suporte (i)
f é o parâmetro informado
LS é o suporte mínimo informado

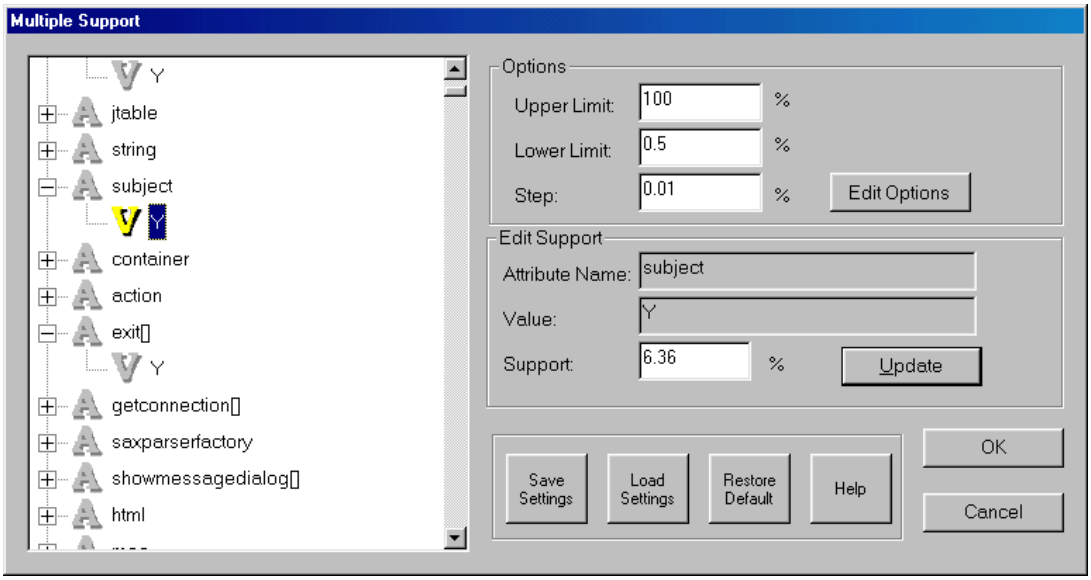
M (i) é o suporte temporário de um item
Mis (i) é o suporte final do item

Exemplo: Considerando-se três itens, 1, 2 e 3 num banco de dados, onde f (1)=1%, f (2)=3% e f (3) = 10% e LS=2% e o parâmetro para suportes multiplos f=0.3.

MIS (1) = 2%
MIS (2) = 2%
MIS (3) = 3%

O suporte individual para cada item é calculado de acordo com o *Mul MinFactor* e antes que a pesquisa seja realizada é possível editar e modificar individualmente o suporte atribuído a cada item através do botão *Edit Support*. A janela de alteração de suporte individual é apresentada através da Figura 4.19.

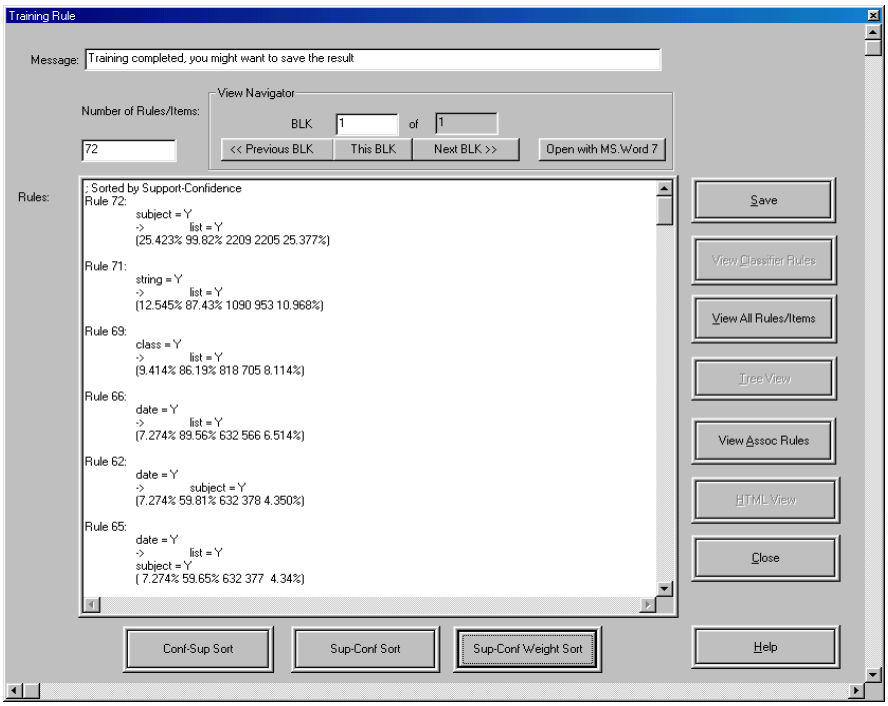
FIGURA 4.19 - Janela de Edição de Suportes Múltiplos Individuais



Após os parâmetros terem sido indicados são gerados resultados que podem ser visualizados de três maneiras: visualização das regras, visualização das regras em html e visualização das regras através de árvore.

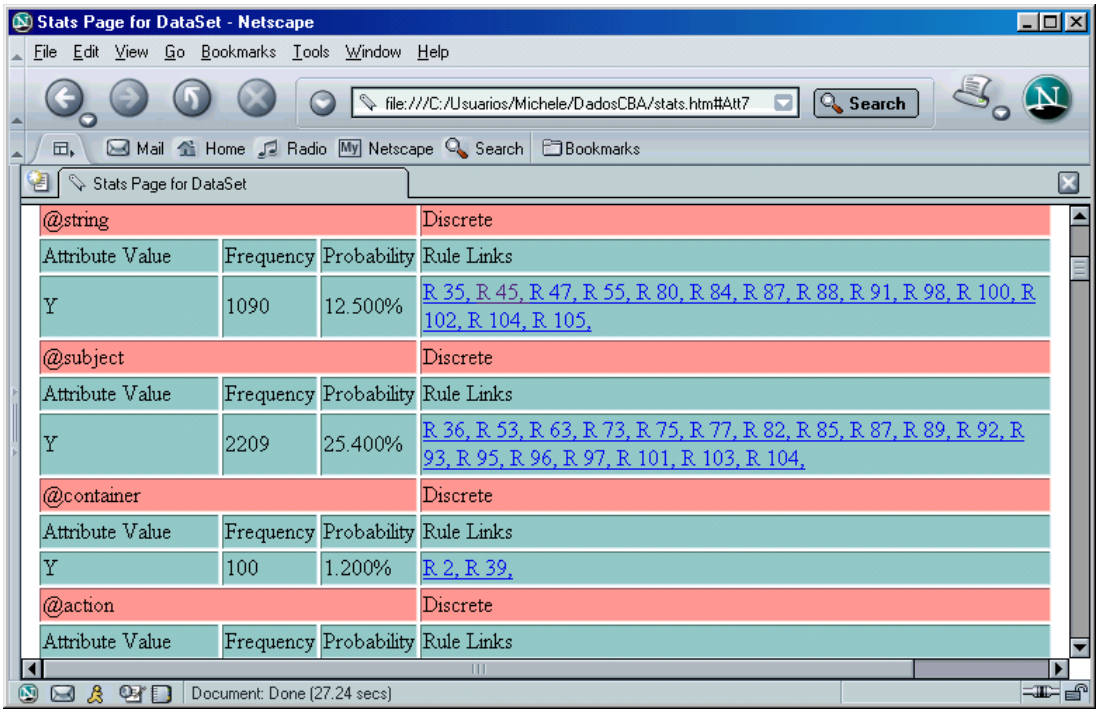
Na Figura 4.20 é apresentada a visualização das regras no formato texto, com possibilidade de ordenamento do resultado pelo suporte, confiança e uma medida que é o produto das duas primeiras medidas.

FIGURA 4.20 - Visualização das Regras Geradas no CBA no Modo Texto



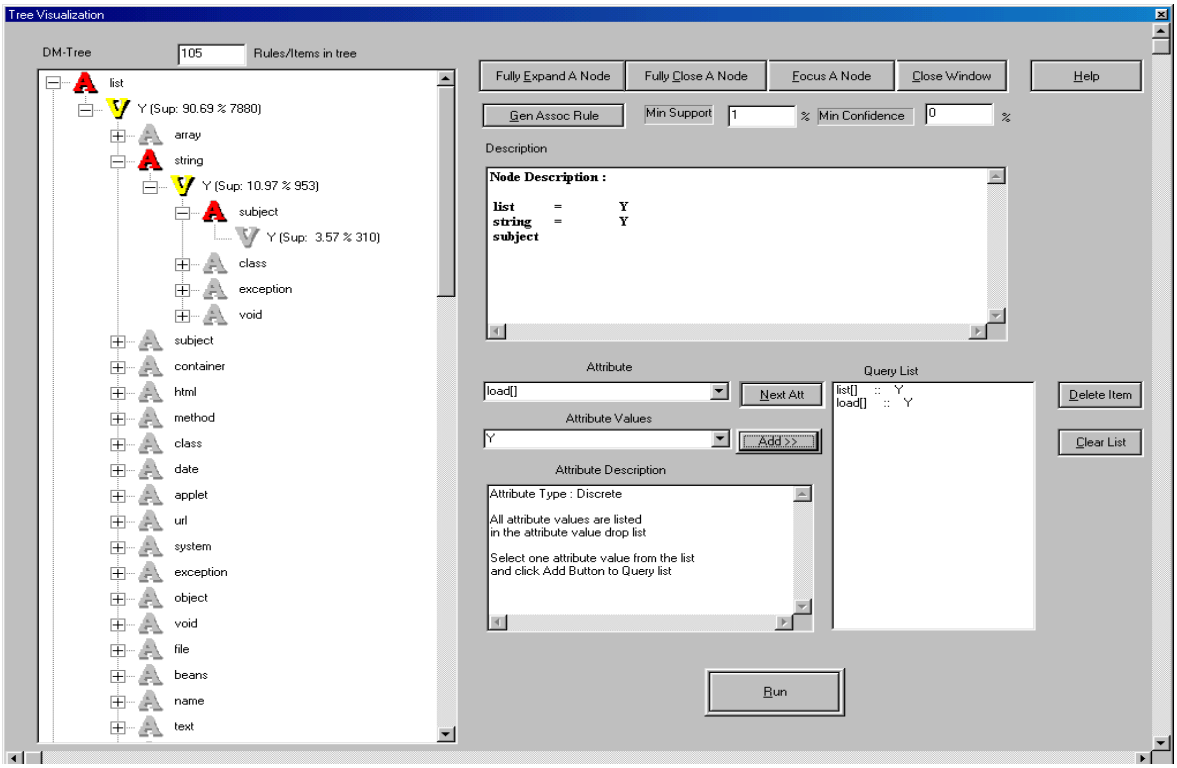
Através do formato HTML são listados os itens que fazem parte das regras descobertas juntamente com um link para as regras em que estão presentes. A visualização em HTML demosntrou ser flexivel e interessante para análise dos resultados.

FIGURA 4.21 - Visualização das Regras Geradas no CBA no Modo HTML



A visualização em árvore é apresentada na figura a seguir. Este formato é eficiente e permite que sejam aplicados filtros para a visulização de regras a partir da especificação de itens no quadro *query list.*

FIGURA 4.22 - Visualização das Regras Geradas no CBA no Modo Árvore



4.9 Ferramentas aplicadas ao caso das Listas de Discussão SouJava

4.9.1 Intelligent Miner

Para aplicação do Intelligent Miner sobre as mensagens é necessário que os dados estejam dispostos conforme a Tabela 4.11. O processo inicia com a definição de uma base de pesquisa que irá conter as definições lógicas dos dados, as pesquisas formuladas pelo usuário, as taxonomias e os resultados. Neste caso, a base de pesquisa será nomeada Projeto Direto e serão criados cinco objetos de dados: DadosIBM, DadosJaneiro, DadosMaio, HierarquiaPC e HierarquiaPCM. O primeiro objeto de dados contém os dados relativos a todas as mensagens, o segundo os dados das mensagens relativas ao mês de janeiro, o terceiro os dados relativos ao mês de maio, o quarto uma hierarquia entre pacotes e classes e o último uma hierarquia referente a pacotes, classes e métodos.

A hierarquia no Intelligent Miner cria conjuntos lógicos nos superiores aos nodos-folha. Os itens propriamente ditos só podem estar presentes no último nível da hierarquia. Serão criadas duas hierarquias para as pesquisas de palavras-chave em mensagens. Uma delas terá as classes da linguagem Java como nodos folha e os pacotes como nodos lógicos

superiores. A outra terá os métodos da linguagem Java nos nodos-folha e as classes e pacotes representando nodos lógicos.

O Intelligent Miner será aplicado sobre as mensagens diversas vezes através de pesquisas utilizando o algoritmo Apriori e variando os parâmetros de suporte e confiança. Posteriorimente serão associadas as hierarquias definidas utilizando os mesmos parâmetros de suporte e confiança. Os resultados obtidos pela aplicação da hierarquia serão comparados aos obtidos pelo simples utilização do algoritmo Apriori.

O Intelligent Miner também será utilizado para aplicar o algoritmo Apriori sobre as mensagens dos meses de janeiro e maio, com o objetivo de fazer uma análise da evolução dos assuntos abordados pelos usuários da lista SouJava.

4.9.2 Magnus Opus

Para o estudo de caso das mensagens da lista SouJava no Magnus Opus serão criados ambos os arquivos citados na seção 4.7 O arquivo *Identifier-Item-File* será utilizado para testar os filtros de regras improdutivas, insignificante e triviais e os parâmetros disponíveis na ferramenta: *Coveraege*, Suporte, *Strenght*, *Lift* e *Leveraege*. Estes parâmetros e filtros serão aplicados sobre as palavras-chaves das mensagens e os resultados obtidos serão relatados, como as regras excluídas ou a diferença no número de regras geradas utilizando-se determinado parâmetro. Para o arquivo *Attribute-value data* serão utilizadas as palavras-chave e os meses. Estes últimos estarão dispostos na forma numérica e serão utilizados para construção e teste de intervalos aplicados as mensagens.

4.9.3 CBA

O CBA será utilizado para a descoberta de regras que contenham métodos pois, eles aparecem com pouca freqüencia nas mensagens. Inicialmente serão geradas regras sobre as palavras-chave utilizando o algoritmo Apriori e posteriorimente será aplicado o parâmetro *MinFactor* diversas vezes, alterando o seu valor. Serão realizadas comparações entre os resultados produzidos pelo algoritmo Apriori e os produzidos com a aplicação do *Mul MinFactor* para os mesmos parâmetros de suporte e confiança. A análise será em termos de número de regras gerada e o conteúdo apresentado pelas regras.

4.10 Considerações Finais

Neste capítulo foi apresentada a proposta de mineração sobre filtros de mensagens e canais de informação para o projeto Direto. Como não existem mensagens públicas disponíveis no Direto, utilizou-se as mensagens de uma lista de discussão sobre a linguagem Java. Foi apresentada a etapa de pré-processamento dos dados e as ferramentas utilizadas para a mineração através de regras de associação.

5 Descrição das Pesquisas

Neste capítulo são apresentados os resultados das pesquisas com regras de associação utilizando as três ferramentas citadas: Intelligent Miner, Magnus Opus e CBA.

Os testes foram realizados sobre os dados das listas de discussão SouJava. Foram coletadas mensagens no período de dezembro de 2001 a setembro de 2002. As mensagens foram pré-processadas conforme descrito na seção 4.5 e foram encontradas 5572 mensagens interessantes que serão utilizadas durante os testes. O total de itens analizados foi 15623 e a média de itens por mensagens foi na ordem de 2,8 itens.

Foram realizados diversos experimentos, dentre eles sete se destacaram de alguma maneira e estão citados neste capítulo. A maioria dos experimentos envolve a análise das palavras-chave que aparecem nas mensagens. O campo mês foi utilizado para fazer a análise das palavras mais abordadas durante determinado período e analisar as mudanças nas regras de associação quantitativas. O campo "de endereço" não foi utilizado na análise, pois não foi possível associar os dados a uma estrutura externa, como um catálogo de endereços, que pudesse trazer alguma informação relevante diminuindo a granularidade dos dados. A maioria dos endereços de email utilizados pelos usuários da lista de discussão não são endereços de instituições. Foi cogitada a hipótese de analisar assuntos abordados por estudantes de diferentes universidades, ou ainda quais as dúvidas mais frequentes que aparecem nos diferentes segmentos.

5.1 Pesquisa 1

Na Pesquisa 1 foi utilizada a ferramenta de mineração Intelligent Miner que implementa o algoritmo Apriori. Os parâmetros utilizados são os seguintes:

Suporte Mínimo: 2%
Confiança Mínima: 25%
Foram encontradas 12 regras apresentadas na Tabela 5.1:

TABELA 5.1 - Resultado da Pesquisa 1

Sup	Conf	Lift	Tipo	Regra
2.0639	39.25	2.01	+	[exception] ==> [string]
7.1429	36.51	2.49	+	[string] ==> [class]
7.1967	36.79	3.90	+	[string] ==> [void]
7.1429	48.66	2.49	+	[class] ==> [string]
6.7301	45.84	4.87	+	[class] ==> [void]
4.9174	43.35	6.44	+	[date] ==> [system]
6.7301	71.43	4.87	+	[void] ==> [class]
7.1967	76.38	3.90	+	[void] ==> [string]
5.3841	74.81	5.10	+	[string] AND [void] ==> [class]
5.3841	80.00	4.09	+	[class] AND [void] ==> [string]
4.9174	73.07	6.44	+	[system] ==> [date]
5.3841	75.38	8.00	+	[string] AND [class] ==> [void]

Análise do Resultado: A pesquisa foi realizada com um suporte mínimo relativamente alto para os dados em questão, buscando as principais combinações de itens utilizados na linguagem Java. O resultado foi apresentado a um progamador Java que julgou as regras coerentes.

5.2 Pesquisa 2

A Pesquisa 2 foi realizada no Intelligent Miner, que implementa o algoritmo Apriori. Para este experimento foi utilizada uma hierarquia onde os nodos folha são as classes da linguagem Java e os pacotes são o nodo superior. A hierarquia citada cria um conjunto de itens lógicos aqui representados por pct_<Nome do Pacote>. Os itens Pacote (java.aplet, java.awt, etc) não deixam de estar presentes na consulta, nem os itens pertencentes às classes (appletcontext, borderlayout, etc). A hierarquia apenas oferece uma habilidade extra onde itens, que podem não estar presentes na consulta simples, poderiam estar caso pudessem ser agrupados através de um critério. A regra pct_java.awt → Pct_java.net não indicaria que as palavras pct_java.awt e pct_java.net aparecem juntas nas mesmas transações com determinado grau de suporte e confiança, mas indicaria que as classes relacionadas a java.awt aparecem nas mesmas transações em que as classes relacionadas ao pacote java.net.

A seguir é citado um exemplo da hierarquia empregada para esta consulta:

TABELA 5.2 - Hierarquia entre Pacotes e Classes

Pacote Lógico	Classe
pct_java.applet	Applet
pct_java.applet	Appletcontext
pct_java.applet	Appletstub
pct_java.awt	activeevent

Os parâmetros utilizados para o experimento são os mesmos utilizados para a Pesquisa 1:

Suporte Mínimo: 2%
Confiança Mínima: 25%

Foram encontradas 25 regras:

TABELA 5.3 - Resultado da Pesquisa 2

Suporte	Confiança	Lift	Tipo	Regra
3.6612	55.74	1.22	+	[pct_java.io] ==> [pct_java.lang]
2.2075	56.42	1.23	+	[name] ==> [pct_java.lang]
6.8198	60.13	1.32	+	[date] ==> [pct_java.lang]
2.0639	39.25	2.01	+	[exception] ==> [string]
7.1429	36.51	2.49	+	[string] ==> [class]
7.1967	36.79	3.90	+	[string] ==> [void]
7.1429	48.66	2.49	+	[class] ==> [string]
2.7100	75.12	1.64	+	[connection] ==> [pct_java.lang]
6.7301	45.84	4.87	+	[class] ==> [void]
4.9174	43.35	6.44	+	[date] ==> [system]
9.0093	95.62	2.09	+	[void] ==> [pct_java.lang]
6.7301	71.43	4.87	+	[void] ==> [class]
7.1967	76.38	3.90	+	[void] ==> [string]
5.3841	74.81	5.10	+	[string] AND [void] ==> [class]
5.3841	80.00	4.09	+	[class] AND [void] ==> [string]
4.9174	73.07	6.44	+	[system] ==> [date]
5.3841	75.38	8.00	+	[string] AND [class] ==> [void]
2.6202	41.83	0.92	-	[driver] ==> [pct_java.lang]
2.0459	41.91	0.92	-	[url] ==> [pct_java.lang]
2.2972	45.23	0.99	-	[resolver] ==> [pct_java.lang]
2.1357	48.18	1.05	-	[pct_java.beans] ==> [pct_java.lang]
2.3690	26.77	0.59	-	[html] ==> [pct_java.lang]
3.3561	29.13	0.64	-	[pct_java.awt] ==> [pct_java.lang]
2.7279	29.29	0.64	-	[pct_java.applet] ==> [pct_java.lang]
2.7279	29.29	0.64	-	[applet] ==> [pct_java.lang]

Análise do Resultado: Com a utilização da hierarquia citada foi descoberto um número maior de regras comparado ao da Pesquisa 1. Muitas das regras descobertas acrescentam conhecimento pois referenciam pacotes de classes que não são mencionadas na Pesquisa 1. Uma das regras citadas, entretanto, não acrescenta conhecimento e a ferramenta não foi capaz de eliminá-la:

A regra [applet] ==> [pct_java.lang] possui o mesmo suporte e confiança que a regra [pct_java.applet] ==> [pct_java.lang]. A primeira regra deveria ser mantida pois, apresenta conhecimento mais específico do que a segunda, que deveria ser eliminada.

5.3 Pesquisa 3

A Pesquisa 2 foi realizada no Intelligent Miner, que implementa o algoritmo Apriori. Para este experimento foi utilizada uma hierarquia com dois níveis lógicos. Os nodos folha da hierarquia criada para a Pesquisa 3 são os métodos da linguagem Java. As

classes e pacotes são os nodos lógicos superiores. A hierarquia citada cria um conjunto de itens lógicos aqui representados por pct_<Nome do Pacote> e clas_<Nome da Classe>. Os itens pacote (java.aplet, java.awt, etc) não deixam de estar presentes na consulta, nem os itens pertencentes às classes (appletcontext, borderlayout, etc). A hierarquia apenas oferece uma habilidade extra onde itens, que podem não estar presentes na consulta simples, poderiam estar caso pudessem estar agrupados através de um critério. Uma regra Pct_java.applet →clas_file significaria que os métodos da classe file aparecem nas mesmas transações dos métodos pertencentes a todas as classes do pacote java.applet com determinado grau de suporte e confiança.

A seguir é citado um exemplo da hierarquia empregada para esta consulta:

TABELA 5.4 - Hierarquia entre Pacotes, Classes e Métodos

Pacote Lógico	Classe Lógica	Métodos
pct_java.applet	clas_applet	destroy()
pct_java.applet	clas_appletstub	getdocument()
pct_java.awt	clas_activeevent	dispatch()

Os parâmetros utilizados para o experimento são os mesmos utilizados na Pesquisa 1 e 2:

Suporte Mínimo: 3%
Confiança Mínima: 25%
Foram encontradas 13 regras.

TABELA 5.5 - Resultado da Pesquisa 3

Suporte	Confiança	Lift	Tipo	Regra
2.0639	39.25	2.01	+	[exception] ==> [string]
7.1429	36.51	2.49	+	[string] ==> [class]
2.0280	43.13	2.20	+	[pct_java.lang] ==> [string]
7.1967	36.79	3.90	+	[string] ==> [void]
7.1429	48.66	2.49	+	[class] ==> [string]
6.7301	45.84	4.87	+	[class] ==> [void]
4.9174	43.35	6.44	+	[date] ==> [system]
6.7301	71.43	4.87	+	[void] ==> [class]
7.1967	76.38	3.90	+	[void] ==> [string]
5.3841	74.81	5.10	+	[string] AND [void] ==> [class]
5.3841	80.00	4.09	+	[class] AND [void] ==> [string]
4.9174	73.07	6.44	+	[system] ==> [date]
5.3841	75.38	8.00	+	[string] AND [class] ==> [void]

Análise dos Resultados: Os resultados obtidos pela Pesquisa 3 são semelhantes aos resultados da Pesquisa 1. Como os métodos aparecem com pouca frequência dentre os itens presentes nas transações, o agrupamento deste nível através da hierarquia não foi suficiente para torná-los frequentes. Outros testes foram realizados e diminuindo-se o valor do suporte, é possível obter regras que envolvem métodos agrupados através da hierarquia da Pesquisa 2, entretanto gera-se uma explosão de regras. Diferente dos pacotes, que também aparecem pouco dentre os itens do conjunto de dados, os métodos não possuem um nível inferior populoso. As classes satisfazem cerca de 80% dos itens encontrados dentre as

palavras-chave, por isso a Pesquisa 2, que possui classes como nodos-filho da hierarquia apresenta uma quantidade de regras maior que a Pesquisa 3.

5.4 Pesquisa 4:

O objetivo da análise mensal é o de acompanhar a evolução dos assuntos abordados e as peculiaridades de cada mês. As regras geradas para cada mês são diferentes entre si e diferentes das regras relativas a todo período. Nesta pesquisa foi utilizado o Inteligent Miner e foram realizados quatro testes.

a. mineração com o algoritmo Apriori para os meses de janeiro e maio;
b. mineração utilizando a hierarquia da Pesquisa 2 para os mesmos meses.

Teste1: Algoritmo Apriori
Suporte Mínimo: 2%
Confiança Mínima: 25%
Filtro Mês = Janeiro

Foram encontradas 16 regras.

TABELA 5.6 - Resultado da Pesquisa 4 para Janeiro

Suporte	Confiança	Lift	Tipo	Regra
3.8513	25.44	3.36	+	[class] ==> [void]
3.1873	25.81	5.11	+	[html] ==> [text]
2.1248	42.11	2.78	+	[text] ==> [class]
2.7888	30.88	8.61	+	[date] ==> [system]
3.8513	50.88	2.95	+	[void] ==> [string]
3.8513	50.88	3.36	+	[void] ==> [class]
2.6560	62.50	3.62	+	[object] ==> [string]
3.1873	63.16	5.11	+	[text] ==> [html]
2.1248	61.54	8.13	+	[class] AND [string] ==> [void]
2.1248	72.73	4.21	+	[method] ==> [string]
2.2576	60.71	22.86	+	[getbytes()] ==> [digest()]
2.7888	77.78	8.61	+	[system] ==> [date]
2.2576	85.00	22.86	+	[digest()] ==> [getbytes()]
2.6560	100.00	6.61	+	[jpanel] ==> [class]
2.1248	55.17	3.20	.	[class] AND [void] ==> [string]
2.1248	55.17	3.64	.	[string] AND [void] ==> [class]

Teste2: Algoritmo Apriori
Suporte Mínimo: 2%
Confiança Mínima: 25%
Filtro Mês = Maio
Foram encontradas 29 regras

TABELA 5.7 - Resultado da Pesquisa 4 para Maio

Suporte	Confiança	Lift	Tipo	Regra
20.380	41.67	1.86	+	[connection] ==> [string]
20.380	41.67	1.86	+	[rehash()] ==> [string]
72.011	32.12	2.81	+	[string] ==> [void]
89.674	40.00	2.32	+	[string] ==> [class]
21.739	31.37	6.41	+	[exception] ==> [connection]
21.739	44.44	2.58	+	[connection] ==> [class]
46.196	34.69	5.94	+	[date] ==> [system]
89.674	51.97	2.32	+	[class] ==> [string]
81.522	47.24	4.14	+	[class] ==> [void]
21.739	44.44	6.41	+	[connection] ==> [exception]
72.011	63.10	2.81	+	[void] ==> [string]
25.815	52.78	11.77	+	[connection] ==> [resultset]
25.815	52.78	18.50	+	[connection] ==> [statement]
25.815	57.58	11.77	+	[resultset] ==> [connection]
81.522	71.43	4.14	+	[void] ==> [class]
27.174	60.61	21.24	+	[resultset] ==> [statement]
46.196	79.07	5.94	+	[system] ==> [date]
25.815	90.48	18.50	+	[statement] ==> [connection]
25.815	95.00	19.42	+	[resultset] AND [statement] ==> [connection]
27.174	95.24	21.24	+	[statement] ==> [resultset]
25.815	100.00	22.30	+	[connection] AND [statement] ==> [resultset]
25.815	100.00	35.05	+	[connection] AND [resultset] ==> [statement]
20.380	29.41	1.31	.	[exception] ==> [string]
23.098	31.48	1.40	.	[driver] ==> [string]
20.380	29.41	1.70	.	[exception] ==> [class]
47.554	58.33	2.60	.	[void] AND [class] ==> [string]
47.554	53.03	4.65	.	[string] AND [class] ==> [void]
47.554	66.04	3.83	.	[string] AND [void] ==> [class]

Teste3: Algoritmo Apriori
Suporte Mínimo: 2%
Confiança Mínima: 25%
Filtro Mês = Janeiro
Hierarquia: Pesquisa 2
Foram encontradas 29 regras

TABELA 5.8 - Resultado da Pesquisa 4 para Janeiro utilizando Hierarquia de Pacote e Classe

Suporte	Confiança	Lift	Tipo	Regra
3.8513	25.44	3.36	+	[class] ==> [void]
2.1248	36.36	2.11	+	[pct_java.io] ==> [string]
3.1873	25.81	5.11	+	[html] ==> [text]
2.1248	36.36	2.40	+	[pct_java.beans] ==> [class]
3.4529	68.42	1.51	+	[text] ==> [pct_java.lang]
2.1248	42.11	2.78	+	[text] ==> [class]
2.7888	30.88	8.61	+	[date] ==> [system]
4.2497	72.73	1.61	+	[pct_java.io] ==> [pct_java.lang]
2.7888	77.78	1.72	+	[name] ==> [pct_java.lang]
3.8513	50.88	2.95	+	[void] ==> [string]
3.8513	50.88	3.36	+	[void] ==> [class]
6.3745	84.21	1.86	+	[void] ==> [pct_java.lang]
2.6560	62.50	3.62	+	[object] ==> [string]
2.7888	95.45	2.11	+	[method] ==> [pct_java.lang]
3.1873	63.16	5.11	+	[text] ==> [html]
2.1248	61.54	8.13	+	[class] AND [string] ==> [void]
2.6560	100.00	2.21	+	[jpanel] ==> [pct_java.lang]
2.1248	72.73	4.21	+	[method] ==> [string]
2.2576	60.71	22.86	+	[getbytes()] ==> [digest()]
2.7888	77.78	8.61	+	[system] ==> [date]
2.2576	85.00	22.86	+	[digest()] ==> [getbytes()]
2.6560	100.00	6.61	+	[jpanel] ==> [class]
3.4529	36.62	0.81	.	[pct_java.awt] ==> [pct_java.lang]
2.6560	45.45	1.00	.	[pct_java.beans] ==> [pct_java.lang]
4.3825	48.53	1.07	.	[date] ==> [pct_java.lang]
2.7888	48.84	1.08	.	[url] ==> [pct_java.lang]
2.3904	64.29	1.42	.	[getbytes()] ==> [pct_java.lang]
2.2576	65.38	1.44	.	[file] ==> [pct_java.lang]
2.2576	70.83	1.56	.	[gc()] ==> [pct_java.lang]
2.1248	55.17	3.20	.	[class] AND [void] ==> [string]
2.1248	55.17	3.64	.	[string] AND [void] ==> [class]
3.9841	29.41	0.65	-	[applet] ==> [pct_java.lang]
3.9841	29.41	0.65	-	[pct_java.applet] ==> [pct_java.lang]
3.7185	30.11	0.66	-	[html] ==> [pct_java.lang]

Teste4: Algoritmo Apriori
Suporte Mínimo: 2%
Confiança Mínima: 25%
Filtro Mês = Maio
Hierarquia: Pesquisa 2
Foram encontradas 79 regras

TABELA 5.9 - Resultado da Pesquisa 4 para Maio utilizando Hierarquia de Pacote e Classe

Suporte	Confiança	Lift	Tipo	Regra
20.380	41.67	1.86	+	[Rehash() ()] ==> [string]
20.380	41.67	1.86	+	[connection] ==> [string]
72.011	32.12	2.81	+	[string] ==> [void]
89.674	40.00	2.32	+	[string] ==> [class]
31.250	74.19	1.50	+	[method] ==> [pct_java.lang]
21.739	31.37	6.41	+	[exception] ==> [connection]
21.739	44.44	2.58	+	[connection] ==> [class]
46.196	34.69	5.94	+	[date] ==> [system]
29.891	78.57	1.59	+	[pct_java.beans] ==> [pct_java.lang]
89.674	51.97	2.32	+	[class] ==> [string]
81.522	47.24	4.14	+	[class] ==> [void]
42.120	86.11	1.75	+	[connection] ==> [pct_java.lang]
21.739	44.44	6.41	+	[connection] ==> [exception]
72.011	63.10	2.81	+	[void] ==> [string]
24.457	90.00	1.82	+	[array] ==> [pct_java.lang]
25.815	52.78	11.77	+	[connection] ==> [resultset]
25.815	52.78	13.87	+	[connection] ==> [pct_java.beans]
112.772	98.81	2.00	+	[void] ==> [pct_java.lang]
25.815	52.78	18.50	+	[connection] ==> [statement]
27.174	100.00	2.03	+	[resultset] AND [pct_java.beans] ==> [pct_java.lang]
28.533	100.00	2.03	+	[statement] ==> [pct_java.lang]
25.815	57.58	11.77	+	[resultset] ==> [connection]
81.522	71.43	4.14	+	[void] ==> [class]
27.174	60.61	15.93	+	[resultset] ==> [pct_java.beans]
27.174	60.61	21.24	+	[resultset] ==> [statement]
25.815	67.86	13.87	+	[pct_java.beans] ==> [connection]
46.196	79.07	5.94	+	[system] ==> [date]
27.174	71.43	15.93	+	[pct_java.beans] ==> [resultset]
25.815	86.36	17.66	+	[pct_java.beans] AND [pct_java.lang] ==> [connection]
25.815	86.36	17.66	+	[resultset] AND [pct_java.lang] ==> [connection]
25.815	100.00	2.03	.	[connection] AND [resultset] AND [statement] ==> [pct_java.lang]
25.815	100.00	2.03	.	[connection] AND [resultset] AND [pct_java.beans] ==> [pct_java.lang]
25.815	100.00	2.03	.	[connection] AND [statement] ==> [pct_java.lang]
25.815	100.00	2.03	.	[connection] AND [pct_java.beans] ==> [pct_java.lang]
25.815	61.29	13.67	.	[connection] AND [pct_java.lang] ==> [resultset]
25.815	61.29	16.11	.	[connection] AND [pct_java.lang] ==> [pct_java.beans]
25.815	61.29	21.48	.	[connection] AND [pct_java.lang] ==> [statement]
25.815	90.48	18.50	.	[statement] AND [pct_java.lang] ==> [connection]

Análise dos Resultados: Foram realizados testes que comprovam que o acompanhamento periódico apresenta mudanças nos assuntos abordados. Para o caso de uma linguagem de programação, apesar da afirmação ser verdadeira, talvez o mais correto fosse fazer uma análise de um período maior, acompanhando mudanças de versões. O caso é citado, entretanto, para fazer uma analogia aos canais de informaçãos, onde o acmpompanhamento periódico da evolução dos temas abordados é vital para criação, exclusão e divulgação dos canais.

5.5 Pesquisa 5

Para esta pesquisa foi utilizada a ferramenta CBA sobre o conjunto de dados da lista SouJava. Como a descoberta de regras de associação envolvendo métodos não está sendo uma tarefa fácil, e segundo [BIN 99] a ferramenta CBA foi desenvolvida para encontrar itens raros, a prioridade na utilização do CBA é a descoberta de regras que contenham métodos.

Para esta pesquisa foram realizados três testes:

Teste1: Pesquisa utilizando o algoritmo Apriori

Suporte Mínimo: 0. 2%
Confiança Mínima: 25%
Limite da Regra: 4 (comprimento máximo)
Resultado: Foram obitidas 396 regras de associação. Verificou-se manualmente que 14 regras possuíam algum item referente a métodos.

Teste2: Pesquisa utilizando Suportes Múltiplos. A função de suportes múltiplos é utilizada para modificar o suporte mínimo dos itens presentes nas transações. Itens com suporte abaixo do suporte mínimo estabelecido têm seu suporte aumentado de acordo com o Sup-MulFactor informado para a pesquisa.

Suporte Mínimo: 0. 2%
Confiança Mínima: 25%
Limite da Regra: 4 (comprimento máximo)
Sup-MulFactor: 6
Resultado: 258 regras de associação. Verificou-se manualmente que 14 regras possuíam algum item referente a métodos e que estas eram identicas as regras encontradas no Teste1.

Teste3: Pesquisa utilizando Suportes Múltiplos

Suporte Mínimo: 0. 2%
Confiança Mínima: 25%
Limite da Regra: 4 (comprimento máximo)
Sup-MulFactor: 12

Resultado: 337 regras de associação. Verificou-se manualmente que 14 regras possuíam algum item referente a métodos e que estas eram identicas as regras encontradas no Teste1 e Teste2.

Teste4: Pesquisa utilizando Suportes Múltiplos

Suporte Mínimo: 0. 2%
Confiança Mínima: 25%
Limite da Regra: 4 (comprimento máximo)
Sup-MulFactor: 18

Resultado: 380 regras de associação. Verificou-se manualmente que 14 regras possuíam algum item referente a métodos e que estas eram identicas as regras encontradas no Teste1, Teste2 e Teste3.

Apresentação das Regras contendo Métodos:

TABELA 5.10 - Resultado produzido pelo CBA

Coverage	Suporte	Confiança	Regra
0.377%	0.215%	57.14%	printstacktrace() → string
0.520%	0.251%	48.28%	getinstance()→ string
0.413%	0.233%	56.52%	getruntime()→ string
0.664%	0.251%	37.84%	close()→ string
0.790%	0.377%	47.73%	next()→ string
0.628%	0.305%	48.57%	newinstance() → string
0.718%	0.305%	42.50%	tostring()→string
0.395%	0.215%	54.55%	readline()→ string
0.574%	0.269%	46.88%	getbytes()→string
0.664%	0.269%	40.54%	rehash() ()→ string
0.664%	0.233%	35.14%	close()→ class
0.790%	0.215%	27.27%	next()→ class
0.628%	0.215%	34.29%	newinstance()→ class
0.377%	0.305%	80.95%	digest()→ getbytes()

Análise dos Resultados: Pelo acompanhamento da evolução dos testes notou-se que o Sup-MulFactor é utilizado para recalcular o suporte individual de cada item. Inicialmente itens que possuem suporte superior ao suporte informado sofrem um decréscimo e itens que possuem suporte inferior ao especificado sofrem um acréscimo. Para o conjuto de itens em questão, onde existem poucos itens com suporte alto, não foi possível encontrar regras diferentes envolvendo métodos as encontradas pelo algoritmo Apriori. O total de regras encontradas para os testes realizados foi menor quando é utilizado um Sup-MulFactor pequeno.

5.6 Pesquisa 6

Nesta pesquisa foram utilizados os recursos de filtros e parâmetros adicionais disponíveis na ferramenta Magnus Opus. A ferramenta não permite que seja utilizado *lift* menor do que 1. O Magnus Opus não gera regras com *lift* negativo como acontece no Intelligent Miner. *Lift* é uma medida dita preditiva que elimina as regras onde a proporção entre a confiança e a quantidade de dados do consequente da regra seja negativa.

Teste 1: Neste teste é utilizado o algoritmo Apriori com os seguintes parâmetros
Suporte Mínimo: 0.5%
Confiança Mínima: 25%
Lift: 1
Resultado: 186 regras disponíveis no anexo 1.

Teste 2: Neste teste é aplicado um filtro extra denominado Coveraege que é a proporção de casos em que o lado esquerdo de uma regra (LHS) ou antecedente aparece sobre o total de casos.
Suporte = 0,5%
Lift = 1.0
Confiança = 25%
Covereage = 0.02 (representam 111 ocorrências)

Foram encontradas 41 regras

TABELA 5.11 - Resultado da Pesquisa 6 utilizando Filtro de Coverage

Cover	(Cover)	Sup	Strength	Lift	Lever	Regras		
0,196	1090	0,072	0,368	3,91	0,0535	string	→	void
0,196	1090	0,071	0,365	2,49	0,0427	string	→	cass
0,147	818	0,071	0,487	2,49	0,0427	class	→	sring
0,147	818	0,067	0,458	4,87	0,0535	class	→	void
0,113	632	0,049	0,434	6,44	0,0415	date	→	system
0,094	525	0,072	0,764	3,91	0,0535	void	→	string
0,094	525	0,067	0,714	4,87	0,0535	void	→	class
0,072	401	0,054	0,748	5,1	0,0433	string & void	→	class
0,071	398	0,054	0,754	8	0,0471	string & class	→	void
0,067	375	0,054	0,8	4,09	0,0407	class & void	→	string
0,067	375	0,049	0,731	6,44	0,0415	system	→	date
0,053	293	0,021	0,392	2,01	0,0104	exception	→	string
0,053	293	0,02	0,372	2,53	0,0118	exception	→	class
0,053	293	0,016	0,307	3,26	0,0112	exception	→	void
0,053	293	0,016	0,3	7,44	0,0137	exception	→	thread
0,049	272	0,012	0,25	1,28	0,0027	url	→	string
0,041	231	0,016	0,377	1,93	0,0075	object	→	string
0,041	231	0,013	0,303	2,06	0,0065	object	→	class
0,04	225	0,016	0,391	7,44	0,0137	thread	→	exception
0,039	218	0,012	0,317	3,58	0,0089	name	→	html

0,039	218	0,011	0,271	1,84	0,0048	name	→	class
0,036	203	0,013	0,35	1,79	0,0056	resultset	→	string
0,036	203	0,012	0,335	22,77	0,0117	resultset	→	statement
0,036	203	0,011	0,315	8,74	0,0102	resultset	→	connection
0,036	203	0,009	0,251	1,71	0,0038	resultset	→	class
0,036	201	0,016	0,438	2,24	0,0087	connection	→	string
0,036	201	0,013	0,363	2,47	0,0078	connection	→	class
0,036	201	0,012	0,323	21,98	0,0111	connection	→	statement
0,036	201	0,011	0,318	8,74	0,0102	connection	→	resultset
0,036	201	0,011	0,303	4,85	0,0087	connection	→	driver
0,036	201	0,009	0,259	5,3	0,0076	connection	→	url
0,031	171	0,009	0,298	2,03	0,0046	error	→	class
0,031	171	0,008	0,275	5,23	0,0068	error	→	exception
0,027	153	0,013	0,458	2,34	0,0072	array	→	string
0,023	127	0,007	0,307	2,09	0,0037	package	→	class
0,022	122	0,01	0,434	2,22	0,0052	method	→	string
0,022	122	0,006	0,279	1,9	0,0029	method	→	class
0,022	122	0,006	0,262	2,78	0,0037	method	→	void
0,021	115	0,014	0,696	7,38	0,0124	string & exception	→	void
0,021	115	0,014	0,661	4,5	0,0106	string & exception	→	class

Teste 3: Neste teste é aplicado um filtro extra denominado *Leveraege* que é a diferença entre a confiança da regra e a proporção de casos presentes caso o antecedente e o consequente da regra fossem independentes um do outro. A medida é capaz de refletir a confiança e o *coveraege* da regra. Para este teste foram utilizados os seguintes parâmetros:

Lift =1.0 (A ferramenta não permite Lift < 1)
Suporte = 0.5%
Confiança =25%
Leveraege = 0.02

Foram encontradas 11 regras.

TABELA 5.12 - Resultado da Pesquisa 6 utilizando Filtro de Leveraege

Cover	Sup	Strenght	Lift	Lever	(Lever.)	Regras		
0,196	0,072	0,368	3,91	0,0535	298	string	→	void
0,196	0,071	0,365	2,49	0,0427	238	string	→	class
0,147	0,071	0,487	2,49	0,0427	238	class	→	string
0,147	0,067	0,458	4,87	0,0535	297	class	→	void
0,113	0,049	0,434	6,44	0,0415	231	date	→	system
0,094	0,072	0,764	3,91	0,0535	298	void	→	string
0,094	0,067	0,714	4,87	0,0535	297	void	→	class
0,072	0,054	0,748	5,1	0,0433	241	string & void	→	class
0,071	0,054	0,754	8	0,0471	262	string & class	→	void
0,067	0,054	0,8	4,09	0,0407	226	class & void	→	string
0,067	0,049	0,731	6,44	0,0415	231	system	→	date

Teste 4: Neste teste é alterado o valor da medida de Lift, que é uma medida dita preditiva pois elimina as regras, onde a proporção entre a confiança e a proporção dos itens do consequente não atinge o limiar estabelecido. Para este teste foram utilizados os seguintes parâmetros:

Suporte: 0.5%
Lift : 3.0
Suporte: 0.5%
Confiança: 25%

Foram encontradas 158 regras. O teste 1 utiliza os mesmos parâmetros com exceção do Lift que é igual a 1. Alterando o valor para 3 foram eliminadas 29 regras das 186 encontradas no teste 1.

TABELA 5.13 - Resultado da Pesquisa 6 utilizando Filtro de Lift

Cover	Sup	Strenght	Lift	Lever	Regra		
0,196	0,071	0,365	2,49	0,0427	string	→	class
0,147	0,071	0,487	2,49	0,0427	class	→	string
0,053	0,021	0,392	2,01	0,0104	exception	→	string
0,053	0,02	0,372	2,53	0,0118	exception	→	class
0,036	0,016	0,438	2,24	0,0087	connection	→	string
0,041	0,016	0,377	1,93	0,0075	object	→	string
0,036	0,013	0,363	2,47	0,0078	connection	→	class
0,036	0,013	0,35	1,79	0,0056	resultset	→	string
0,041	0,013	0,303	2,06	0,0065	object	→	class
0,027	0,013	0,458	2,34	0,0072	array	→	string
0,049	0,012	0,25	1,28	0,0027	url	→	string
0,039	0,011	0,271	1,84	0,0048	name	→	class
0,019	0,01	0,509	2,6	0,0061	double	→	string
0,02	0,01	0,495	2,53	0,0059	byte	→	string
0,022	0,01	0,434	2,22	0,0052	method	→	string
0,036	0,009	0,251	1,71	0,0038	resultset	→	class
0,031	0,009	0,298	2,03	0,0046	error	→	class
0,023	0,007	0,307	2,09	0,0037	package	→	class
0,016	0,007	0,425	2,9	0,0043	string & object	→	class
0,013	0,007	0,529	2,7	0,0042	class & object	→	string
0,022	0,006	0,279	1,9	0,0029	method	→	class
0,022	0,006	0,262	2,78	0,0037	method	→	void
0,014	0,006	0,405	2,76	0,0037	string & driver	→	class
0,011	0,006	0,508	2,6	0,0035	url & driver	→	string
0,019	0,005	0,278	1,89	0,0025	text	→	class
0,014	0,005	0,385	1,97	0,0026	vector	→	string
0,019	0,005	0,269	1,37	0,0014	text	→	string
0,012	0,005	0,439	2,25	0,0029	boolean	→	string
0,012	0,005	0,439	2,25	0,0029	rehash()	→	string

Teste 5: Para o teste 5 foi utilizado o filtro Trivial com os seguintes parâmetros:

Suporte: 0.5%
Confiança: 25%
Lift =1
Filter out: Trivial

Resultado: Foram encontradas 186 regras e nenhuma regra foi eliminada.

Teste 6: Para o teste 6 foi utilizado o filtro Unproductive, que é semelhante ao Improvment citado em [AGR 99], com os seguintes parâmetros:

Suporte: 0.5%
Confiança: 25%
Lift =1
Filter out: Unproductive
Resultado: Foram eliminadas 25 regras

TABELA 5.14 - Regras eliminadas através da Pesquisa 6 utilizando Filtro para Regras Improdutivas

Cover	Sup	Strenght	Lift	Lever	Regra		
0,008	0,007	0,932	6,35	0,0062	string & void & connection	→	class
0,008	0,007	0,953	4,88	0,0058	class & void & connection	→	string
0,012	0,007	0,588	3,01	0,0048	resultset & statement	→	string
0,009	0,007	0,8	21,96	0,0069	string & statement	→	resultset
0,011	0,007	0,627	3,21	0,0046	connection & resultset &	→	string
0,007	0,007	0,902	24,77	0,0064	string & connection &	→	resultset
0,013	0,006	0,457	4,85	0,0046	class & object	→	void
0,01	0,006	0,604	4,11	0,0043	object & void	→	class
0,008	0,006	0,711	3,64	0,0042	class & connection &	→	string
0,008	0,006	0,711	3,64	0,0042	class & resultset & statement	→	string
0,007	0,006	0,865	23,74	0,0055	string & class & statement	→	resultset
0,007	0,006	0,865	23,98	0,0055	string & class & statement	→	connection
0,008	0,005	0,682	4,65	0,0042	string & object & void	→	class
0,008	0,005	0,698	3,57	0,0039	class & connection & resultset	→	string
0,006	0,005	0,938	25,74	0,0052	string & class & connection &	→	resultset
0,006	0,005	0,938	25,99	0,0052	string & class & resultset &	→	connection
0,006	0,005	0,938	0,938	63,72	string & class & connection &	→	statement
0,008	0,005	0,636	0,636	43,25	string & void & connection	→	statement
0,008	0,005	0,636	0,636	17,47	string & void & connection	→	resultset
0,008	0,005	0,651	0,651	44,26	class & void & connection	→	statement
0,008	0,005	0,651	0,651	17,88	class & void & connection	→	resultset
0,005	0,005	0,933	0,933	6,36	void & connection &	→	class
0,005	0,005	0,933	0,933	4,77	void & connection &	→	string
0,005	0,005	0,933	0,933	6,36	void & connection & resultset	→	class
0,005	0,005	0,933	0,933	4,77	void & connection & resultset	→	string

Teste 7: Para o teste 7 foi utilizado o filtro Insignificant01, com os seguintes parâmetros:

Suporte: 0.5%
Confiança: 25%
Lift =1
Filter out: Insignificant01
Resultado: foram excluídos 90 registros

Teste 8: Para o teste 8 foi utilizado o filtro Insignificant05, com os seguintes parâmetros:

Suporte: 0.5%
Confiança: 25%
Lift =1
Filter out: Insignificant05
Resultado: foram excluídos 79 registros

Análise dos resultados: Neste experimento foram apresentados os parâmetros e filtros que podem ser aplicados, melhorando a qualidade das regras encontradas. Nos testes 1, 2 e 3 foram apresentadas parâmetros capazes de medir a quantidade mínima de ocorrência de itens nas regras de associação sob aspectos diferentes. Nos testes 4, 5, 6, 7 e 8 foram apresentados filtros que segundo a literatura possuem habilidades preditivas capazes de eliminar regras redundantes. Quando o objetivo de busca for bem específico, pode-se obter resultados precisos e enxutos utilizando-se a combinação de filtros e parâmetros disponíveis no Magnus Opus.

5.7 Pesquisa 7

Para os testes com regras de associação quantitativas foram utilizadas as palavras-chave e o campo mês. Na pesquisa foi utilizado o rescurso disponível no Magnus Opus de divisão de campos numéricos em intervalos. O campo mês foi utlizado na forma numérica para que a divisão pudesse ocorrer. Foram realizados diversos testes, com alteração no número de intervalos.

Teste 01:
Suporte: 0.5%
Confiança: 25%
Intervalos: Mês <= 4 e Mês ==> 4
Foram encontradas 14 regras.

TABELA 5.15 - Resultado Produzido pelo Magnus Opus utilizando Dois Intervalos para Mês

Cover	Sup	(Sup)	Strenght	Lift	Lever	Regras	
0,04	0,021	324	0,513	1,33	0,0051	Pal_Chave=date	→ Mês > 4
0,033	0,024	373	0,719	1,17	0,0035	Pal_Chave=applet	→ Mês <= 4
0,009	0,008	132	0,904	1,47	0,0027	Pal_Chave=security	→ Mês <= 4
0,07	0,03	461	0,423	1,1	0,0026	Pal_Chave=string	→ Mês > 4
0,024	0,011	179	0,477	1,24	0,0022	Pal_Chave=system	→ Mês > 4
0,052	0,034	536	0,655	1,07	0,0022	Pal_Chave=class	→ Mês <= 4
0,034	0,022	351	0,669	1,09	0,0018	Pal_Chave=void	→ Mês <= 4
0,022	0,01	162	0,464	1,2	0,0017	Pal_Chave=driver	→ Mês > 4
0,015	0,011	168	0,727	1,18	0,0017	Pal_Chave=object	→ Mês <= 4
0,032	0,021	328	0,665	1,08	0,0016	Pal_Chave=html	→ Mês <= 4
0,013	0,006	99	0,493	1,28	0,0014	Pal_Chave=connection	→ Mês > 4
0,013	0,006	93	0,458	1,19	0,0009	Pal_Chave=resultset	→ Mês > 4
0,009	0,007	104	0,703	1,14	0,0008	Pal_Chave=source	→ Mês <= 4
0,014	0,006	100	0,444	1,15	0,0008	Pal_Chave=thread	→ Mês > 4

Teste 02:
Suporte: 0.5%
Confiança: 25%
Intervalos: Mês < 3, 3 < = Mês <= 5 e Mês > 5
Foram encontradas 22 regras

TABELA 5.16 - Resultado Produzido pelo Magnus Opus utilizando Três Intervalos para Mês

Cover	Sup	(Sup)	Strenght	Lift	Lever	Regras		
0.070	0.033	513	0.471	1.01	0.0004	Pal_Chave=string	→	3 <= Mês <= 5
0.052	0.028	432	0.528	1.14	0.0033	Pal_Chave=class	→	3 <= Mês <= 5
0.070	0.019	296	0.272	1.08	0.0014	Pal_Chave=string	→	Mês > 5
0.034	0.018	285	0.543	1.17	0.0026	Pal_Chave=void	→	3 <= Mês <= 5
0.033	0.017	264	0.509	1.09	0.0014	Pal_Chave=applet	→	3 <= Mês <= 5
0.040	0.014	226	0.358	1.43	0.0043	Pal_Chave=date	→	Mês > 5
0.033	0.011	175	0.337	1.19	0.0018	Pal_Chave=applet	→	Mês < 3
0.022	0.011	167	0.479	1.03	0.0003	Pal_Chave=driver	→	3 <= Mês <= 5
0.032	0.010	160	0.325	1.14	0.0013	Pal_Chave=html	→	Mês < 3
0.019	0.010	153	0.522	1.12	0.0011	Pal_Chave=exception	→	3 <= Mês <= 5
0.034	0.010	150	0.286	1.01	0.0001	Pal_Chave=void	→	Mês < 3
0.024	0.009	136	0.363	1.45	0.0027	Pal_Chave=system	→	Mês > 5
0.012	0.008	127	0.676	1.45	0.0025	Pal_Chave=line	→	3 <= Mês <= 5
0.022	0.007	108	0.309	1.23	0.0013	Pal_Chave=driver	→	Mês > 5
0.014	0.007	108	0.495	1.07	0.0004	Pal_Chave=name	→	3 <= Mês <= 5
0.013	0.007	105	0.522	1.12	0.0007	Pal_Chave=connection	→	3 <= Mês <= 5
0.015	0.006	101	0.437	1.54	0.0023	Pal_Chave=object	→	Mês < 3
0.013	0.006	97	0.492	1.06	0.0003	Pal_Chave=file	→	3 <= Mês <= 5
0.013	0.006	96	0.473	1.02	0.0001	Pal_Chave=resultset	→	3 <= Mês <= 5
0.011	0.006	92	0.538	1.16	0.0008	Pal_Chave=error	→	3 <= Mês <= 5
0.017	0.005	84	0.309	1.09	0.0004	Pal_Chave=url	→	Mês < 3
0.018	0.005	81	0.286	1.01	0.0000	Pal_Chave=resolver	→	Mês < 3

Teste 03:

Suporte: 0.5%
Confiança: 25%
Intervalos: Mês < =2, 2 < = 4, 4 < Mês <=6, Mês > 6
Foram encontradas 15 regras

TABELA 5.17 - Resultado Produzido pelo Magnus Opus utilizando Quatro Intervalos para Mês

Cover	Sup	(Sup)	Strenght	Lift	Lever	Regra		
0.052	0.020	305	0.373	1.13	0.0023	Pal_Chave=class	→	2 < Mês <= 4
0.040	0.013	201	0.318	1.45	0.0040	Pal_Chave=date	→	Mês > 6
0.034	0.013	201	0.383	1.16	0.0018	Pal_Chave=void	→	2 < Mês <= 4
0.033	0.013	198	0.382	1.16	0.0017	Pal_Chave=applet	→	2 < Mês <= 4
0.033	0.011	175	0.337	1.19	0.0018	Pal_Chave=applet	→	Mês <= 2
0.032	0.011	168	0.341	1.03	0.0003	Pal_Chave=html	→	2 < Mês <= 4
0.032	0.010	160	0.325	1.14	0.0013	Pal_Chave=html	→	Mês <= 2
0.034	0.010	150	0.286	1.01	0.0001	Pal_Chave=void	→	Mês <= 2
0.024	0.008	132	0.352	1.60	0.0032	Pal_Chave=system	→	Mês > 6
0.019	0.007	102	0.348	1.06	0.0003	Pal_Chave=exception	→	2 < Mês <= 4
0.015	0.006	101	0.437	1.54	0.0023	Pal_Chave=object	→	Mês <= 2
0.022	0.006	94	0.269	1.23	0.0011	Pal_Chave=driver	→	Mês > 6
0.017	0.005	84	0.309	1.09	0.0004	Pal_Chave=url	→	Mês <= 2
0.014	0.005	84	0.385	1.17	0.0008	Pal_Chave=name	→	2 < Mês <= 4
0.018	0.005	81	0.286	1.01	0.0000	Pal_Chave=resolver	→	Mês <= 2

Análise dos resultados: Através do exemplo foi ilustrado o problema da divisão de intervalos. Caso existam muitos intervalos, talvez não seja possível alcançar o suporte mínimo. O Magnus Opus, de um modo geral, não pode ser bem empregado para regras quantitativas para o estudo de caso em questão. A análise de palavras-chave de forma agrupada não pode ser realizada, pois a ferramenta não apresentou flexibilidade. Cogitou-se a possibilidade de atribuir-se um código numérico para cada palavra, mas os intervalos só poderiam ser criados sobre uma determinada coluna. No caso das palavras-chave, uma determinada palavra poderia aparecer tanto na coluna Pal_Chave1 quanto na Pal_Chave5 e os resultados gerados não seriam válidos.

6 Conclusão

Neste trabalho, apresentou-se uma proposta para mineração de dados através de regras de associação sobre os dados de uma lista de discussão da linguagem Java. A proposta pode ser aplicada aos filtros de mensagens e canais de informação do software Direto para descobrir o perfil dos usuários e auxialiar na sua manutenção.

Inicialmente é apresentada uma revisão bibliográfica do processo de descoberta de conhecimento e algumas propostas para regras de associação, que representam uma método de mineração de dados. A partir do estudo realizado sobre os diferentes tipos de algoritmos para regras de associação podemos direcioná-los aos diferentes tipos de necessidades do usuário. São apresentadas as regras de associação aplicadas a bancos de dados muito densos, regras de associação taxonômicas, regras de associação quantitativas, regras de associação negativas e uma proposta de aplicação de suportes múltiplos às regras de associação. Foram também citados os trabalhos mais recentes desenvolvidos sobre o tema regras de associação pelos alunos da UFRGS.

Na etapa seguinte é apresentado o software Direto que é uma ferramenta que inclui os serviços de correio eletrônico, agenda e catálogo corporativo e que se destina a integrar todos os órgãos do governo estadual. O governo do Estado pode ser visto como uma organização distribuída capaz de gerar inúmeras informações de diversas naturezas. A divulgação das informações geradas é uma tarefa complexa, pois existe a dificuldade de localizar o público realmente interessado, sem que haja a necessidade de distribuí-la de uma forma generalizada. A popularização do correio eletrônico fez com que a quantidade de informações geradas causasse uma sobrecarga na caixa postal do usuário e como conseqüência à perda de informações ou a dificuldade para identificá-las. É apresentada uma proposta de um serviço de filtro de correio eletrônico capaz de classificar e rejeitar mensagens podendo contornar o problema da sobrecarga de informações. Além disso, é apresentada uma proposta de serviço de canais de informação com a finalidade de divulgar informações através de assinaturas de canais.

Posteriormente é definida uma proposta para mineração de dados para o projeto Direto, capaz de auxiliar na identificação dos perfis de seus usuários nos filtros de mensagens, canais de informaçãos e possíveis listas de discussão. Como os filtros de mensagens, os canais de informação e as listas de discussão ainda não foram disponibilizados aos usuários e como consequência o Direto não possui mensagens públicas, foram utilizadas para os experimentos dados obtidos a partir de uma lista de discussão sobre a linguagem Java. É descrita a etapa de pré-processamento realizada sobre a lista de discussão e as ferramentas utilizadas para os experimentos. A primeira ferramenta escolhida é o Intelligent Miner da IBM, onde é possível descobrir regras utilizando o algortimo Apriori, além de permitir a criação de uma taxonomia para os dados. A segunda ferramenta é o Magnus Opus que permite a criação de intervalos entre os itens e conseqüentemente a simulação de resultados para regras quantitativas, além de propor alguns filtros preditivos para regras de associação. A terceira ferramenta é o CBA que permite a especificação de suportes diferentes para os diversos itens, proporcionando que itens que ocorrem com pouca freqüência possam ser estudados, associados a itens que aparecem mais freqüentemente nos resultados. Neste capítulo também é apresentado como as ferramentas são aplicadas sobre as mensagens Java.

Na etapa seguinte são apresentados os resultados gerados pelo Intelligent Miner, Magnus Opus e CBA para o estudo de caso, juntamente com a análise sobre as pesquisas realizadas.

A utilização da hierarquia no Intelligent Miner, entre pacotes e classes, possibilitou de forma indireta a análise dos pacotes mais utilizados pelos usuários. Apesar da baixa ocorrência de palavras-chave do tipo pacote, por possuir um nível inferior populoso foi possivel formar regras de associação envolvendo este tipo de palavra. A descoberta envolvendo pacotes não teve sucesso em outras pesquisas que não envolviam hierarquia. Esta foi fundamental para a descoberta deste tipo de regra. No caso da segunda hierarquia proposta envolvendo pacotes, classes e métodos, o seu emprego não foi muito bem sucedido pois o último nível possuia uma proporção de dados pequena se comparada ao conjunto total. Através do Intelligent Miner foram apresentados testes que comprovam que é importante fazer um acompanhamento periódico das regras geradas. Para o caso de uma linguagem de programação, apesar de terem sido encontradas diferenças entre as regras geradas entre um mês e outro, talvez o mais correto fosse analisar um período maior, acompanhando as mudanças de versões da linguagem Java. O caso é citado, entretanto, para fazer uma analogia aos canais de informaçãos, onde o acompanhamento periódico da evolução dos temas abordados é vital para a criação, a exclusão e a divulgação dos canais.

Para o conjuto de itens das mensagens a utilização de suportes múltiplos, através do CBA, buscando itens raros não acrescentou regras diferentes envolvendo métodos as encontradas pelo algoritmo Apriori. A quantidade de regras encontradas, entretanto, foi menor. O resultado da pesquisa talvez possa ser explicado pelo fato de que a maioria dos itens do conjunto estudado possui suportes baixos.

No Magnus Opus foram testados diversos parâmetros e filtros sobre os dados das listas. Alguns pareceram muito eficientes como é o caso do parâmetro *lift*, filtro para regras Improdutivas e Insignificantes. Para os testes com regras de associação quantitativas foram utilizadas as palavras-chave e o campo mês. Utilizando variação no número de intervalos definidos sobre o campo mês e mantendo o mesmo suporte e confiança, quando o número de intervalos é aumentado percebeu-se que intervalo que referenciava os meses mais recentes não alcançou o suporte mínimo. Provalvelmente os meses referentes a este intervalo possuem um número menor de mensagens com palavras-chave, ou ainda existe uma variação maior nas palavras-chave dos assuntos abordados neste período.

As principais contribuições deste trabalho foram:

- o estudo e validação de três ferramentas que implementam regras de associação e que puderam ser aplicadas ao estudo de caso das listas de discussão;
- uma proposta para detectar perfis de usuários de listas de discussão, canais de informação e filtros de mensagens. É importante que uma coorporação acompanhe as mudanças nos serviços e produtos para poder oferecer serviços de qualidade. Particularmente no Direto a detecção de perfis de usuários vai auxiliar na manutenção e divulgação dos canais de informação e filtros de mensagens.

6.1 Trabalhos Futuros

Acompanhar as mudanças no ambiente de uma coorporação é fundamental para oferecer e manter serviços de qualidade. Como trabalho futuro propõe-se a identificação e detecção das mudanças ocorridas periodicamente nas regras geradas.

Uma sugestão poderia ser o armazenamento das regras geradas mensalmente numa estrutura de dados e aplicação da técnica de OLAP sobre estes dados, já que esta possui facilidades para a construção de dimensões temporais e criação de campos calculados.

Anexo 1

Antecedente	Consequente	Cover	Sup	Strenght	Lift	Lever
string	void	0,196	0,072	0,368	3,91	0,0535
void	string	0,094	0,072	0,764	3,91	0,0535
string	class	0,196	0,071	0,365	2,49	0,0427
class	string	0,147	0,071	0,487	2,49	0,0427
class	void	0,147	0,067	0,458	4,87	0,0535
void	class	0,094	0,067	0,714	4,87	0,0535
string & void	class	0,072	0,054	0,748	5,1	0,0433
string & class	void	0,071	0,054	0,754	8	0,0471
class & void	string	0,067	0,054	0,8	4,09	0,0407
date	system	0,113	0,049	0,434	6,44	0,0415
system	date	0,067	0,049	0,731	6,44	0,0415
exception	string	0,053	0,021	0,392	2,01	0,0104
exception	class	0,053	0,02	0,372	2,53	0,0118
exception	void	0,053	0,016	0,307	3,26	0,0112
exception	thread	0,053	0,016	0,3	7,44	0,0137
thread	exception	0,04	0,016	0,391	7,44	0,0137
connection	string	0,036	0,016	0,438	2,24	0,0087
object	string	0,041	0,016	0,377	1,93	0,0075
string & exception	void	0,021	0,014	0,696	7,38	0,0124
exception & void	string	0,016	0,014	0,889	4,54	0,0112
string & exception	class	0,021	0,014	0,661	4,5	0,0106
class & exception	string	0,02	0,014	0,697	3,56	0,0098
connection	class	0,036	0,013	0,363	2,47	0,0078
class & exception	void	0,02	0,013	0,67	7,11	0,0113
exception & void	class	0,016	0,013	0,811	5,53	0,0107
resultset	string	0,036	0,013	0,35	1,79	0,0056
object	class	0,041	0,013	0,303	2,06	0,0065
array	string	0,027	0,013	0,458	2,34	0,0072
name	html	0,039	0,012	0,317	3,58	0,0089
url	string	0,049	0,012	0,25	1,28	0,0027
resultset	statement	0,036	0,012	0,335	22,77	0,0117
statement	resultset	0,015	0,012	0,829	22,77	0,0117
connection	statement	0,036	0,012	0,323	21,98	0,0111
statement	connection	0,015	0,012	0,793	21,98	0,0111
string & exception & void	class	0,014	0,012	0,813	5,54	0,0096
string & class & exception	void	0,014	0,012	0,855	9,08	0,0104
class & exception & void	string	0,013	0,012	0,89	4,55	0,0091
resultset	connection	0,036	0,011	0,315	8,74	0,0102
connection	resultset	0,036	0,011	0,318	8,74	0,0102
connection	driver	0,036	0,011	0,303	4,85	0,0087
name	class	0,039	0,011	0,271	1,84	0,0048
text	html	0,019	0,011	0,546	6,18	0,0089
resultset & statement	connection	0,012	0,011	0,868	24,06	0,0101
connection & statement	resultset	0,012	0,011	0,908	24,92	0,0102

double	string	0,019	0,01	0,509	2,6	0,0061
byte	string	0,02	0,01	0,495	2,53	0,0059
method	string	0,022	0,01	0,434	2,22	0,0052
connection	url	0,036	0,009	0,259	5,3	0,0076
resultset	class	0,036	0,009	0,251	1,71	0,0038
error	class	0,031	0,009	0,298	2,03	0,0046
statement	class	0,015	0,009	0,61	4,15	0,0068
statement	string	0,015	0,009	0,61	3,12	0,0061
string & connection	class	0,016	0,009	0,557	3,79	0,0065
class & connection	string	0,013	0,009	0,671	3,43	0,0062
error	exception	0,031	0,008	0,275	5,23	0,0068
class & connection	statement	0,013	0,008	0,616	41,9	0,0079
class & connection	resultset	0,013	0,008	0,616	16,92	0,0076
resultset & statement	class	0,012	0,008	0,662	4,51	0,0063
connection & statement	class	0,012	0,008	0,692	4,72	0,0064
connection & resultset	class	0,011	0,008	0,703	4,79	0,0064
class & resultset	statement	0,009	0,008	0,882	59,97	0,0079
class & resultset	connection	0,009	0,008	0,882	24,46	0,0077
class & statement	resultset	0,009	0,008	0,9	24,71	0,0077
class & statement	connection	0,009	0,008	0,9	24,95	0,0078
string & connection	void	0,016	0,008	0,5	5,31	0,0064
string & object	void	0,016	0,008	0,506	5,37	0,0064
object & void	string	0,01	0,008	0,83	4,24	0,006
void & connection	string	0,008	0,008	0,957	4,89	0,0063
string & connection	url	0,016	0,008	0,489	10,01	0,0069
class & connection	void	0,013	0,008	0,589	6,25	0,0065
string & url	connection	0,012	0,008	0,632	17,53	0,0073
connection & resultset & Statement	class	0,011	0,008	0,729	4,97	0,0062
url & connection	string	0,009	0,008	0,827	4,23	0,0059
void & connection	class	0,008	0,008	0,935	6,37	0,0065
class & connection & statement	resultset	0,008	0,008	0,956	26,23	0,0074
class & resultset & statement	connection	0,008	0,008	0,956	26,49	0,0074
class & connection & resultset	statement	0,008	0,008	0,956	64,94	0,0076
string & connection	statement	0,016	0,007	0,466	31,66	0,0071
connection & statement	string	0,012	0,007	0,631	3,23	0,0051
string & statement	connection	0,009	0,007	0,82	22,74	0,007
string & class & connection	void	0,009	0,007	0,837	8,88	0,0065
internal	error	0,008	0,007	0,911	29,69	0,0071
string & void & connection	class	0,008	0,007	0,932	6,35	0,0062
class & void & connection	string	0,008	0,007	0,953	4,88	0,0058
string & connection	resultset	0,016	0,007	0,455	12,48	0,0066
string & connection	driver	0,016	0,007	0,455	7,26	0,0062
string & driver	connection	0,014	0,007	0,506	14,04	0,0067
string & resultset	statement	0,013	0,007	0,563	38,29	0,007
string & resultset	connection	0,013	0,007	0,563	15,62	0,0067
resultset & statement	string	0,012	0,007	0,588	3,01	0,0048
connection & resultset	string	0,011	0,007	0,625	3,2	0,0049

string & statement	resultset	0,009	0,007	0,8	21,96	0,0069
package	class	0,023	0,007	0,307	2,09	0,0037
stringbuffer	string	0,01	0,007	0,679	3,47	0,0049
string & object	class	0,016	0,007	0,425	2,9	0,0043
class & object	string	0,013	0,007	0,529	2,7	0,0042
connection & resultset & statement	string	0,011	0,007	0,627	3,21	0,0046
class & statement	string	0,009	0,007	0,74	3,78	0,0049
string & statement	class	0,009	0,007	0,74	5,04	0,0053
string & connection & statement	resultset	0,007	0,007	0,902	24,77	0,0064
string & resultset & statement	connection	0,007	0,007	0,925	25,65	0,0064
string & connection & resultset	statement	0,007	0,007	0,925	62,87	0,0065
statement	void	0,015	0,006	0,439	4,66	0,0051
string & resultset	class	0,013	0,006	0,507	3,45	0,0046
class & resultset	string	0,009	0,006	0,706	3,61	0,0047
method	class	0,022	0,006	0,279	1,9	0,0029
string & resultset	void	0,013	0,006	0,465	4,93	0,0047
string & statement	void	0,009	0,006	0,66	7,01	0,0051
void & resultset	string	0,007	0,006	0,892	4,56	0,0046
void & statement	string	0,006	0,006	0,917	4,69	0,0047
method	void	0,022	0,006	0,262	2,78	0,0037
string & driver	url	0,014	0,006	0,405	8,3	0,0051
string & driver	class	0,014	0,006	0,405	2,76	0,0037
class & object	void	0,013	0,006	0,457	4,85	0,0046
string & url	driver	0,012	0,006	0,471	7,51	0,005
url & driver	string	0,011	0,006	0,508	2,6	0,0035
object & void	class	0,01	0,006	0,604	4,11	0,0043
string & class & connection	statement	0,009	0,006	0,653	44,38	0,0056
string & class & connection	resultset	0,009	0,006	0,653	17,93	0,0054
class & driver	string	0,008	0,006	0,696	3,56	0,0041
class & connection & statement	string	0,008	0,006	0,711	3,64	0,0042
class & resultset & statement	string	0,008	0,006	0,711	3,64	0,0042
class & connection & resultset	string	0,008	0,006	0,711	3,64	0,0042
string & connection & statement	Class	0,007	0,006	0,78	5,32	0,0047
string & resultset & statement	Class	0,007	0,006	0,8	5,45	0,0047
string & connection & resultset	class	0,007	0,006	0,8	5,45	0,0047
string & class & statement	resultset	0,007	0,006	0,865	23,74	0,0055
string & class & statement	connection	0,007	0,006	0,865	23,98	0,0055
string & class & resultset	statement	0,006	0,006	0,889	60,41	0,0056
string & class & resultset	connection	0,006	0,006	0,889	24,65	0,0055
text	name	0,019	0,006	0,287	7,34	0,0048
string & thread	void	0,007	0,006	0,775	8,23	0,0049
void & thread	string	0,007	0,006	0,795	4,06	0,0042
stringtokenizer	string	0,006	0,006	0,886	4,53	0,0043
text	class	0,019	0,005	0,278	1,89	0,0025
vector	string	0,014	0,005	0,385	1,97	0,0026
resultset & statement	void	0,012	0,005	0,441	4,68	0,0042
connection & statement	void	0,012	0,005	0,462	4,9	0,0043

class & statement	void	0,009	0,005	0,588	6,24	0,0045
void & connection	void	0,009	0,005	0,6	6,37	0,0045
void & connection	statement	0,008	0,005	0,652	44,32	0,0053
string & object & void	resultset	0,008	0,005	0,652	17,9	0,0051
class & connection & resultset & statement	class	0,008	0,005	0,682	4,65	0,0042
string & connection & resultset & statement	string	0,008	0,005	0,698	3,57	0,0039
void & resultset	class	0,007	0,005	0,811	5,52	0,0044
void & resultset	statement	0,007	0,005	0,811	55,11	0,0053
void & resultset	connection	0,007	0,005	0,811	22,48	0,0051
string & class & object	class	0,007	0,005	0,811	5,52	0,0044
void & statement	void	0,007	0,005	0,811	8,61	0,0048
void & statement	resultset	0,006	0,005	0,833	22,88	0,0051
void & statement	connection	0,006	0,005	0,833	23,11	0,0052
string & class & connection & statement	class	0,006	0,005	0,833	5,68	0,0044
string & class & resultset & statement	resultset	0,006	0,005	0,938	25,74	0,0052
string & class & connection & resultset	connection	0,006	0,005	0,938	25,99	0,0052
class & object & void	statement	0,006	0,005	0,938	63,72	0,0053
text	string	0,006	0,005	0,938	4,79	0,0043
boolean	string	0,019	0,005	0,269	1,37	0,0014
hashtable	string	0,012	0,005	0,439	2,25	0,0029
paint	string	0,012	0,005	0,439	2,25	0,0029
connection & resultset & statement	class	0,006	0,005	0,879	5,99	0,0043
class & connection & statement	void	0,011	0,005	0,475	5,04	0,004
class & connection & resultset	void	0,008	0,005	0,622	6,61	0,0043
string & void & connection	void	0,008	0,005	0,622	6,61	0,0043
string & void & connection	statement	0,008	0,005	0,636	43,25	0,0049
class & void & connection	resultset	0,008	0,005	0,636	17,47	0,0047
class & void & connection	statement	0,008	0,005	0,651	44,26	0,0049
string & connection & statement	resultset	0,008	0,005	0,651	17,88	0,0047
string & connection & resultset	void	0,007	0,005	0,683	7,25	0,0043
string & class & statement	void	0,007	0,005	0,7	7,43	0,0043
string & void & statement	void	0,007	0,005	0,757	8,03	0,0044
string & void & statement	connection	0,006	0,005	0,848	23,53	0,0048
string & void & resultset	class	0,006	0,005	0,848	5,78	0,0042
void & connection & statement	connection	0,006	0,005	0,848	23,53	0,0048
void & connection & statement	resultset	0,005	0,005	0,933	25,62	0,0048
void & connection & statement	class	0,005	0,005	0,933	6,36	0,0042
void & resultset & statement	string	0,005	0,005	0,933	4,77	0,004
class & void & statement	connection	0,005	0,005	0,933	25,88	0,0048
class & void & statement	connection	0,005	0,005	0,933	25,88	0,0048
void & connection & resultset	string	0,005	0,005	0,933	4,77	0,004
void & connection & resultset	statement	0,005	0,005	0,933	63,43	0,0049
void & connection & resultset	class	0,005	0,005	0,933	6,36	0,0042
class & void & resultset	string	0,005	0,005	0,933	4,77	0,004

7 Referências

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Mining Association Rules between set of itens in Large Databases. In: ACM SIGMOD INTERNACIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD, 1993, Washington, D.C. **Proceedings…** [S.l.:s.n.], 1993. p.207-216.

AGRAWAL, R.; SRIKANT, R. Fast Algorithms for Mining Association Rules. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 20., 1994, Santiago. **Proceedings…** Hove: Morgan Kaufmann, 1994.

AGRAWAL, Rakesh; IMIELINSKI, Tomasz; SWAMI, Arun. **Constraint-Based Rule Mining in Large, Dense Database.** Disponível em: <http://www.ibm.almaden.com> Acesso em: 16 nov. 2001.

BECKER, K.; CARDOSO, M.; NICHELLE, C.M.; FRIGHETTO, M. Mail-By-Example: a Visual Query Interface for Email Management**.** In: INTERNATIONAL CONFERENCE ON ADVANCED VISUAL INTERFECES, 2000. **Proccedings**...Palermo: ACM Press, 2000. p.280-281.

BALINSKY, R. **Filtragem de informação no Ambiente Direto**. 2002. Dissertação (Mestrado em Ciencias da Computação) – Instituto de Informatica, UFRGS, Porto Alegre.

BERRY, Michael J.A.; LINOFF, Gordon. **Data Mining Techniques.** New York : John Wiley, 1997. 454p.

BING L.; WYNNE H.; YIMING M. Mining Association Rules with Multiple Minimum Supports. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY & DATA MINING, KDD, 1999. **Proceedings**… San Diego:[s.n.]. CA. USA.

BRADLEYS P.; GEHRKE J. Scaling Mining Algotithms to Large Database. **Communications of the ACM**. [S.l.]: ACM, v. 45, Issue 8, 2000.

BRUSSO, M. J. **O Uso de Mineração de Dados na Descoberta do Comportamento do Usuário da Web**. 2000. Dissertação de Mestrado (Mestrado em Ciências da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

Camargo, S. S. **MIRABIT :mineração de regras de associação em bancos de dados de comércio varejista de confecção**. 2002. Dissertação de Mestrado (Mestrado em Ciências da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

CBA: **Version 2.0**. Singapore: National University, 1998 –1999.

CERVO, L. V. **Implementação de uma ferramenta para descoberta de regras de associação em bancos de dados relacionais**. 1999. Projeto de Diplomação (Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

CHAN K. C. C.; WAI-HO A. An effective algorithm for mining interesting quantitative association rules. IN: ACM SYMPOSIUM ON APPLIED COMPUTING, 1997. **Proceedings**… , San Jose: [s.n.]. p. 88-90.

DIRETO. **Software de Correio, Catálogo e Agenda Corporativo**. Disponível em: <http://www.direto.org.br >. Acesso em: 30 set. 2002

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G. ; SMYTH, P. From Data Mining to Knowledge Discovery: An Overview. In: **Advances in Knowledge Discovery and Data Mining.** Menlo Park: AAAI Press, 1996. 611 p. p.11-34.

FREITAS, A.A. **Mining very large databases with parallel processing**. Boston: Kluwer Academic, 1998.

HAN, J.; FU, I. Discovery of Multiple-Level Association Rule from Large Database. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 21., 1995, Zurich. **Proceedings …** Hove: Morgan Kaufmann, 1994. San Francisco: Morgan Kauffmann, 1995.

IBM. **Utilizando o Intelligent Miner for Data:** versão 6 Realese. [S.l.:s.n.], 1999.

MAGNUS OPUS VERSION 1.3. [S.l.:s.n.]: G. I. Webb & Associates Pty, 1999 – 2001.

MELLO, L. C. **Um Assistente de Feedback para o Serviço de Filtragem do Software Direto**. 2002. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SALTON G.; MCGILL M. J. **A Vector Space Model for Automatic Indexing**, New York: Cornell University, 1975.

SAVARASE, A.; OMIECINSKI, S.; NAVATHE, S. Mining for Strong Negative Associations in a large Database of Customer Transaction, In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 1998, Florida. **Proceedings…** [S.l.:s.n.], 1998. p.494- 502.

SRIKANT, R.; AGRAWAL, R. Mining Generalized Associtions Rules. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 1995, Zurich. **Proceedings …** San Francisco: Morgan Kauffmann,1995.

SRIKANT, R.; AGRAWAL, R. Mining Quantitative Association Rules in Large Relational Table In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD, 1996, Montreal. **Proceedings…** Nova York: ACM New York, 1996.

SUN MICROSYSTEMS. **The Source for JavaTM Technology.** Disponível em: <www.java.sun.com>. Acesso: 28 set 2002.

WEBER, I. On pruning strategies for discovery of generalized and quantitative association rules. IN: PACIFIC RIM INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, PRICAI, 5., 1998, Singapore. **Proceedings …** Tarritown: Pergamon Press, 1998.