Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Algoritmo *SSDM* para a Mineração de Dados Semanticamente Similares

EDUARDO LUÍS GARCIA ESCOVAR

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

São Carlos

Setembro/2004

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

E74as

Escovar, Eduardo Luís Garcia.
Algoritmo SSDM para a mineração de dados semanticamente similares / Eduardo Luís Garcia Escovar. -- São Carlos : UFSCar, 2004.
86 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2004.

1. Banco de dados. 2. Data mining (mineração de dados). 3. Lógica nebulosa. 4. Regras de associação. 5. Similaridade. 6. Semântica. I. Título.

CDD: 005.74 (20ª)

Para minha família, especialmente meus pais, a quem devo tudo o que sou e tudo o que serei no futuro. Espero um dia retribuir ao menos metade de tudo o que vocês fazem por mim.

Para minha namorada, que me apoiou muito na reta final deste trabalho.

Você sabe o quanto é especial para mim.

Para meus amigos, que sempre me apoiaram e torceram por mim, e que tornaram mais alegre o meu dia-a-dia.

AGRADECIMENTOS

A Deus, que me permite ter a saúde e a paz necessária para aproveitar a oportunidade de aprender a cada dia mais.

Aos meus orientadores, Prof. Mauro e Prof.ª Marina, pela dedicação, paciência e pelo grande aprendizado que me proporcionaram.

Aos meus colegas do grupo de banco de dados, pelo apoio, cooperação e amizade.

À CAPES, que me auxiliou no desenvolvimento deste trabalho.

Muito obrigado a todos vocês.

ESCOVAR, E. L. G. **Algoritmo *SSDM* para a mineração de dados semanticamente similares**. 2004. Dissertação (Mestrado) – Departamento de Computação (DC), UFSCar, São Carlos, 2004.

RESUMO

Neste trabalho é apresentado o algoritmo *SSDM*, criado para permitir a mineração de dados semanticamente similares. Usando conceitos de lógica nebulosa, esse algoritmo analisa o grau de similaridade entre os itens, e o considera caso ele seja maior do que um parâmetro definido pelo usuário. Quando isso ocorre, são estabelecidas associações nebulosas entre os itens, que são expressas nas regras de associação obtidas. Assim, além das associações descobertas por algoritmos convencionais, o *SSDM* também descobre associações semânticas, e as exibe junto às demais regras obtidas. Para isso, são definidas estratégias para descobrir essas associações e para calcular o suporte e a confiança das regras onde elas aparecem.

Palavras-chave: lógica nebulosa, mineração de dados, regras de associação, similaridade, semântica.

ESCOVAR, E. L. G. ***SSDM* algorithm for semantically similar data mining**. 2004. Master Thesis – Department of Computer Science (DC), UFSCar, São Carlos, 2004.

ABSTRACT

The *SSDM* algorithm, created to allow semantically similar data mining, is presented in this work. Using fuzzy logic concepts, this algorithm analyzes the similarity grade between items, considering it if it is greater than a user-defined parameter. When this occurs, fuzzy associations between items are established, and are expressed in the association rules obtained. Therefore, besides associations discovered by conventional algorithms, *SSDM* also discovers semantic associations, showing them together with the other rules obtained. To do that, strategies are defined to discover these associations and calculate the support and the confidence of the rules where they appear.

Keywords: fuzzy logic, data mining, association rules, similarity, semantics.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1. MOTIVAÇÃO E OBJETIVOS

A mineração de dados é uma área que vem a cada dia sendo mais pesquisada, dada sua utilidade em diversas aplicações. As quantidades de dados a serem processados são cada vez maiores, assim como sua complexidade. Como conseqüência, é cada vez mais difícil descobrir conhecimento a partir desses dados, e por isso é grande o interesse em se obter um processo automatizado para tal. Várias formas de se realizar essa tarefa já foram pesquisadas, e entre as mais utilizadas está a mineração de regras de associação. Geralmente, as abordagens que propõem a realização dessa tarefa classificam os dados como sendo categóricos (textuais) ou quantitativos (numéricos).

Os algoritmos de mineração de dados categóricos analisam a base de dados em busca de itens desse tipo, e verificam a ocorrência deles ao longo de toda a base, assim como a sua associação com outros itens. Nesse processamento, os itens são tratados como simples seqüências de caracteres, sem analisar o seu significado semântico. Realizar uma análise da semântica dos dados durante a mineração deve conduzir a informações mais ricas, favorecendo a interpretação do conhecimento. Essa análise também pode ser feita sobre os metadados de tipos de dados mais complexos, como dados multimídia, por exemplo. Se esses metadados compreenderem as informações semânticas das mídias, independentemente do tipo de mídia em questão, as características assumem a forma textual e estão prontas para o processo de mineração.

Dessa forma, o objetivo deste trabalho foi a realização da mineração de dados semanticamente similares, usando conceitos de lógica nebulosa para processar e representar as similaridades semânticas encontradas entre os itens armazenados em uma base de dados. Para isso, foi necessária a definição de estratégias para se descobrir associações nebulosas entre itens similares, permitindo assim a sua consideração entre os itens a serem minerados. Para atingir este objetivo, foi criado um algoritmo que, considerando a existência dessas associações, estabelece novas formas de cálculo para o suporte e a confiança das regras a serem descobertas.

1.2. ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação está organizada da seguinte maneira:

- no capitulo 2 é apresentada a teoria básica de mineração de dados, além de um estudo sobre a mineração de regras de associação e seus algoritmos, incluindo a mineração de regras de associação generalizadas e a mineração de dados quantitativos;
- no capítulo 3 são apresentados os conceitos de lógica nebulosa utilizados nesse trabalho, tais como as teorias de conjuntos nebulosos e relações nebulosas;
- no capítulo 4 são apresentadas abordagens que envolvem o uso de conceitos da lógica nebulosa na mineração de dados;
- no capítulo 5 são apresentados trabalhos que versam sobre mineração de dados multimídia;
- no capítulo 6 é apresentado o algoritmo *SSDM*, criado neste trabalho para a mineração de dados semanticamente similares;
- no capítulo 7 é apresentado um estudo de caso, na qual o *SSDM* é usado na mineração das informações semânticas de uma aplicação multimídia;
- e finalmente, no capítulo 8, é apresentada a conclusão deste trabalho, com seus resultados, contribuições e trabalhos futuros.

2. MINERAÇÃO DE DADOS

2.1. INTRODUÇÃO

De acordo com [ELMASRI; NAVATHE, 2004], *mineração de dados* é um termo que se refere à mineração ou descoberta de novas informações em uma grande quantidade de dados, usando padrões ou regras. As novas informações podem ser úteis para guiar decisões sobre atividades futuras, e para ser útil na prática, a mineração de dados deve ser realizada eficientemente em grandes arquivos ou bancos de dados.

A revolução digital proporcionou novas técnicas de captura, processamento, armazenamento, distribuição e transmissão de informações digitalizadas. Como conseqüência, bancos de dados cada vez maiores vêm sendo criados. Além disso, os dados não são mais restritos somente a tuplas de representação numérica ou de caracteres. A tecnologia avançada de gerenciamento de bancos de dados é capaz de integrar diferentes tipos de dados, como imagem, vídeo e texto, por exemplo. A descoberta de conhecimento a partir desse enorme volume de dados é, portanto, um desafio [MITRA; ACHARYA, 2003]. A mineração de dados, que tem como propósito enfrentar esse desafio, é considerada uma área fortemente interdisciplinar, já que agrega conceitos e técnicas de outras áreas para viabilizar a sua aplicação [FAYYAD;PIATETSKY-SHAPIRO; SMYTH, 1997], [MARCHIORI, 2000]. Entre elas, pode-se destacar áreas como estatística, aprendizado de máquina, redes neurais e algoritmos genéticos [DEOGUN *et al.*, 1997]. Além disso, a mineração de dados pode ser relacionada a uma área mais ampla, conhecida como *descoberta de conhecimento em bancos de dados* (*Knowledge Discovery in Databases – KDD)*. Segundo [MITRA; ACHARYA, 2003], *KDD* é o processo não trivial que identifica padrões válidos, novos, potencialmente úteis e compreensíveis nos dados. O processo como um todo consiste em fazer com que dados em baixo nível tornem-se conhecimento em alto nível, e pode ser dividido em passos [HAN; KAMBER, 2001]:

1. Limpeza dos dados – organização dos dados e tratamento dos dados inconsistentes, incompletos ou irrelevantes.
2. Integração dos dados – Múltiplas fontes de dados heterogêneas podem ser integradas em uma única.

3. Seleção dos dados – seleciona da base de dados os dados relevantes para a análise.
4. Transformação dos dados – transformação ou consolidação dos dados em formatos apropriados para o processo de mineração de dados.
5. Mineração de dados – processo essencial onde métodos inteligentes são aplicados para extrair padrões dos dados.
6. Avaliação dos padrões - identificação da importância dos padrões encontrados.
7. Apresentação e representação do conhecimento - uso de técnicas de visualização e representação do conhecimento para apresentar ao usuário o conhecimento obtido.

A mineração de dados é um passo essencial dentro do processo de *KDD* [MANNILA, 1997]. Por esse motivo, os termos mineração de dados e *KDD* são comumente tratados como sinônimos.

Os objetivos da descoberta do conhecimento são definidos pelo uso pretendido do sistema. Em outras palavras, os objetivos dependem dos dados a serem analisados e do tipo de informação que se pretende obter. Segundo [FAYYAD;PIATETSKY-SHAPIRO; SMYTH, 1997], pode-se distinguir dois tipos de objetivos, a princípio: *verificação* e *descoberta*. Na verificação, o sistema é limitado a verificar as hipóteses formuladas pelo usuário. Na descoberta, o sistema autonomamente encontra novos padrões. Esses padrões podem ser classificados em duas categorias: *descritivos* e *preditivos* [HAN; KAMBER, 2001]. Tarefas descritivas caracterizam as propriedades gerais dos dados na base de dados. Tarefas preditivas fazem uma inferência a partir dos dados presentes, de maneira a fazer predições sobre dados futuros. Existem diversas *tarefas* de mineração de dados e, de acordo com os objetivos da aplicação e da natureza dos dados, uma determinada tarefa é utilizada. As tarefas de mineração de dados mais conhecidas são: *associação*, *classificação*, *agrupamento* (*clustering*) e *regressão*. A tarefa de associação, que consiste na descoberta de *regras de associação* (seção 2.2), é uma das mais utilizadas entre elas.

2.2. REGRAS DE ASSOCIAÇÃO

A tarefa de associação, que envolve a descoberta de regras de associação, é uma das tecnologias predominantes em mineração de dados. O banco de dados é tratado como uma coleção de transações, sendo que cada uma envolve um conjunto de itens [ELMASRI; NAVATHE, 2004].

Uma regra de associação é uma expressão da forma

$$X \rightarrow Y,$$

onde X e Y são conjuntos de itens. O significado de tal regra é que transações da base de dados que contém X tendem a conter Y também. O conjunto de itens que aparece à esquerda da seta (representado por X) é chamado de *antecedente* da regra. Já o conjunto de itens que aparece à direita da seta (representado por Y) é o *conseqüente* da regra. Assim, uma regra de associação tem o seguinte formato:

Antecedente → C*onseqüente*

A cada regra são associados dois fatores: *suporte* e *confiança*. Para uma regra de associação $X \rightarrow Y$, o suporte indica a porcentagem de registros em que aparecem X e Y simultaneamente, sobre o total de registros. Já a confiança indica a porcentagem de registros que contém X e Y, sobre o total de registros que possuem X. Um conjunto de itens é chamado de *itemset* e seu suporte é a porcentagem das transações que contêm todos os itens do *itemset*. Um *itemset* é dito freqüente quando o seu suporte é maior ou igual a um valor de suporte mínimo definido pelo usuário.

A tarefa de mineração de regras de associação consiste em duas etapas:

1. Encontrar todos os *itemsets* freqüentes.
2. Gerar regras de associação a partir dos *itemsets* freqüentes.

Se na base de dados de transações existir m itens diferentes, o número de possíveis *itemsets* distintos é 2^m. Desta maneira, a maior dificuldade na mineração de regras de associação está em determinar todos os *itemsets* freqüentes.

Seja, por exemplo, $I = \{1, 2, 3, 4\}$ o conjunto de todos os possíveis itens em uma base de dados. Dessa maneira tem-se $2^4 = 16$ possíveis *itemsets*. Na Figura 2.1 é apresentada uma estrutura, denominada *lattice*, que representa todos os possíveis *itemsets* do conjunto I. Nessa estrutura, a borda em destaque separa os *itemsets* freqüentes (localizados na parte superior) dos *itemsets* não freqüentes (localizados na parte inferior). A existência dessa borda é garantida pela propriedade de que para um *itemset* ser freqüente, todos os seus subconjuntos também devem ser freqüentes [AGRAWAL; SRIKANT, 1994]. No entanto, o seu formato depende do suporte de cada *itemset* e do suporte mínimo especificado.

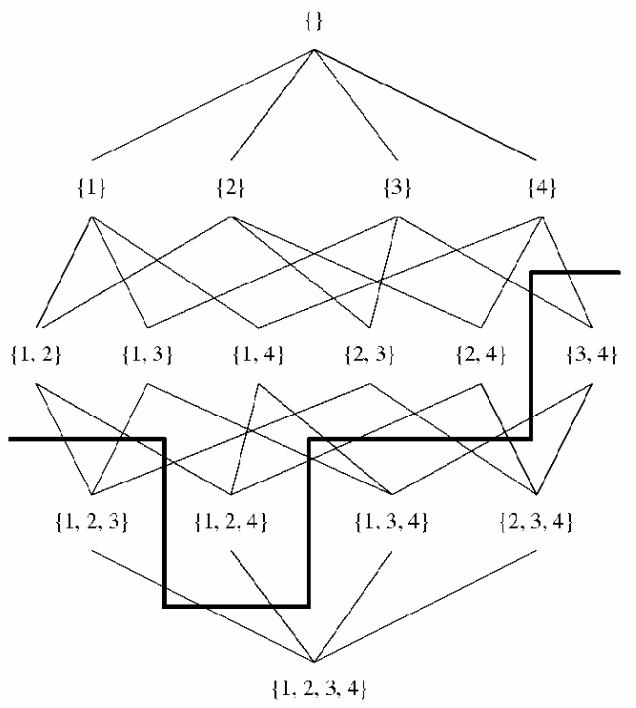

Figura 2.1 – *Lattice* de *I* = {1, 2, 3, 4} [HIPP;GÜNTZER; NAKHAEIZADEH, 2000].

O princípio básico dos algoritmos de mineração de regras de associação é empregar essa borda de maneira a diminuir o número de *itemsets* analisados quanto a serem freqüentes ou não. Assim que a borda é encontrada, os algoritmos restringem-se a determinar o suporte dos *itemsets* acima da mesma, e ignorar os que se encontram abaixo dela [HIPP;GÜNTZER; NAKHAEIZADEH, 2000].

As estratégias mais comuns para se encontrar a borda entre os *itemsets* freqüentes e não freqüentes são a busca em largura e a busca em profundidade. A busca em largura determina o suporte de todos os *itemsets* de tamanho *k-1* antes de determinar o suporte para os de tamanho *k*. Em contraste, a busca em profundidade desce recursivamente a estrutura apresentada na Figura 2.1.

Para determinar se um *itemset* é freqüente, é preciso determinar o seu suporte. Um *itemset* que possivelmente é freqüente e que, durante a fase em que se percorre os *itemsets*, decide-se que terá seu suporte determinado, é chamado de *itemset* candidato ou simplesmente candidato.

Uma abordagem comum para se calcular o suporte de um candidato é contar diretamente na base de dados a sua ocorrência. Para esse propósito um contador é iniciado com zero e todas as

transações da base de dados são analisadas. Cada vez que um candidato é reconhecido como um subconjunto da transação, seu contador é incrementado.

Outra abordagem é determinar o suporte dos candidatos através de intersecções entre conjuntos. Um *tid* é um identificador único de uma transação. Para um item, sua *tidlist* é o conjunto dos identificadores das transações que o contém. Da mesma maneira, *tidlists* também existem para cada *itemset X*, e são representadas por *X.tidlist*. A *tidlist* de um candidato $C = X \bigcup Y$ é obtida através de $C.tidlist = X.tidlist \bigcap Y.tidlist$. O suporte do candidato *C* é obtido através de $\frac{|C.tidlist|}{|D|}$, onde $|C.tidlist|$ é o número de *tids* na *tidlist* e $|D|$ é o número de transações na base de dados *D*.

As estratégias para determinar *itemsets* freqüentes são caracterizadas pelo modo como são percorridos os possíveis *itemsets* e como são determinados os valores de suporte para cada *itemset*. Neste trabalho são apresentados 4 algoritmos, representando cada estratégia para determinar os *itemsets* freqüentes:

- Apriori [AGRAWAL; SRIKANT, 1994]
- Partition [SAVARESE;OMIECINSKI; NAVATHE, 1995]
- FP-growth [HAN;PEI; YIN, 2000]
- Eclat [ZAKI *et al.*, 1997]

A Figura 2.2 apresenta qual estratégia é adotada por cada algoritmo. Esses algoritmos são detalhados na seção 2.3.

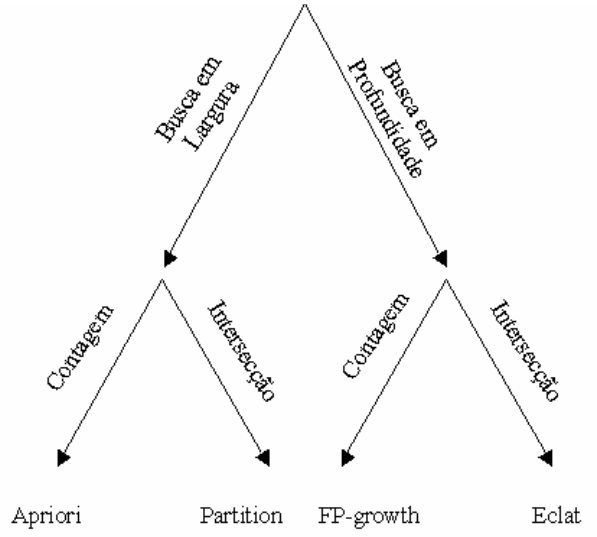

Figura 2.2 – Estratégias de mineração [HIPP;GÜNTZER; NAKHAEIZADEH, 2000].

2.3. ALGORITMOS DE ASSOCIAÇÃO

2.3.1. *Apriori*

Entre os diversos algoritmos para realizar a mineração de dados buscando regras de associação, um dos mais famosos e utilizados é o *Apriori* [AGRAWAL; SRIKANT, 1994]. Sua criação representou um grande diferencial em relação aos algoritmos anteriores a ele, principalmente no que se refere ao desempenho e à estratégia de solução do problema de mineração de regras de associação [WOJCIECHOWSKI; ZAKRZEWICZ, 2002]. Por esse motivo, o algoritmo *Apriori* é considerado um algoritmo clássico, e a partir dele muitos algoritmos foram posteriormente criados, formando o que muitos chamam de "família *Apriori*" [ORLANDO;PALMERINI; PEREGO, 2001].

O objetivo do algoritmo pode ser descrito dessa forma: dados

- um conjunto de transações D, $D = \{T \mid T$ é um conjunto de itens$\}$;
- um suporte mínimo *minsup*;
- uma confiança mínima *minconf.*

obter todas as regras de associação que possuam

- confiança $\geq$ *minconf;*
- suporte $\geq$ *minsup.*

Assim, a estratégia do *Apriori* é identificar os conjuntos de *itemsets* freqüentes (cujo suporte seja maior ou igual a *minsup*) e construir regras a partir desses conjuntos, que possuam confiança maior ou igual a *minconf*. A idéia inovadora desse algoritmo é a de que se um conjunto está abaixo do suporte mínimo, pode-se ignorar todos os seus superconjuntos. Com isso, o algoritmo ganha em desempenho, já que não perde tempo analisando esses superconjuntos que não são freqüentes. Essa otimização é possível porque a busca em largura garante que os valores dos suportes de todos os subconjuntos de um candidato são conhecidos antecipadamente. Em uma varredura da base de dados, o *Apriori* realiza simultaneamente a contagem de todos os candidatos de tamanho k. A parte crítica do algoritmo é procurar pelos candidatos em cada transação. Para esse propósito, foi introduzida uma estrutura de dados chamada *hash-tree*. Os itens em cada transação são utilizados para percorrer a *hash-tree*. Quando é alcançada uma de suas folhas, é encontrado um conjunto de candidatos que possui um prefixo comum. Esses candidatos são procurados na transação e, se encontrados, o contador deles na *hash-tree* é incrementado.

O algoritmo *Apriori* é apresentado na Figura 2.3.

1) L_1 = {Conjunto dos *itemsets* freqüentes de tamanho 1}

2) **para** ($k = 2$; $L_{k-1} \neq \varnothing$; k++)

3) C_k = *apriori-gen*(L_{k-1}) // Geração de candidatos

4) **para todas** as transações t

5) C_t = *subset*(C_k , t) // Candidatos contidos na transação t

6) **para todos** os candidatos c em C_t fazer c.contagem++

7) **fim de para todas**

8) L_k = { c em C_k | c.contagem $\geq$ *minsup* }

9) **fim de para**

10) Resposta = Reunião de todos os L_k

Figura 2.3 – O algoritmo Apriori.

No algoritmo:

- k indica o passo;
- L_k é o conjunto de *itemsets* freqüentes que possui tamanho k; e
- C_k é o conjunto de *itemsets* candidatos que possui tamanho k.

A função *subset*, cuja chamada aparece na linha 5 do algoritmo apresentado na Figura 2.3, é a responsável por analisar se em uma transação algum de seus subconjuntos é um *itemset* candidato. Para isso, os *itemsets* candidatos são inicialmente armazenados em uma *hash-tree*, sendo que os itens de um *itemset* são armazenados de maneira ordenada. Um nó de uma *hash-tree* ou contém uma lista de *itemsets* (nó folha) ou contém uma tabela *hash* (nó interior). Em um nó interior existem apontadores para outros nós. A raiz de uma *hash-tree* é definida como sendo de profundidade 1. Um nó interior localizado a uma profundidade p aponta para nós à profundidade p+1. *Itemsets* são armazenados nos nós folha. Quando se deseja armazenar um *itemset*, inicia-se pela raiz e percorre-se a *hash-tree* até se chegar em um nó folha. Em um nó interior a uma profundidade p, decide-se qual caminho seguir aplicando-se uma função *hash* no p-ésimo item do *itemset*. Todos os nós são criados como nós folha. À medida que o número de *itemsets* em um nó ultrapassa um determinado limite, o nó é convertido em um nó interior.

O *Apriori* pode ser dividido em 2 fases:

1. *Geração de candidatos*: busca itens freqüentes e os identifica.
2. *Geração de regras*: uso dos itens freqüentes para gerar as regras desejadas.

Na geração de candidatos, a função *apriori-gen* recebe como argumento L_{k-1} (o conjunto de todos os itens freqüentes de tamanho *k-1*), e retorna um conjunto de todos os itens freqüentes de tamanho k. A tarefa realizada por essa função pode ser dividida em dois passos: *Join* (Junção) e *Prune* (Ajuste). No passo *Join* são geradas as combinações, através da junção de L_{k-1} com L_{k-1}, conforme mostra a Figura 2.4.

1) **insert into** C_k

2) **select** $p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}$

3) **from** $L_{k-1}\ p, L_{k-1}\ q$

4) **where** $p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1};$

Figura 2.4 – O passo *Join* [AGRAWAL; SRIKANT, 1994].

No passo *Prune* são eliminados os candidatos que possuem subconjuntos não freqüentes. Em outras palavras, é feito um ajuste no qual são removidos todos os itens $c \in C_k$ tal que qualquer sub-item de c de tamanho $k-1$ não esteja em L_{k-1}. O passo *Prune* é mostrado na Figura 2.5.

1) **forall** itemsets $c \in C_k$ do

2) **forall** ($k-1$)-subsets s of c do

3) **if** ($s \notin L_{k-1}$) then

4) **delete** c from C_k;

Figura 2.5 – O passo *Prune* [AGRAWAL; SRIKANT, 1994].

Após o passo *Prune*, já é possível reunir todos os itens freqüentes (L_k). A partir deles, são geradas as regras de associação, que constituem a resposta do algoritmo. Nesse ponto inicia-se a fase de geração de regras. Um dos avanços do *Apriori,* em relação aos algoritmos de associação anteriores a ele, é a geração de regras de associação com conseqüentes que possuam mais de um item [AGRAWAL; SRIKANT, 1994].

A idéia da geração de regras a partir dos itens freqüentes pode ser apresentada da seguinte forma: a partir de *itemsets* freqüentes l, construir regras da forma

$$(l - a) \rightarrow a,$$

em que $a \subseteq l$. A confiança dessa regra deve ser maior ou igual a *minconf*, e é calculada conforme mostra a Figura 2.6.

$$Confiança[(l-a) \rightarrow a] = \frac{Suporte(l)}{Suporte(l-a)}$$

Figura 2.6 – Cálculo da confiança para as regras geradas.

De maneira menos formal, pode-se dizer que o cálculo da confiança é obtido através da divisão entre o suporte da regra (*l*) e o suporte de seu antecedente (*l* - *a*).

O algoritmo de geração de regras é mostrado na Figura 2.7, onde aparece a chamada para a função *ap-genrules* (linha 3). Essa função é detalhada na Figura 2.8. O cálculo da confiança aparece na linha 5 do algoritmo dessa mesma figura.

1) **para todos** os conjuntos freqüentes de tamanho k, L_k, $k \geq 2$
2) $H_1 = \{c \mid c$ em $L_k\}$ //conseqüentes derivados de L_k com apenas 1 item.
3) **chamar** *ap-genrules*(L_k, H_1)
4) **Fim**

Figura 2.7 – Algoritmo de geração de regras.

```
1)  ap-genrules( Lk, Hm: conseqüentes de tamanho m)
2)  se k > m + 1 então
3)          Hm+1 = apriori-gen(Hm)
4)                  para todos hm+1 em Hm+1
5)                          conf = suporte(Lk)/suporte(Lk - hm+1)
6)                          se conf ≥ minconf então
7)                                  escreve a regra (Lk - hm+1) → hm+1
8)                                  com confiança = conf e suporte(Lk)
9)                          senão
10)                                 apagar hm+1 de Hm+1      //prune
11)                 fim de para todos
12)                 chama ap-genrules(Lk, Hm+1)
13) Fim
```

Figura 2.8 – Função *ap-genrules*.

Dessa forma todas as regras são geradas. As que possuírem suporte maior ou igual a *minsup* e confiança maior ou igual a *minconf* serão consideradas regras válidas, e serão exibidas ao final da execução do algoritmo, juntamente com o valor de suporte e confiança de cada uma delas.

2.3.2. *Partition*

O algoritmo *Partition* [SAVARESE;OMIECINSKI; NAVATHE, 1995] é um algoritmo parecido com o *Apriori*, mas que utiliza intersecções entre conjuntos para determinar os valores de suporte. Como descrito anteriormente, o *Apriori* conta o suporte de todos os *itemsets* candidatos de tamanho k–1 antes de contar o suporte dos *itemsets* candidatos de tamanho k. O *Partition* utiliza os *tidlists* dos *itemsets* freqüentes de tamanho k-1 para gerar as *tidlists* dos *itemsets* candidatos de tamanho k. Desta maneira é bem provável que o tamanho dos resultados intermediários fique maior do que a memória física disponível. Para evitar este problema, o *Partition* divide a base de dados em vários pedaços que são tratados de maneira independente. O tamanho de cada pedaço é escolhido de maneira que todas as suas *tidlists* intermediárias caibam na memória. Depois de determinar os *itemsets* freqüentes em cada pedaço, uma passagem extra sobre a base de dados é necessária para garantir que os *itemsets* que são freqüentes localmente também são freqüentes em toda a base de dados. A chave para o funcionamento do algoritmo é que um *itemset* freqüente em toda a base de dados é freqüente em pelo menos uma das n partições [SAVARESE;OMIECINSKI; NAVATHE, 1995].

A Figura 2.9 apresenta o algoritmo *Partition*. A Tabela 2.1 apresenta a notação utilizada.

Algoritmo: *Partition*

Entrada: Base de dados D, suporte mínimo *min_sup*

Saída: *itemsets* freqüentes L^G

Método:

1) P = particiona_base_de_dados(D);
2) n = número de partições;

// Fase 1

3) para ($i = 1$; $i \leq n$; i++) {
4) lê_partição($p_i \in P$);

5) L^i = gen_itemsets_grandes(p_i);
6) }
// Fase da junção
7) para ($i = 2$; $L_i^j \neq 0$, $j = 1,2,...,n$; i++) {
8) $C_i^G = U_{j=1,2,...,n} L_i^j$;
9) }
// Fase 2
10) para ($i = 1$; $i \leq n$; i++) {
11) lê_partição($p_i \in P$);
12) gen_count(C^G, p_i);
13) }
14) $L^G = \{c \in C^G \mid c.\text{count} \geq min\_sup\}$
15) Retorna L^G

gen_itemsets_grandes(p: partição da base de dados)

1) L_1^p = {*itemsets* freqüentes de tamanho 1 com suas *tidlists*}
2) para($k = 2$; $L_k^p \neq 0$; k++){
3) para todos *itemsets* $l_1 \in L_{k-1}^p$ {
4) para todos *itemsets* $l_2 \in L_{k-1}^p$ {
5) se ($l_1[1] = l_2[1]$) e ($l_1[2] = l_2[2]$) e ... e ($l_1[k\text{-}1] < l_2[k\text{-}1]$) então {
6) $c = l_1[1] \cdot l_1[2] \cdots l_1[k–1] \cdot l_2[k–1]$;
7) se não_descartável(c) então {
8) $c.tidlist = l_1.tidlist \cap l_2.tidlist$;
9) se $|c.tidlist| / |p| \geq min\_sup$ então
10) $L_k^p = L_k^p \cup \{c\}$;
11) }
12) }
13) }
14) }
15) }

```
16) retorna ⋃_k L_k;

    não_descartável(c: itemset de tamanho k)

1)  para cada subset s de c, de tamanho k-1
2)      se s ∉ L_{k-1} então
3)          retorna FALSO
4)  retorna VERDADEIRO

    gen_count(C^G: conjunto dos candidatos globais; p: partição da base de dados)

1)  para todos os itemsets de tamanho 1 gerar tidlists
2)  para (k = 2; C_k^G ≠ 0; k++)
3)      para todos os itemsets c de tamanho k, c ∈ C_k^G {
4)          templist = c[1].tidlist ∩ c[2].tidlist ∩ … ∩ c[k].tidlist;
5)          c.count = c.count + |templist|
6)          }
7)  }
```

Figura 2.9 – Algoritmo *Partition.*

Tabela 2.1 - Notação utilizada na Figura 2.9.

L_k	Conjunto de *itemsets* freqüentes de tamanho k
L_k^p	Conjunto de *itemsets* freqüentes de tamanho k em uma partição p
C_k^G	Conjunto de *itemsets* candidatos globais de tamanho k
C^G	Conjunto de candidatos globais
L^i	Conjunto de *itemsets* em uma partição p
L^G	Conjunto de *itemsets* globais
p_i	i-ésima partição da base de dados

Eis o funcionamento do Partition [SAVARESE;OMIECINSKI; NAVATHE, 1995]: inicialmente a base de dados D é dividida logicamente em n partições. A Fase 1 do algoritmo possui n iterações. Durante a iteração i, somente a partição p_i é considerada. O procedimento *gen_itemsets_grandes* recebe a partição p_i como entrada e gera como saída os *itemsets* de todos

os tamanhos, L_2^i, L_3^i, ... , L_k^i, freqüentes dentro da partição. Na fase da junção os *itemsets* freqüentes de todas as partições e de mesmo tamanho são combinados para gerar os *itemsets* candidatos globais. O conjunto de *itemsets* globais de tamanho *i* é computado como

$$C_i^G = U_{j=1,2,...,n} L_i^j.$$

Na Fase 2, o algoritmo inicia contadores para cada *itemset* candidato global e conta os seus suportes em cada uma das *n* partições. O algoritmo retorna os *itemsets* que possuem suporte maior do que o mínimo especificado. O algoritmo percorre inteiramente a base de dados uma vez na Fase 1 e uma vez na Fase 2.

O procedimento *gen_itemsets_grandes* inicialmente gera as *tidlists* para todos os *itemsets* freqüentes de tamanho 1 diretamente através da leitura da partição recebida como entrada. A *tidlist* de um *itemset* candidato de tamanho *k* é gerada através da junção das *tidlists* dos *itemsets* de tamanho *k*-1 que foram utilizados para gerar o *itemset* candidato de tamanho *k*. Sempre que um candidato possuir subconjuntos não freqüentes, ele será descartado, não sendo assim gerada a sua *tidlist* para posterior contagem do seu suporte.

A contagem final do suporte para os *itemsets* em relação a toda base de dados é feita na Fase 2 do algoritmo. O conjunto de *itemsets* candidatos globais é gerado através da união de todos os *itemsets* freqüentes de todas as *n* partições. Essa fase também possui *n* iterações. Inicialmente um contador é iniciado com zero para cada candidato. Para cada partição, são geradas as *tidlists* para todos os *itemsets* de tamanho 1 que pertencem àquela partição. O suporte para um candidato é calculado através da intersecção das *tidlists* de todos os *itemsets* de tamanho 1 que formam o candidato. A soma dos suportes para cada partição é o valor de suporte para a base de dados.

2.3.3. FP-growth

O algoritmo *FP-growth* [HAN;PEI; YIN, 2000] possui um passo de pré-processamento no qual é derivada uma representação altamente condensada da base de dados, chamada *FP-tree*. A geração da *FP-tree* é feita através de busca em profundidade e contagem de ocorrências. Em um segundo passo, o *FP-growth* usa a *FP-tree* para determinar os valores de suporte para todos os *itemsets* freqüentes. A Figura 2.10 apresenta o processo de construção da *FP-tree* e o algoritmo *FP-growth*.

Entrada: Base de dados *D*, suporte mínimo *min_sup*
Saída: Conjunto de *itemsets* freqüentes
Método:

1. A *FP-tree* é construída através dos seguintes passos:
    a. Percorrer a base de dados *D* uma vez. Determinar o conjunto de itens freqüentes *F* e seus suportes. Ordenar *F* em ordem decrescente em função do suporte e chamá-la de *L*, a lista de itens freqüentes.
    b. Criar a raiz da *FP-tree* e nomeá-la como "*null*". Para cada transação *t* em *D* fazer o seguinte:
        i. Selecionar e ordenar os itens freqüentes em *t* de acordo com a ordem de *L*, sendo a lista de itens freqüentes em *t* igual a [*p*|*P*], onde *p* é o primeiro elemento e *P* é o resto da lista.
        ii. Executar insere_tree([*p*|*P*], *T*). Se *T* tiver um filho *N* de tal maneira que *N*.nome_item = *p*.nome_item, então incrementar o contador de *N* por 1. Caso contrário, criar um novo nó *N*, e iniciar seu contador com 1. Ligar o seu *parent-link* a *T*, e seu *node-link* aos nós de mesmo (nome_item) através da estrutura dos *node-links*. Se *P* não for vazio, chamar insere_tree(*P*,*N*) recursivamente.
2. A mineração da *FP-tree* é feita através da execução de FP-growth(*Tree*, *null*), cujo algoritmo é apresentado a seguir.

```
FP-growth(Tree: FP-tree, α: itemset)
1) se Tree contém apenas um caminho P então
2)     para cada combinação β de nós no caminho P
3)         gerar o padrão β ∪ α com suporte = suporte mínimo dos nós em β;
4) senão para cada a_i na tabela de node-links de Tree{
5)     gerar o padrão β = a_i ∪ α com suporte = a_i.suporte
6)     construir a base de padrões condicionada de β e então construir a FP-tree
       condicionada de β chamada de Tree_β;
7)     se Tree_β ≠ Ø então
8)         FP-growth(Tree_β, β);
9) }
```

Figura 2.10 – Algoritmo de construção da *FP-tree* e o algoritmo *FP-growth.*

Para uma melhor explicação do funcionamento do *FP-growth* e da criação da *FP-tree*, considere o exemplo a seguir. As duas primeiras colunas da Tabela 2.2 mostram um exemplo de base de dados de transações. A terceira coluna apresenta a lista de itens freqüentes ordenados em função do suporte. Para o exemplo a seguir considerou-se o suporte mínimo como sendo 60%.

Tabela 2.2– Base de dados de transações [HAN;PEI; YIN, 2000].

TID	Itens	Itens freqüentes (ordenados)
100	*f,a,c,d,g,i,m,p*	*f,c,a,m,p*
200	*a,b,c,f,l,m,o*	*f,c,a,b,m*
300	*b,f,h,j,o*	*f,b*
400	*b,c,k,s,p*	*c,b,p*
500	*a,f,c,e,l,p,m,n*	*f,c,a,m,p*

Inicialmente percorrendo a base de dados, é determinada a lista de itens freqüentes {(*f*:4), (*c*:4), (*a*:3), (*b*:3), (*m*:3), (*p*:3)}, onde o número após o ":" é quantidade de ocorrências de um determinado item. A lista é ordenada em ordem decrescente em função do suporte.

Em seguida é criada a raiz da *FP-tree*, nomeada "*null*". A base de dados então é percorrida uma segunda vez. A primeira transação leva à construção do primeiro ramo da árvore {(*f*:1), (*c*:1), (*a*:1), (*m*:1), (*p*:1)}, seguindo a ordem da lista de itens freqüentes. Para a segunda transação, como a sua lista de itens freqüentes (*f*,*c*,*a*,*b*,*m*) compartilha o mesmo prefixo (*f*,*c*,*a*) com o ramo existente (*f*,*c*,*a*,*m*,*p*), o contador para cada nó do prefixo é incrementado de 1, e um novo nó (*b*:1) é criado e ligado como filho de (*a*:2), e um outro nó (*m*:1) é criado e ligado como filho de (*b*:1). Para a terceira transação, como sua lista de itens freqüentes (*f*,*b*) compartilha apenas o nó (*f*) com o ramo da *FP-tree* iniciado com prefixo *f*, o contador de *f* é incrementado de 1 e um novo nó (*b*:1) é criado e ligado como filho de (*f*:3). A quarta transação leva à construção do segundo ramo da *FP-tree*, {(*c*:1), (*b*:1), (*p*:1)}, já que essa transação não tem nenhum prefixo em comum com o ramo já existente. Para a última transação, já que sua lista de itens freqüentes é idêntica ao da primeira, todos os nós que participam dessa lista têm o seu contador incrementado de 1. Para se percorrer mais facilmente a *FP-tree*, uma tabela é criada na qual cada item dela aponta para uma

seqüência de nós com mesmo nome através de uma estrutura chamada *node-links*. Um nó é ligado aos seus filhos através de um *parent-link*. A Figura 2.11 apresenta a *FP-tree* criada.

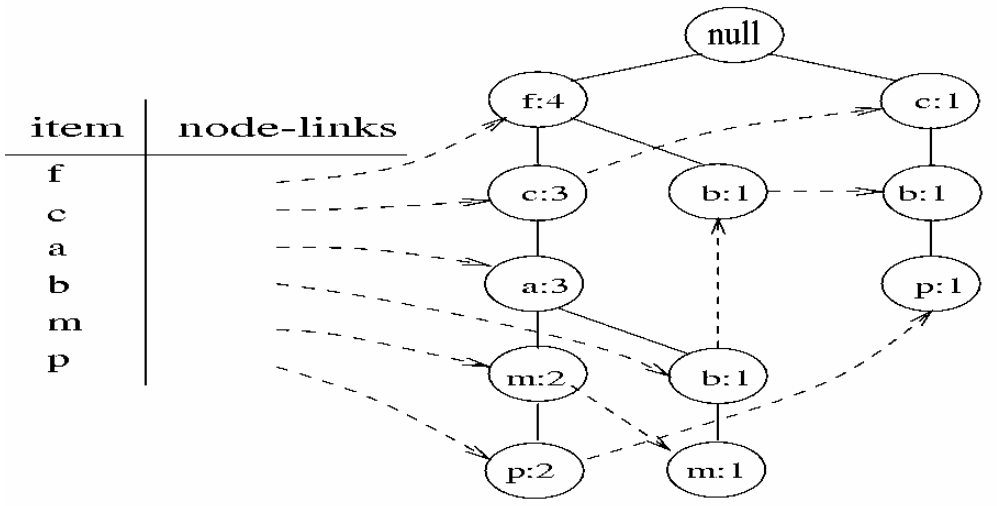

Figura 2.11 – *FP-Tree* [HAN;PEI; YIN, 2000].

O processo de mineração da *FP-tree* começa de baixo para cima na tabela de *node-links*. Para *p*, é derivado o *itemset* (*p*:3) e dois caminhos da *FP-tree*: {*f*:4, *c*:3, *a*:3, *m*:2, *p*:2} e {*c*:1, *b*:1, *p*:1}. O primeiro caminho mostra que (*f*,*c*,*a*,*m*,*p*) aparece 2 vezes na base de dados. Apesar de (*f*,*c*,*a*) aparecer 3 vezes e (*f*) sozinho aparecer 4 vezes na base de dados, eles só aparecem em conjunto com *p* duas vezes. Assim, para se estudar quais itens aparecem com *p*, apenas o prefixo do caminho de *p* {*f*:2,*c*:2,*a*:2,*m*:2} é necessário. De maneira similar, (*c*,*b*,*p*) aparece apenas uma vez na base de dados, e o prefixo de caminho de *p* é {*c*:1, *b*:1}. Os dois caminhos que prefixam *p*, {*f*:2,*c*:2,*a*:2,*m*:2} e {*c*:1, *b*:1} formam a base de subpadrões de *p*, que é chamada de base de padrões condicionada (isto é, a base de subpadrões sob a condição da existência de *p*). A construção de uma *FP-tree* nessa base de padrões condicionada (chamada de *FP-tree* condicionada) leva a apenas um ramo, (*c*:3). Desta maneira o único *itemset* derivado é (*cp*:3). Assim a busca para *itemset* freqüentes associados a *p* termina.

Para *m*, é derivado o *itemset* (*m*:3) e dois caminhos: {*f*:4, *c*:3, *a*:3, *m*:2} e {*f*:4, *c*:3, *a*:3, *b*:1, *m*:1}. Na análise de *m*, *p* não é incluído já que qualquer *itemset* freqüente envolvendo *p* já foi analisado previamente. De maneira similar à analise de *p*, a base de dados condicionada de *m* é {*f*:2, *c*:2, *a*:2} e {*f*:1, *c*:1, *a*:1, *b*:1}. Construindo uma *FP-tree* baseada nessa base de dados condicionada, é derivada a *FP-tree* condicionada de *m*, {*f*:3, *c*:3, *a*:3}, que contém apenas um

ramo. Após isso, pode-se executar recursivamente um algoritmo de mineração baseado em *FP-tree* (como o *FP-growth*), por exemplo, “minere({*f*:3,*c*:3,*a*:3}|*m*)”.

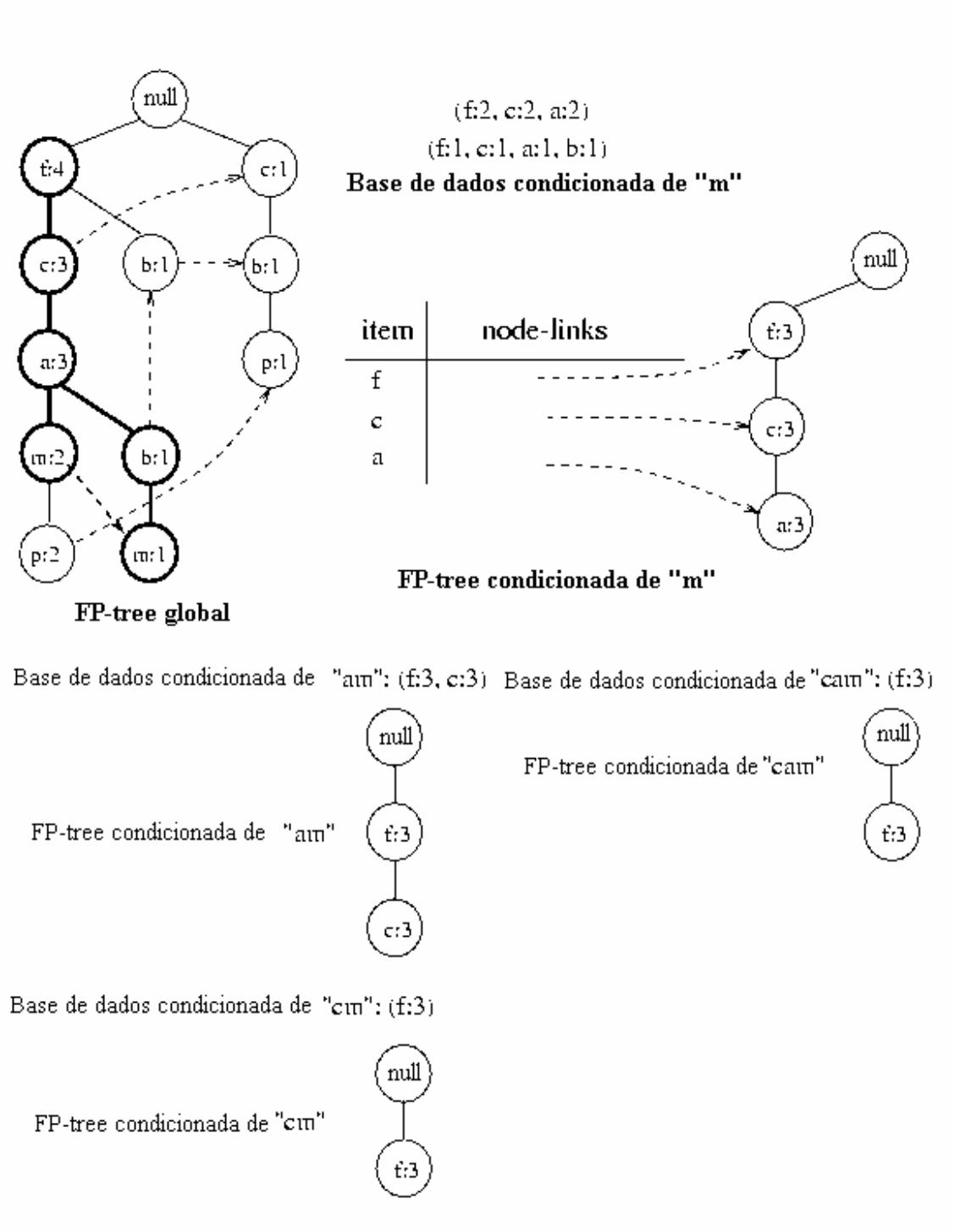

Figura 2.12 – *FP-tree* condicionada de *m* [HAN;PEI; YIN, 2000].

A Figura 2.12 mostra que "minere({*f*:3,*c*:3,*a*:3}|*m*)" envolve minerar três itens (*a*), (*c*), (*f*) em seqüência. O primeiro item deriva o *itemset* freqüente (*am*:3) e chama "minere({*f*:3,*c*:3}|*am*)". O segundo deriva o *itemset* freqüente (*cm*:3) e chama "minere({*f*:3}|*cm*)". O terceiro deriva apenas o *itemset* freqüente (*fm*:3). A chamada a "minere({*f*:3,*c*:3}|*am*)" deriva (*cam*:3), (*fam*:3) e gera uma chamada para "minere({*f*:3}|*cam*)", que deriva o *itemset* mais longo (*fcam*:3). De maneira similar, a chamada a "minere({*f*:3}|*cm*)" deriva o *itemset* (*fcm*:3). Desta maneira, o conjunto de *itemsets* freqüentes envolvendo *m* é {(*m*:3), (*am*:3), (*cm*:3), (*fm*:3), (*cam*:3), (*fam*:3), (*fcm*:3), (*fcam*:3)}. Isso indica que uma *FP-tree* com apenas um caminho pode ser minerada através da combinação de todos os itens no caminho.

A mineração do nó *b* deriva (*b*:3) e três caminhos: (*f*:4, *c*:3, *a*:3, *b*:1), (*f*:4, *b*:1) e (*c*:1,*b*:1). Como a base de dados condicionada de *b* — {*f*:1,*c*:1,*a*:1}, {*f*:1} e {*c*:1} — não gera nenhum item freqüente, a mineração para *b* termina. O nó *a* deriva um *itemset* freqüente (*a*:3) e uma base de dados condicionada (*f*:3, *c*:3), uma *FP-tree* com apenas um caminho. Dessa maneira podemos gerar o conjunto de padrões freqüentes através da combinação dos itens da base. Concatenando eles com (*a*:3), temos {(*fa*:3), (*ca*:3), (*fca*:3)}. O nó *c* deriva (*c*:4) e uma base de dados condicionada (*f*:3), gerando o padrão (*fc*:3). O nó *f* deriva apenas (*f*:4) e nenhuma base de dados condicionada. A Tabela 2.3 apresenta todas as bases de dados condicionadas, as *FP-trees* condicionadas e os *itemsets* freqüentes gerados.

Tabela 2.3 – Bases de dados condicionadas, *FP-tree* condicionada e *Itemsets* freqüentes.

Item	Base de dados condicionada	*FP-Tree* condicionada	*Itemsets* freqüentes
p	{(*f*:2,*c*:2,*a*:2,*m*:2), (*c*:1,*b*:1)}	{(*c*:3)}|*p*	{(*p*:3, *cp*:3)}
m	{(*f*:2,*c*:2,*a*:2),(*f*:1,*c*:1,*a*:1,*b*:1)}	{(*f*:3,*c*:3,*a*:3)}|*m*	{(*m*:3), (*am*:3), (*cm*:3), (*fm*:3), (*cam*:3), (*fam*:3), (*fcm*:3), (*fcam*:3)}
b	{(*f*:1,*c*:1,*a*:1), (*f*:1), (*c*:1)}	Ø	{(*b*:3)}
a	{(*f*:3,*c*:3)}	{(*f*:3,*c*:3)}|*a*	{(*a*:3), (*fa*:3), (*ca*:3), (*fca*:3)}
c	{(*f*:3)}	{(*f*:3)}|*c*	{(*c*:4), (*fc*:3)}
f	Ø	Ø	{(*f*:4)}

2.3.4. Eclat

O algoritmo *Eclat* foi introduzido em [ZAKI *et al.*, 1997], combinando a busca em profundidade com intersecções entre conjuntos. Quando é utilizada a busca em profundidade, é necessário apenas manter em memória as *tidlists* dos *itemsets* correntemente em análise. Desta maneira, dividir a base de dados como o *Partition* não é mais necessário. O *Eclat* utiliza uma otimização chamada de "intersecções rápidas". Sempre que é efetuada a intersecção entre duas *tidlists*, a *tidlist* resultante só é interessante se possuir suporte maior do que o mínimo especificado pelo usuário. Nas "intersecções rápidas", o processo de intersecção é interrompido assim que se reconhece que o resultado não atingirá o suporte mínimo.

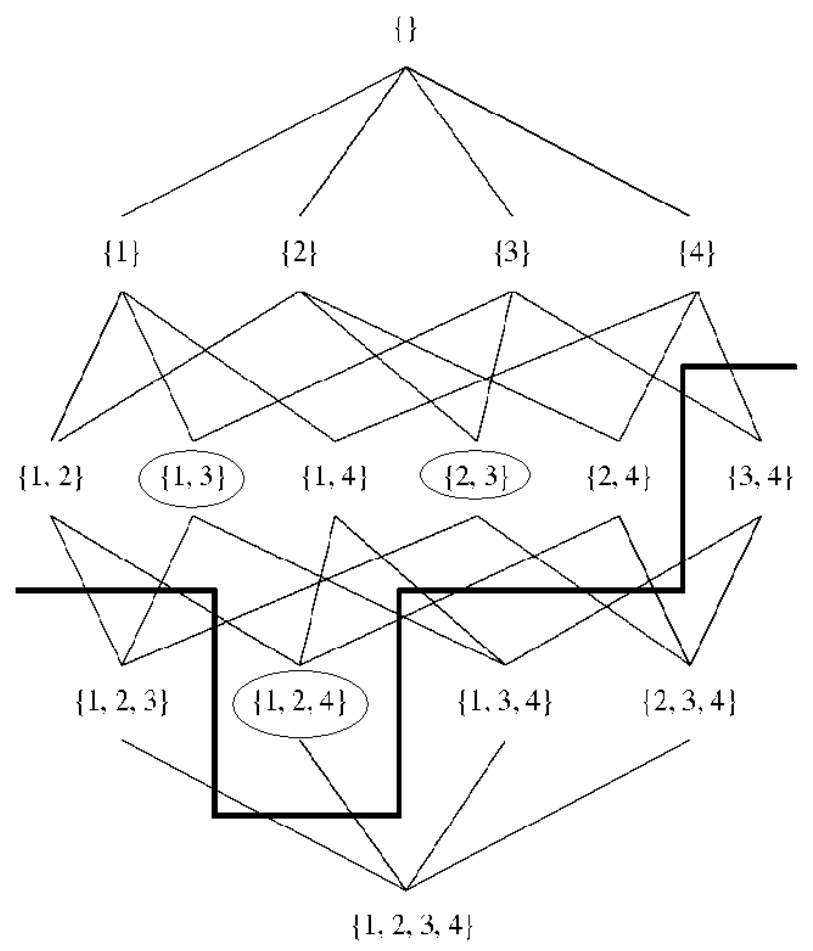

Figura 2.13 – *Itemsets* freqüentes máximos.

Na Figura 2.13, a borda separa os *itemsets* freqüentes dos não freqüentes, sendo que acima dela encontram-se os *itemsets* freqüentes. O itemset {3,4}, por exemplo, não é freqüente, pois se

encontra abaixo da borda. Um algoritmo de mineração de regras de associação deve determinar a borda de maneira eficiente, de modo a apenas testar e enumerar os *itemsets* que são freqüentes. A borda pode ser determinada precisamente pelos *itemsets* freqüentes máximos. Um *itemset* freqüente é *máximo* quando ele não é um subconjunto de nenhum outro *itemset* freqüente. Na Figura 2.13, os *itemsets* freqüentes máximos são apresentados dentro de uma elipse. A borda corresponde aos *itemsets* máximos e seus subconjuntos.

Dado o conhecimento de *itemsets* freqüentes máximos, seria desejável que um algoritmo obtivesse o suporte para eles e para os seus subconjuntos percorrendo apenas uma vez a base de dados. De maneira geral não é possível determinar os *itemsets* freqüentes máximos nos passos intermediários do algoritmo, mas é possível ter uma noção aproximada de quais serão esses *itemsets*. O *Eclat* utiliza uma técnica de clusterização para agrupar itens de modo a obter superconjuntos dos *itemsets* freqüentes máximos (*itemsets* freqüentes máximos em potencial), chamada de clusterização por classe de equivalência.

A clusterização por classe de equivalência ocorre da seguinte maneira. Seja L_2 = {AB, AC, AD, AE, BC, BD, BE, DE} o conjunto de *itemsets* freqüentes de tamanho 2. Para gerar os *itemsets* candidatos de tamanho 3, devemos combinar os *itemsets* de L_2, ou seja, C_3 = {ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE}. Assumindo então que L_{k-1} está ordenado lexicograficamente, pode-se particionar os *itemsets* de L_{k-1} em classes de equivalência baseado nos seus prefixos comuns de tamanho k-2. Os *itemsets* candidatos de tamanho k podem ser gerados a partir dos *itemsets* de uma classe, através de junção de pares de itens da classe e concatenação com o identificador da classe como prefixo. Para L_2 temos as seguintes classes de equivalência: [A] = {B, C, D, E}, [B] = {C, D, E} e [D] = {E}. A classe de equivalência [A] gera os *itemsets* {ABC, ABD, ABE, ACD, ACE, ADE} e [B] gera {BCD, BCE, BDE}, sendo que os *itemsets* gerados por uma classe são independentes dos gerados por outra. Qualquer classe com apenas 1 elemento pode ser descartada, já que nenhum *itemset* pode ser gerado a partir dela, e desta maneira a classe [D] é eliminada.

Em qualquer passo intermediário do algoritmo, quando o conjunto de *itemsets* freqüentes L_k com $k \geq 2$ tiver sido gerado, é possível gerar um conjunto de *itemsets* freqüentes máximos em potencial através das classes de equivalência. Os *itemsets* freqüentes máximos em potencial são gerados concatenando-se todos os itens de uma classe de equivalência, e tendo o identificador da

classe como prefixo. A classe [A] gera o *itemset* freqüente máximo em potencial ABCDE, e a classe [B] gera BCDE. Como BCDE é subconjunto de ABCDE, ele é descartado como *itemset* freqüente máximo em potencial. Quanto maior o valor de k, mais preciso o processo de clusterização. Note-se que para $k = 1$, todo o universo de itens será considerado como *itemset* máximo. No entanto, para $k \geq 2$ é possível extrair um conhecimento mais preciso sobre a associação entre os itens.

Uma vez determinados os *itemsets* freqüentes máximos em potencial, devem ser encontrados os verdadeiros *itemsets* freqüentes máximos. Cada *itemset* freqüente máximo em potencial pode ser decomposto em *itemsets*, sendo esse conjunto de *itemsets* mais o *itemset* freqüente máximo em potencial um subconjunto dos *itemsets* da base de dados. Todos esses *itemsets* devem ser analisados para se determinar o verdadeiro *itemset* freqüente máximo. A Figura 2.14 mostra os *itemsets* que compõem os *itemsets* freqüentes máximos em potencial da Figura 2.13.

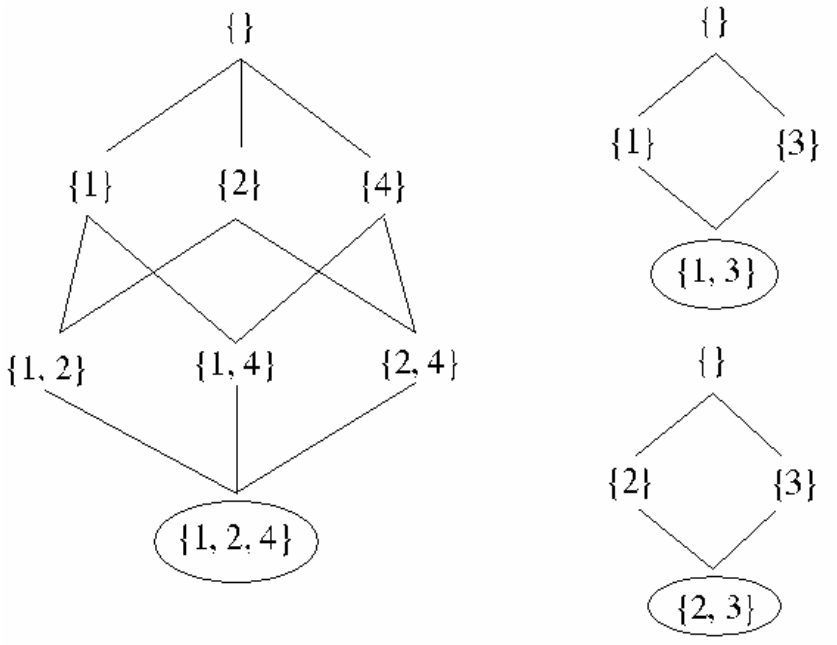

Figura 2.14 – Composição dos *itemsets* freqüentes máximos em potencial.

Os *itemsets* gerados são percorridos utilizando uma busca em largura como no *Apriori*. Como os *itemsets* gerados são um subconjunto dos *itemsets* da base de dados, pode-se dizer que em relação a toda a base de dados, o *Eclat* faz busca em profundidade. A Figura 2.15 mostra como o *Eclat* percorre os *itemsets* gerados.

Algoritmo: *Eclat*

Entrada: $F_k = \{I_1..I_n\}$ cluster de *itemsets* freqüentes de tamanho *k*, suporte mínimo *min_sup*

Saída: *itemsets* freqüentes de tamanho *l*, $l > k$

Método: **Eclat**(F_k)

```
1.  para todo itemset I_i ∈ F_k {
2.      F_{k+1} = Ø;
3.      para todo itemset I_j ∈ F_k, i < j {
4.          N = (I_i ⋂ I_j);
5.          se N.suporte ≥ min_sup
6.              F_{k+1} = F_{k+1} ⋃ {N}
7.          }
8.      se F_{k+1} ≠ Ø
9.          Eclat(F_{k+1});
10. }
```

Figura 2.15 – Análise dos *itemsets* gerados utilizando o *Eclat.*

No passo 4 do algoritmo o *Eclat* utiliza intersecções de *tidlists* e a otimização "intersecções rápidas".

2.4. REGRAS DE ASSOCIAÇÃO GENERALIZADAS

Dado um grande banco de dados, onde cada transação consiste em um conjunto de itens, e uma taxonomia (ou hierarquia "é-um*"*) entre os itens, é possível encontrar associações entre os itens em qualquer nível da taxonomia [SRIKANT; AGRAWAL, 1995].

Um exemplo de taxonomia é mostrado na Figura 2.16. Trata-se de uma hierarquia de *roupas*, categorizadas como sendo *agasalhos* (consistindo de *jaquetas* e *calças*) e *camisas*. E uma outra hierarquia de *calçados*, agrupando *sapatos* e *botas*. Para o usuário, é interessante que as regras geradas transponham os diversos níveis da taxonomia [SRIKANT; AGRAWAL, 1995]. Por exemplo, poder-se-ia inferir que pessoas que compram agasalhos compram botas, a partir do fato de que pessoas que compram jaquetas compram botas e de que pessoas que compram calças compram botas. No entanto, o suporte da regra "*agasalhos* → *botas"* pode não ser a soma dos

suportes das regras "*jaquetas* → *botas*" e "*calças* → *botas*", já que algumas pessoas podem ter comprado jaquetas, calças e botas na mesma transação. Além disso, "*agasalhos* → *botas*" pode ser uma regra válida (caso atinja os valores mínimos de suporte e confiança), enquanto que as regras "*jaquetas* → *botas"* e "*roupas* → *botas*" podem não ser. A primeira delas pode não possuir suporte mínimo, e a outra pode não possuir confiança mínima.

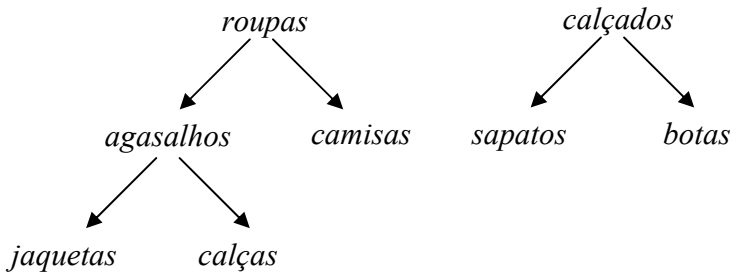

Figura 2.16 – Exemplo de uma taxonomia [SRIKANT; AGRAWAL, 1995].

Para resolver esse problema, a solução mais óbvia é adicionar à transação todos os ancestrais[1] de cada item que a ela pertence, e então executar o algoritmo de mineração de regras de associação sobre essas "transações estendidas" [SRIKANT; AGRAWAL, 1995]. Porém essa estratégia requer que o algoritmo seja ainda mais rápido, e em busca desse objetivo, novos algoritmos têm sido criados para minerar regras de associação generalizadas.

2.5. MINERAÇÃO DE DADOS QUANTITATIVOS

Algoritmos como o *Apriori* (apresentado na seção 2.3.1) possuem uma lógica que basicamente funciona da seguinte maneira: o valor de um atributo para um dado registro é "1" se o item relativo ao atributo está presente na transação correspondente ao registro, e "0" caso contrário. Por isso, as regras de associação obtidas dessa forma são conhecidas como *regras de associação booleanas*. Na maioria das aplicações científicas e comerciais, porém, os bancos de dados possuem tabelas relacionais que contêm maior riqueza de tipos de atributos: eles podem ser quantitativos (como idade e rendimento, por exemplo) ou categóricos (como CEP e marca do

[1] Os ancestrais de um item são todos os itens acima dele na hierarquia. No exemplo da Figura 2.16, por exemplo, os ancestrais do item "jaquetas" são os itens "agasalhos" e "roupas".

carro, por exemplo). Dessa forma, tratar todos os atributos como se eles fossem *booleanos* implica na perda de informações.

Por isso, para tratar regras de associação quantitativas (envolvendo atributos quantitativos e categóricos) Srikant e Agrawal [SRIKANT; AGRAWAL, 1996] desenvolveram um algoritmo que aproveita a estrutura básica do *Apriori*, mas que para obter uma implementação mais rápida, tem novos detalhes computacionais de como os candidatos são gerados e como os suportes deles são contados. A idéia é mapear os atributos categóricos e quantitativos para conjuntos de inteiros consecutivos. Para os atributos categóricos o mapeamento é direto: para cada categoria, um valor inteiro. Já para os atributos quantitativos, primeiramente há um particionamento em intervalos, e então cada intervalo é mapeado para um valor inteiro. Esse particionamento acaba gerando a possibilidade de alguns problemas:

- se os intervalos são muito grandes, algumas regras podem não ter confiança mínima
- se os intervalos são muito pequenos, algumas regras podem não ter suporte mínimo

Uma possível solução para os problemas acima é combinar intervalos ou valores adjacentes. No entanto, essa solução pode apresentar efeitos colaterais:

- tempo de execução: se o atributo quantitativo possuir muitos valores, o número de itens por registro aumenta muito, o que aumenta também o tempo de execução.
- número de regras: se um valor (ou intervalo) de um atributo quantitativo tem suporte mínimo, então qualquer intervalo que o contenha também terá. Assim, o número de regras aumenta muito (muitas das regras podem não ser interessantes).

Seguindo essa abordagem, para decidir quando particionar um atributo quantitativo ou não, e quantas partições deve haver, são estabelecidas métricas:

- *partial completeness measure* (métrica de integridade parcial): indica a quantidade de informação perdida no particionamento. O princípio desta métrica é que quanto mais longe (em termos comparativos) a regra mais próxima estiver, maior a perda de informações no particionamento. Esta métrica é usada para decidir particionar ou não um atributo quantitativo, e em caso afirmativo, o número de partições.
- *interest measure* (métrica de interesse de regras): a noção de "interesse" é expressa através de um parâmetro dado pelo usuário, que assim estipula quais devem ser os valores

de suporte e/ou confiança nas regras descobertas. Isso evita que a aplicação direta da métrica de integridade parcial resulte em muitas regras similares.

O tratamento de atributos categóricos e quantitativos é feito de maneira uniforme, no sentido que são representados de uma forma única após o mapeamento. Quando se tratam de atributos categóricos, os valores do atributo são mapeados para um conjunto de inteiros consecutivos. Para atributos quantitativos, têm-se duas possibilidades: se não estiverem particionados em intervalos, então os valores são mapeados para inteiros consecutivos de forma que a ordem dos valores seja preservada. Se estiverem, então os intervalos são mapeados para inteiros consecutivos, de forma que a ordem dos intervalos seja preservada. Assim, o mapeamento leva a um conjunto de pares `<atributo, valor inteiro>`, que é tratado de forma transparente pelo algoritmo.

O fato de que os itens[1] em uma regra podem ser quantitativos ou categóricos não aparece na definição de regra de associação feita em [SRIKANT; AGRAWAL, 1996]. O problema que a mineração busca solucionar é o de encontrar todas as regras que satisfaçam suporte e confiança mínimos (*minsup* e *minconf*, respectivamente). Assim, o algoritmo é executado da mesma forma, seja qual for a natureza dos atributos (quantitativos ou categóricos).

2.6. CONSIDERAÇÕES FINAIS

A mineração de dados tem sido largamente utilizada em diversas aplicações. Em grande parte delas, a tarefa realizada é a mineração de regras de associação, e quatro algoritmos que a efetuam foram apresentados neste capítulo, cada um deles representando uma estratégia diferente. Além disso, a mineração de regras de associação generalizadas e a mineração de dados quantitativos também foram discutidas.

Por maior que seja o avanço da pesquisa nessa área, a solução de alguns problemas na mineração de dados permanece em aberto. O uso da lógica nebulosa na mineração de dados surge como uma alternativa para a solução de alguns desses problemas. Os princípios da lógica

[1] item aqui é uma tripla que pode representar tanto um atributo categórico quanto um atributo quantitativo com seu intervalo (o valor de um atributo quantitativo pode ser representado como um intervalo onde os limites superior e inferior são os mesmos).

nebulosa são apresentados no capítulo 3, e algumas abordagens usando a lógica nebulosa na mineração de dados são mostradas no capítulo 4.

3. LÓGICA NEBULOSA

3.1. INTRODUÇÃO

A Lógica Nebulosa — também conhecida como Lógica *Fuzzy* (*Fuzzy Logic*) ou Lógica Difusa — baseia-se na teoria de conjuntos nebulosos (*Fuzzy Sets*), e teve seus conceitos e princípios introduzidos por Zadeh [ZADEH, 1987a] na década de 60. Ela trata matematicamente informações imprecisas usualmente empregadas na comunicação humana, permitindo inferir uma resposta aproximada para uma questão baseada em um conhecimento que é inexato, incompleto ou não totalmente confiável. É nesta natureza da informação que reside a nebulosidade. Enquanto na lógica booleana, usualmente empregada em computação, são definidos apenas dois valores possíveis — verdadeiro (1) ou falso (0) —, a Lógica Nebulosa é multivalorada (ou seja, há um conjunto de valores possíveis) e nesse aspecto ela pode ser considerada uma extensão da primeira.

3.2. CONJUNTOS NEBULOSOS

A função característica de um conjunto clássico pode assumir somente os valores 0 ou 1, determinando dessa forma quem são os membros e os não-membros desse conjunto. Essa função pode ser generalizada de forma que ela possa assumir valores em um determinado intervalo, e o valor assumido indica o *grau de pertinência* do elemento no conjunto em questão. Essa função é chamada de *função de pertinência* [KLIR; YUAN, 1995].

A função de pertinência de um conjunto nebuloso A é denotada por μ_A, desta forma:

$$\mu_A : X \rightarrow [0,1].$$

A função de pertinência mapeia os elementos de um conjunto clássico X em números reais no intervalo [0,1]. Assim, um conjunto nebuloso A é caracterizado por uma função de pertinência $\mu_A(x)$, que associa a cada elemento do conjunto um número real no intervalo [0,1]. Desta forma, o valor de $\mu_A(x)$ representa o grau de pertinência do elemento x no conjunto A. Quanto maior o valor de $\mu_A(x)$, maior o grau de pertinência de x no conjunto A.

Suponha-se, como exemplo, 3 conjuntos nebulosos que representam os conceitos de jovem, adulto e idoso. As funções de pertinência para cada um desses conjuntos (respectivamente $\mu_J(x)$, $\mu_A(x)$ e $\mu_I(x)$) são exibidas na Figura 3.1.

$\mu_J(x) = \begin{cases} 1 \\ (35-x)/15 \\ 0 \end{cases}$	Quando x ≤ 20 Quando 20 < x < 35 Quando x ≥ 35
$\mu_A(x) = \begin{cases} 0 \\ (x-20)/15 \\ (60-x)/15 \\ 1 \end{cases}$	Quando x ≤ 20 ou x ≥ 60 Quando 20 < x < 35 Quando 45 < x < 60 Quando 35 ≤ x ≤ 45
$\mu_I(x) = \begin{cases} 0 \\ (x-45)/15 \\ 1 \end{cases}$	Quando x ≤ 45 Quando 45 < x < 60 Quando x ≥ 60

Figura 3.1 – Funções de pertinência para os conjuntos jovem, adulto e idoso [KLIR; YUAN, 1995].

Nesse exemplo, o conjunto clássico *X* que contém as idades de 0 a 80 é mapeado, através da função de pertinência, em conjuntos nebulosos, que assumem valores no intervalo [0,1]. Uma representação gráfica desses conjuntos é mostrada na Figura 3.2.

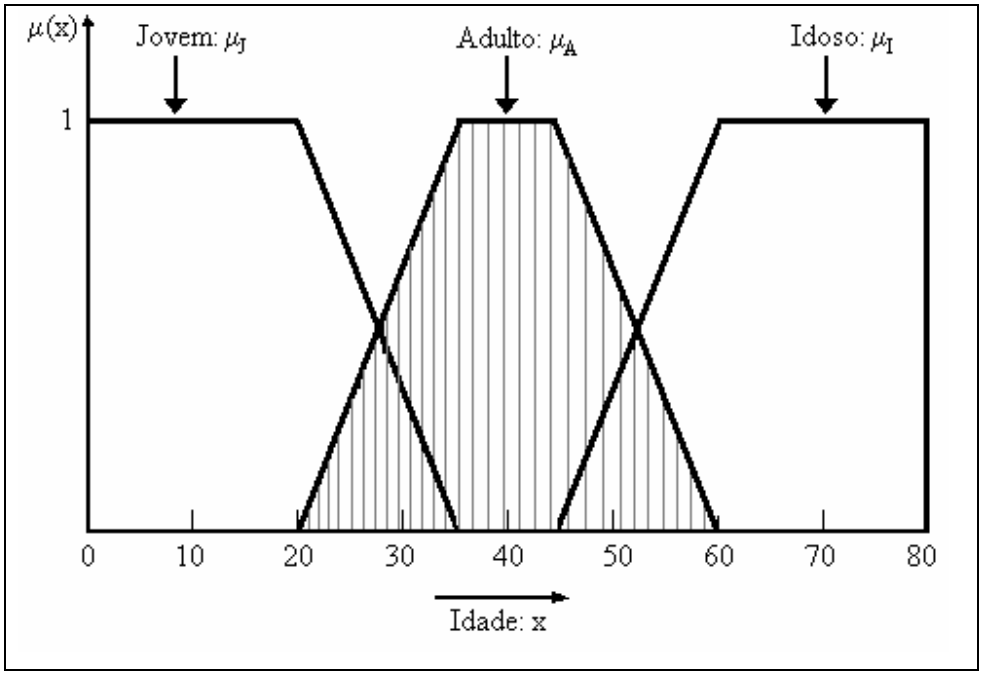

Figura 3.2 – Conjuntos representando os conceitos de jovem, adulto e idoso [KLIR; YUAN, 1995].

Assim como ocorre para conjuntos clássicos, também são definidas operações entre conjuntos nebulosos. Os conceitos de união ($\bigcup$) e intersecção ($\bigcap$) de conjuntos clássicos, por exemplo, podem ser estendidos para conjuntos nebulosos pelas seguintes fórmulas, propostas por Zadeh [ZADEH, 1987a]:

$$\forall x \in X, \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)),$$

$$\forall x \in X, \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)),$$

onde $\mu_{A \cup B}(x)$ e $\mu_{A \cap B}(x)$ são respectivamente as funções de pertinência de $A \bigcup B$ e $A \bigcap B$, e max e min são respectivamente os operadores de máximo e mínimo.

Os elementos de dois ou mais conjuntos nebulosos podem ser relacionados para representar presença ou ausência de associação, interação ou interconexão entre eles. Essas relações são chamadas de *relações nebulosas*, e são apresentadas na seção 3.3.

3.3. RELAÇÕES NEBULOSAS

As relações nebulosas podem envolver diversos conjuntos nebulosos, e quando envolvem dois conjuntos são chamadas *binárias*. É possível, no entanto, definir uma relação binária entre elementos de um único conjunto *X*, denotada por *R(X,X)*. Essa relação também é conhecida como *grafo direcionado* ou *dígrafo* [KLIR; YUAN, 1995], e é caracterizada por três propriedades: *reflexividade*, *simetria* e *transitividade* [ZADEH, 1987b].

Uma relação binária é *reflexiva* se e somente se

$$R(x,x) = 1$$

para qualquer $x \in X$. Ela é *simétrica* se e somente se

$$R(x,y) = R(y,x)$$

para qualquer x e $y \in X$, e é *transitiva* (ou mais especificamente, *max-min transitiva*) se e somente se

$$R(x,z) \geq \max_{y \in Y} \min[R(x,y), R(y,z)]$$

é satisfeito para cada par $\langle x,z \rangle \in Z$. Na expressão, *max* é o operador de máximo (que obtém o valor máximo da relação) e *min* é o operador de mínimo (que obtém o valor mínimo da relação). A expressão que define a transitividade é resultado da composição de duas relações nebulosas, e nesse caso essa composição é a *max-min*. Alternativamente, outras definições de transitividade nebulosa são possíveis e úteis em algumas aplicações [KLIR; YUAN, 1995], [DVORAK; SEDA, 2004].

Uma relação nebulosa binária que é reflexiva, simétrica e transitiva é conhecida como *relação nebulosa de equivalência* ou *relação de similaridade* [DUBOIS; PRADE, 1980], [PEDRYCZ; GOMIDE, 1998]. Segundo Zadeh [ZADEH, 1987b], "o conceito da relação de similaridade é essencialmente uma generalização do conceito da relação de equivalência".

Uma relação de similaridade é representada através de uma matriz de similaridade. A Tabela 3.1 apresenta uma matriz de similaridade para os elementos do conjunto nebuloso *Idade*.

Tabela 3.1 - Relação de similaridade entre os membros do conjunto nebuloso Idade.

	Muito Velho	Velho	Jovem	Muito Jovem	Criança
Muito Velho	1.0	0.7	0.0	0.0	0.0
Velho	0.7	1.0	0.0	0.0	0.0
Jovem	0.0	0.0	1.0	0.8	0.3
Muito Jovem	0.0	0.0	0.8	1.0	0.3
Criança	0.0	0.0	0.3	0.3	1.0

Uma relação nebulosa binária que é reflexiva e simétrica é geralmente chamada de *relação de proximidade*. Desta forma, trata-se de uma relação "menos restritiva" do que a relação de similaridade, já que ela não atende à propriedade transitiva.

3.4. CONSIDERAÇÕES FINAIS

Conceitos de lógica nebulosa são utilizados em diversas aplicações, com o objetivo de melhor representar as incertezas ou imprecisões geralmente existentes em situações do mundo real. Dessa forma, a lógica nebulosa fornece poderosas ferramentas para a computação de informações cujo significado não pode ser diretamente representado. Em aplicações de mineração de dados, muitas vezes a análise do conhecimento obtido pode ser influenciada pela qualidade de

representação das informações, e na tentativa de melhorá-la, o uso de conceitos da lógica nebulosa em mineração de dados tem sido bastante explorado.

Algumas abordagens usando lógica nebulosa na mineração de dados são apresentadas no capítulo 4 a seguir.

4. A LÓGICA NEBULOSA NA MINERAÇÃO DE DADOS

4.1. INTRODUÇÃO

A descoberta de conhecimento em bancos de dados está principalmente ligada à identificação de padrões interessantes e na descrição desses padrões de uma maneira concisa e significativa [PEDRYCZ, 1998]. Apesar da crescente versatilidade de sistemas de descoberta de conhecimento, a interação humana é um importante componente inerente a qualquer processo de representação, manipulação e processamento do conhecimento. O uso de conjuntos nebulosos, que são naturalmente inclinados a lidar com o conhecimento lingüístico do domínio, pode produzir soluções mais interpretáveis [MITRA; ACHARYA, 2003].

Por isso, é cada vez maior o número de trabalhos que utilizam a lógica nebulosa na mineração de dados. Algumas dessas abordagens são apresentadas na seção 4.2.

4.2. ABORDAGENS

A mineração de regras de associação envolvendo dados quantitativos (seção 2.5) requer a especificação de intervalos apropriados para cada atributo. No entanto, muitas vezes esses intervalos podem não ser suficientemente concisos e significativos. Assim, ao invés de usar intervalos, algumas abordagens empregam *termos lingüísticos*[1]. Em tese, a representação lingüística faz com que as regras descobertas sejam muito mais naturais para a compreensão humana. A definição de termos lingüísticos é baseada na teoria de conjuntos nebulosos (seção 3.2), e por isso diz-se que regras que possuem esses termos são *regras de associação nebulosas*. Termos lingüísticos são associados a atributos quantitativos, e o conjunto de termos lingüísticos é representado por um conjunto nebuloso.

Entre os algoritmos que realizam a mineração de regras de associação nebulosas, estão o *F-APACS* [CHAN; AU, 1998] e o *FARM* (*Fuzzy Association Rule Miner*) [AU; CHAN, 1999], ambos criados por Au e Chan. Nos algoritmos para mineração de dados quantitativos, a

[1] Termos lingüísticos são expressões ou modificadores lingüísticos usados para representar valores nebulosos. Exemplos de termos lingüísticos: “muito”, “pouco”, “bastante”, “muito pouco”.

identificação de regras interessantes é feita através de parâmetros definidos pelo usuário. Uma fragilidade dessa abordagem, segundo os autores do *F-APACS* e do *FARM*, é que muitos usuários não têm idéia de como estabelecer esses parâmetros. Se o seu valor for muito alto, o usuário pode perder regras úteis, e se o valor for muito baixo, o usuário pode ser "inundado" por muitas regras irrelevantes. Para resolver esse problema, tanto no *F-APACS* quanto no *FARM*, os termos lingüísticos são associados, e existe um cálculo chamado *diferença ajustada* cujo propósito é identificar quais dessas associações são interessantes, não sendo assim necessário o uso de um parâmetro definido pelo usuário. Assim que as associações entre termos lingüísticos são identificadas como sendo interessantes, a formação das regras de associação é realizada com base em uma métrica de confiança chamada *peso de evidência*, que é usada para representar a incerteza nas regras de associação nebulosas. O peso de evidência tem valor positivo se um valor de atributo (ou termo lingüístico) determina a presença de outro valor de atributo, e tem valor negativo se um valor de atributo determina a ausência de outro valor de atributo. Assim, pode-se dizer que os algoritmos *F-APACS* e *FARM* permitem a descoberta de regras de associação nebulosas *positivas* ou *negativas*, de acordo com seus pesos de evidência.

Outro trabalho envolvendo o uso de lógica nebulosa na mineração de dados quantitativos é o de Lee e Lee-Kwang [LEE; LEE-KWANG, 1997]. Através de conjuntos nebulosos definidos pelo usuário, as tuplas com dados quantitativos são estendidas e a seguir convertidas em tuplas com dados binários. Para finalizar, aplica-se um algoritmo convencional de mineração de regras de associação nas tuplas com os dados binários obtidos na conversão, e então são obtidas as *regras de associação estendidas*. Um exemplo de regra de associação estendida é

(*Hambúrguer*, \$5) → (*Refrigerante*, \$2)

que pode ser interpretada como "clientes que gastam \$5 com hambúrguer tendem a gastar \$2 com refrigerante". Como se pode notar no exemplo, as regras de associação estendidas lidam com pares (*atributo*, *valor*), ou seja, as regras de associações descobertas são entre pares (*atributo*, *valor*). Usando-se conjuntos nebulosos, é possível reduzir o número desses pares nas regras. Além disso, conjuntos nebulosos tornam a descrição das regras de associação concisas e generalizadas. Por exemplo, se houverem as seguintes regras de associação,

(*Hambúrguer*, \$5) → (*Refrigerante*, \$2)

(*Hambúrguer*, \$6) → (*Refrigerante*, \$3)

(*Hambúrguer*, $4) → (*Refrigerante*, $1.5)

elas podem ser escritas assim:

(*Hambúrguer*, *Médio*) → (*Refrigerante*, *Pequeno*)

Segundo os autores, dessa forma os usuários podem facilmente compreender as relações entre atributos, porque as regras de associação podem ser apresentadas em formas lingüísticas.

O trabalho de Kuok, Fu e Wong [KUOK;FU; WONG, 1998] também usa os conceitos de conjuntos nebulosos para minerar dados quantitativos. Nele, considera-se que os conjuntos nebulosos e as funções de pertinência correspondentes são fornecidos por especialistas. As regras de associação nebulosas possuem a forma

Se *X* é *A* então *Y* é *B,*

onde *X* e *Y* são atributos e *A* e *B* são conjuntos nebulosos que caracterizam *X* e *Y* respectivamente. São definidos dois fatores: a *significância* para *itemsets*, análoga ao suporte usualmente calculado para *itemsets* que não envolvem conjuntos nebulosos, e a *certeza* para regras de associação, análoga à confiança usualmente calculada para as regras de associação não nebulosas. A notação $\langle X, A \rangle$ representa um par *itemset* - *conjunto nebuloso*, onde $X = \{x_1, x_2, ..., x_p\}$ é um conjunto de atributos e $A = \{ f_{x_1}, f_{x_2}, ..., f_{x_p} \}$ contém os conjuntos nebulosos associados com os atributos correspondentes em *X*. O cálculo da significância leva em consideração o grau de pertinência de cada atributo de um registro no conjunto nebuloso correspondente, multiplica esses valores e obtém um valor para cada registro. Os valores de cada registro são somados e divididos pelo número total de registros. Os *itemsets* freqüentes são aqueles cuja significância é maior do que a significância mínima definida pelo usuário. Os *itemsets* freqüentes descobertos são usados para gerar todas as possíveis regras de associação, e para definir se a regra gerada é interessante ou não, devem ser consideradas a significância do *itemset* formado pela união do antecedente e do conseqüente da regra e a certeza da regra. Os autores oferecem duas opções para o cálculo da certeza: seu valor pode ser obtido através da significância ou através de uma correlação, diferente da definida na estatística, que é chamada de *CorrelaçãoXY*. Nos experimentos realizados, os autores constataram que o método que usa a significância para obter a certeza obteve melhor desempenho e que, por outro lado, o método que usa a correlação obteve resultados mais precisos.

O trabalho de Hong, Kuo e Chi [HONG; KUO; CHI, 1999] integra conceitos de conjuntos nebulosos com o algoritmo *Apriori*, e usa o resultado para encontrar itemsets interessantes e regras de associação nebulosas em dados quantitativos. É proposto um novo algoritmo de mineração, chamado de *FTDA* (*fuzzy transaction data-mining algorithm*). Ele transforma valores quantitativos nas transações em termos lingüísticos, e então os filtra para encontrar regras de associação através de uma modificação do algoritmo *Apriori*. Nela, os cálculos de suporte e confiança levam em consideração as funções de pertinência dos atributos quantitativos nos conjuntos nebulosos correspondentes.

Até aqui, foram apresentadas várias abordagens que utilizam a lógica nebulosa na mineração de dados quantitativos. Embora bastante abordada, essa não é a única forma de utilizar conceitos de conjuntos nebulosos na mineração de dados. A mineração de regras de associação generalizadas (seção 2.4) em taxonomias nebulosas é uma outra forma considerada em alguns trabalhos. A idéia central nesses trabalhos é a seguinte: enquanto que em uma taxonomia convencional assume-se que um filho pertence ao seu ancestral com grau 1, em uma taxonomia nebulosa esse grau corresponde a μ ($0 \leq \mu \leq 1$). Esse grau de pertinência de um item em uma taxonomia é levado em consideração na determinação dos graus de suporte e confiança.

O trabalho de Chen, Wei e Kerre [CHEN;WEI; KERRE, 2000], propõe a mineração em estruturas taxonômicas nebulosas. Segundo esses autores, em muitas aplicações do mundo real um item pode pertencer parcialmente a um ancestral na taxonomia. Por exemplo, um *tomate* pode ser considerado uma *fruta* ou um *vegetal*, embora em diferentes graus cada um. Um exemplo de estrutura taxonômica nebulosa é mostrado na Figura 4.1. Nela, um subitem pertence ao seu superitem com um certo grau. Nesse contexto, o cálculo do suporte e da confiança precisa ser estendido para considerar as características nebulosas da taxonomia. Os autores estenderam o algoritmo de mineração de regras generalizadas [SRIKANT; AGRAWAL, 1995], para incorporar os conceitos de suporte, confiança e interesse das regras considerando a taxonomia nebulosa. Esse algoritmo, no qual são considerados os graus de pertinência de cada subitem na taxonomia em relação a seu superitem para calcular o suporte e a confiança das regras obtidas, é chamado por eles de *Extended Algorithm*.

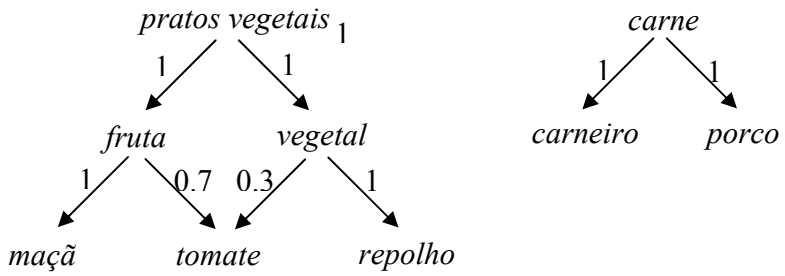

Figura 4.1 – Exemplo de taxonomia nebulosa [CHEN;WEI; KERRE, 2000].

Posteriormente, Chen e Wei desenvolveram outro trabalho envolvendo a mineração de regras generalizadas em taxonomias nebulosas [CHEN; WEI, 2002], mas dessa vez acrescentado o uso de *hedges lingüísticos*[1] nas regras de associação nebulosas, visando expressar mais naturalmente o conhecimento obtido. Um exemplo de regra de associação nebulosa com *hedges* lingüísticos é "produtos muito caros → tipo de fruta". Segundo os autores, há duas razões para se usar *hedges* lingüísticos na mineração de regras de associação nebulosas: a primeira é que o conhecimento descoberto se torna mais compreensível e próximo da linguagem humana. Para quem toma decisões, especialmente os gerentes dos mais altos níveis, esse tipo de conhecimento pode ser mais freqüentemente usado e significativo; a segunda razão é que o uso deles pode enriquecer a semântica das regras de associação e tornar as regras descobertas mais granulares. Podem ser obtidas regras como "maçã → jeans", "maçã cara → jeans legal" e "maçã muito cara → jeans mais ou menos legal", por exemplo. A aplicação de *hedges* lingüísticos modifica a taxonomia (pensando na taxonomia nebulosa da Figura 4.1, além do item "maçã" teria-se também os itens "maçã cara" ou "maçã muito cara", por exemplo), e então uma estratégia para construir uma nova estrutura taxonômica nebulosa é necessária. Após a construção dessa nova estrutura taxonômica, um algoritmo criado para minerá-la é aplicado. Os autores se referem ao algoritmo de mineração de regras de associação generalizadas clássico [SRIKANT; AGRAWAL, 1995] através da sigla *GAR* (*Generalized Association Rules*), e ao algoritmo por eles desenvolvido para lidar com taxonomias nebulosas [CHEN;WEI; KERRE, 2000] através da sigla *FGAR*. Seguindo a mesma

[1] *Hedges lingüísticos* são termos lingüísticos, como "muito", "mais ou menos", "tipo de", que acabam por modificar o significado do termo que o segue, tal qual um modificador lingüístico. Maiores detalhes podem ser vistos em [CHEN; WEI, 2002] e [DUBOIS; PRADE, 1980].

linha de nomenclatura, esse algoritmo que aplica *hedges* lingüísticos criando uma nova taxonomia nebulosa antes de minerá-la recebe o nome de *HFGAR*. Em testes experimentais, os autores comparam o desempenho do *GAR*, *FGAR* e *HFGAR*, e concluem que o *HFGAR* é o mais eficiente deles.

4.3. CONSIDERAÇÕES FINAIS

Ao longo da seção 4.2 foram apresentadas diversas abordagens que usam conceitos da lógica nebulosa na mineração de dados. Essas abordagens mostram que a teoria de conjuntos nebulosos, usada na definição de termos lingüísticos, é útil para tornar as regras de associação nebulosas mais concisas e compreensíveis, numa representação mais próxima da linguagem humana. Além disso, os conjuntos nebulosos podem ser usados para representar os intervalos de dados quantitativos, cujos limites podem não ser precisos, de uma forma mais natural. O conceito de função de pertinência em conjuntos nebulosos serve de ferramental para a mineração de regras de associação generalizadas em taxonomias nebulosas, e freqüentemente os graus de pertinência dos atributos nos conjuntos nebulosos são considerados para o cálculo do suporte e da confiança das regras de associação obtidas.

Os problemas encontrados na mineração de dados convencional motivaram o uso da lógica nebulosa na mineração de dados em cada uma das abordagens, e os resultados nelas obtidos encorajam o desenvolvimento de novos trabalhos usando essa tecnologia.

Até aqui foi discutida a mineração de dados classificados como sendo quantitativos ou categóricos. No capítulo 5 são apresentadas abordagens envolvendo a mineração de dados multimídia.

5. MINERAÇÃO DE DADOS MULTIMÍDIA

5.1. INTRODUÇÃO

A mineração de dados multimídia tornou-se uma subárea de grande importância dentro da área de mineração de dados, dada a crescente demanda de aplicações que utilizam dados multimídia. Imagem, áudio e vídeo, entre outros tipos de mídia, são cada vez mais presentes em aplicações, e conseqüentemente torna-se cada vez mais interessante extrair conhecimento a partir desse tipo de dado.

Dados multimídia são geralmente semi-estruturados, e muitas vezes não estruturados, principalmente quando comparados com as estruturas de dados numéricas e alfanuméricas que são geralmente utilizadas nos sistemas gerenciadores de bancos de dados tradicionais. A maior parte dos algoritmos e técnicas de mineração de dados clássicos foram desenvolvidos principalmente para minerar informações dos dados estruturados, como no caso de dados relacionais e transacionais, por exemplo. Eles têm sido usados tipicamente em análises financeiras e de negócios, previsões, etc. Entretanto, na realidade nem todos os tipos de dados são bem estruturados, e uma quantidade substancial de informação está disponível como dados multimídia semi-estruturados ou não estruturados, o que faz da mineração de dados multimídia um desafio [MITRA; ACHARYA, 2003].

Diferentes técnicas são utilizadas para minerar os dados multimídia. Em geral, realiza-se uma extração automática de características, e a partir delas são estabelecidas informações que são então mineradas usando mineração de regras de associação, classificação, *clustering* ou redes neurais.

Na seção 5.2 são apresentadas abordagens que utilizam essas diversas técnicas, ilustrando o panorama atual da pesquisa nessa área.

5.2. ABORDAGENS

Uma análise do desenvolvimento da mineração de dados multimídia leva aos principais tipos de mineração pesquisados: mineração de textos (*text mining*), mineração de imagens (*image*

mining) e mineração de vídeo (*video mining*). São significativos também os avanços na mineração da *Web*[1] (*Web mining*) e na mineração de áudio.

A *mineração de textos* consiste na mineração de grandes coleções de bancos de dados de textos. Há uma vasta quantidade de informação disponível na forma de publicações eletrônicas de livros, bibliotecas digitais, mensagens eletrônicas, documentos técnicos e de negócios, etc. Os mecanismos de busca na Internet (*search engines*), combinados com várias técnicas de análise de textos, têm ido na direção da mineração de textos *online* também [MITRA; ACHARYA, 2003].

Grande parte dos trabalhos em mineração de dados multimídia é referente à *mineração de imagens*. Dada sua importância, foi realizado por Zhang, Hsu e Lee [ZHANG;HSU; LEE, 2001] um estudo bastante interessante acerca do que se pode chamar de "estado da arte" da mineração de imagens. Segundo eles, a mineração de imagens trata da extração de conhecimento implícito, relacionamentos entre dados de imagens ou outros padrões não armazenados explicitamente nas imagens. A mineração de imagens é então mais do que somente uma extensão no domínio de imagens de mineração de dados; é um esforço interdisciplinar que exige experiência em processamento de imagens, recuperação de imagens, mineração de dados, aprendizado de máquina, banco de dados e inteligência artificial. Sistemas de mineração de dados que possam extrair automaticamente informação semântica significativa (conhecimento) a partir de imagens têm demanda cada vez maior. Apesar de ter essa característica multidisciplinar, a mineração de imagens tem escopo bem delimitado. Por definição, a mineração de imagens cuida da extração de padrões de imagens a partir de grandes coleções de imagens, e é isso que a diferencia de técnicas de processamento de imagens, cujo objetivo é entender e/ou extrair características específicas de uma única imagem. Na mineração de imagens, o objetivo é descobrir padrões de imagens que sejam significantes em uma dada coleção de imagens.

Os primeiros trabalhos em mineração de imagens focaram o desenvolvimento de um *framework* adequado para realizar a tarefa da mineração. Um banco de dados de imagens não processadas não pode ser diretamente usado para mineração. As imagens precisam passar antes por um "pré-processamento", que torne as informações adequadas para que a mineração seja

---

[1] *Web* é uma abreviação de *World Wide Web* (grande teia mundial), uma forma de se referir à rede mundial de computadores, a Internet.

mais efetiva. Basicamente dois tipos de *frameworks* aparecem em sistemas de mineração de imagens: *frameworks* dirigidos à função, que focam as funcionalidades dos diferentes módulos para organizar os sistemas de mineração de imagens; e *frameworks* dirigidos à informação, que são destinados a serem uma estrutura hierárquica com ênfase especial às informações nos vários níveis da hierarquia. A maior parte das arquiteturas de sistemas de mineração de imagens utiliza *frameworks* dirigidos à função. Neles, as informações são exclusivamente orientadas à aplicação e o *framework* é organizado de acordo com a funcionalidade do módulo. Já os *frameworks* dirigidos à informação têm como objetivo salientar o papel da informação nos diversos níveis de representação. Esse tipo de *framework* considera quatro níveis de informação:

- Nível de pixel – informações de baixo nível sobre a imagem;
- Nível de objeto – informações do objeto ou da região baseadas no nível de pixel;
- Nível de conceito semântico – considera o conhecimento do domínio para gerar conceitos semânticos de alto nível a partir dos objetos e regiões identificados;
- Nível de padrão e conhecimento – incorpora dados alfanuméricos relacionados ao domínio e os conceitos semânticos obtidos dos dados da imagem para descobrir padrões e conhecimento.

Entre as técnicas de mineração de imagens, destacam-se:

- reconhecimento de objetos
- recuperação de imagens
- indexação de imagens
- classificação de imagens
- agrupamento (*clustering*) de imagens
- mineração de regras de associação
- uso de redes neurais

A abordagem de Ordonez e Omiecinski [ORDONEZ; OMIECINSKI, 1999], por exemplo, trabalha com imagens geométricas simples e faz extração de características (cor, textura, formato e tamanho) a partir dessas imagens coloridas bidimensionais para realizar o reconhecimento de objetos. É apresentado um algoritmo, similar ao *Partition* (seção 2.3.2), para a mineração de associações no contexto das imagens, cujos quatro principais passos são:

1. Extração de características: imagens são segmentadas em regiões identificáveis por descritores de região (*blobs*). Idealmente um *blob* representa um objeto. Esse passo é também chamado de segmentação.
2. Identificação de objeto e criação de registro: objetos de uma imagem são comparados a objetos e todas as outras imagens. Cada objeto é identificado por um *id*. Esse passo é chamado de algoritmo de pré-processamento.
3. Criação de imagens auxiliares: imagens são geradas com objetos identificados para interpretar as regras de associação obtidas no próximo passo (criação da página html).
4. Execução do algoritmo de mineração de dados para produzir regras de associação de objetos.

O objetivo de Ordonez e Omiecinski foi comparar os resultados obtidos através da aplicação desse algoritmo com aqueles obtidos através da identificação manual das formas nas imagens. Ao final dos experimentos eles concluem que a mineração de imagens é possível para se obter regras simples a partir de imagens não complexas, com poucos objetos simples, mas que todavia a intervenção humana e algum conhecimento do domínio são necessários para se alcançar melhores resultados.

O trabalho de Anthoine, Zaïane e Coman [ANTHONIE;ZAÏANE; COMAN, 2001] tem por objetivo a classificação de imagens de mamografias, que podem pertencer a três categorias: normal, benigno ou maligno. A primeira categoria classifica pacientes saudáveis, enquanto que a segunda classifica pacientes que tenham tumores benignos e a terceira classifica pacientes que tenham tumores malignos. É feito um pré-processamento das imagens de mamogramas (limpeza e eliminação de ruídos dos dados). Após esse pré-processamento, é realizada a extração de características, que depende do tipo de tecido (denso, gorduroso ou gorduroso-glandular) e da posição do seio no exame (esquerda ou direita). É realizado um trabalho comparativo entre o uso de redes neurais e a mineração de regras de associação para obter conhecimento que auxilie os médicos na análise dos exames. Para descobrir regras entre as características extraídas do banco de dados de mamografias, aplica-se o algoritmo *Apriori* (seção 2.3.1) na mineração de regras de associação. As regras são da forma: *características* → *categoria* (que pode ser normal ou anormal). O *Apriori* é aplicado para encontrar regras dessa forma, e com essas restrições adicionais na forma de regras a serem descobertas, consegue-se gerar um conjunto relativamente

pequeno de regras que associem conjuntos de características a nomes de classes. Essas regras constituem o modelo de classificação, e a descoberta dessas regras representa a fase de treinamento do classificador. Para classificar um novo mamograma é necessário extrair as características da imagem (como foi feito com o conjunto de treinamento) e aplicar as regras de associação nas características extraídas para identificar a classe à qual o novo mamograma deve pertencer.

Já o trabalho de Djeraba [DJERABA, 2001] propõe um algoritmo que descobre "relacionamentos escondidos" entre características de imagens. Segundo o autor, há diferenças entre as características de imagens dos exemplos (por exemplo: cor, textura) e a semântica (por exemplo: "flores em frente a um lago") que o usuário está procurando. O que está faltando é a capacidade de se extrair conhecimento. Para Djeraba, o desafio é obter um sistema que possa extrair não somente as características visuais (cores e texturas), mas também os relacionamentos escondidos entre as características, para tornar possível a busca semântica. A proposta é um novo esquema para categorização hierárquica automática de imagens. São usadas características de baixo nível — correlogramas[1] para cores e descritores de Fourier[2] para texturas — que são, juntos, eficientes para a recuperação de imagens baseada em conteúdo. Os relacionamentos tornam mais precisas as classes de imagens (para posterior classificação), e os melhores relacionamentos são selecionados através de medidas de confiança (*probabilidade condicional* e *intensidade de implicação*). Segundo o autor, a descoberta de relacionamentos escondidos contribui para tornar a recuperação baseada em conteúdo mais eficiente. É proposto também um *framework*, para prover indexação e recuperação. Na indexação, os conteúdos das imagens são extraídos automaticamente. Os métodos podem identificar regiões relevantes em imagens e computar características como cor e textura ou computar características visuais da imagem toda. Os conteúdos extraídos são representados como (ou transformados em) modelos e estruturas de dados adequadas, e então armazenados em bancos de dados. Já a recuperação consiste na busca

---

[1] Correlograma é um gráfico que mostra a correlação de um sinal com outro (nesse caso, o sinal é obtido a partir da imagem).

[2]Técnica de processamento de imagens. Maiores detalhes em:

C. T. Zahn, R. Z. Roskies, "Fourier descriptors for plane closed curves", IEEE Trans. On Computers, 1972.

de imagens através da seleção de alvos ou propriedades de conteúdo tais como cor, textura ou regiões de imagens, ou combinações desses. O sistema inclui uma ferramenta de consulta visual que permite aos usuários formular uma consulta através de um desenho ou rascunho, selecionando cores e texturas. Destaque também para o *Visual Thesaurus*, que é criado por um algoritmo baseado em uma estratégia de *clustering* e agrupa imagens que possuam cores e texturas similares, reduzindo o número de combinações de relacionamentos. Os relacionamentos encontrados são da forma *premissa* → *conclusão*, com uma medida de confiança. A conclusão do autor é que os relacionamentos são muito úteis para a categorização automática de novas imagens durante a sua inserção em grandes bancos de dados, obtendo resultados com maior semântica e melhorando o processo de recuperação das imagens.

Outro trabalho nessa linha é o de Bianchi-Berthouze e Hayashi [BIANCHI-BERTHOUZE; HAYASHI, 2002], que fala do processo de mineração *Kansei*. Essa é uma palavra japonesa que significa "impressão subjetiva", e o processo leva esse nome porque o objetivo dessa abordagem é justamente contemplar aspectos subjetivos dos usuários na recuperação de informações através de conteúdo. Exemplos de consultas usando impressões subjetivas seriam "recupere imagens românticas de aviões" ou "traga imagens quentes da praia". É utilizada uma estratégia de *clustering*, sendo feito até mesmo um estudo de como imagens, por exemplo, permitem uma múltipla interpretação de seu conteúdo de acordo com os mecanismos de atenção e seleção que o cérebro humano usa para filtrar as informações.

Já a abordagem de Datcu e Seidel [DATCU; SEIDEL, 2002] realiza extração de características de imagens, e propõe o uso de um grande número de *clusters* que agrupem automaticamente as imagens inseridas. Os *clusters* são modelados parametricamente, reduzindo dados, e o sistema é treinado por um conjunto de exemplos, tendo um aprendizado de semântica conforme o usuário interage com ele.

5.3. CONSIDERAÇÕES FINAIS

Embora os pesquisadores estejam procurando formas cada vez mais eficientes e inovadoras de extrair características a partir das mídias, uma extração automática que permita a riqueza semântica desejada ainda é um desafio. As abordagens têm buscado obter descritores das mídias que sejam confiáveis e ao mesmo tempo rapidamente obtidos. A saída até agora tem sido adotar

uma aplicação específica, que contenha dados somente de um determinado tipo de mídia, para que a extração de características seja uma tarefa possível, embora muitas vezes não muito simples. Nota-se também que muitas das abordagens apresentadas têm como objetivo da mineração de dados tornar mais eficiente a recuperação das mídias em consultas.

Por ser a mineração de dados multimídia uma área de pesquisa multidisciplinar e bastante recente, não são poucos os obstáculos encontrados nos trabalhos nela desenvolvidos. Com o avanço da pesquisa, porém, eles tendem a se tornar cada vez menores, caminhando em direção ao objetivo de obter-se, através da extração de características da mídia, dados semânticos tão bons quantos os produzidos pelo ser humano, porém de uma forma mais rápida e confiável.

A melhoria do processo de mineração, com o provimento de maior riqueza semântica aos dados processados, é a motivação deste trabalho. Para tanto, foi desenvolvido um algoritmo que considera as similaridades semânticas entre os dados para obter as regras de associação. Esse algoritmo é apresentado no capítulo 6.

6. MINERAÇÃO DE DADOS SEMANTICAMENTE SIMILARES

6.1. INTRODUÇÃO

Os algoritmos de mineração de regras de associação conhecidos utilizam várias estratégias para obter associações entre itens que ocorram freqüentemente em uma base de dados, conforme foi apresentado nas seções 2.2 e 2.3. Utilizando diversas estruturas de dados, esses algoritmos são aplicados a dados classificados como categóricos (atributos textuais), ou quantitativos (atributos envolvendo números ou intervalos de números). Dessa classificação surgiram as definições de mineração de regras de associação generalizadas, que envolve dados categóricos em taxonomias (apresentada na seção 2.4), e de mineração de dados quantitativos (apresentada na seção 2.5).

Nos trabalhos apresentados no capítulo 4, conceitos de lógica nebulosa foram usados na mineração de dados, com resultados promissores. De acordo com o levantamento bibliográfico realizado, grande parte dessas abordagens destina-se à mineração de regras generalizadas ou à mineração de dados quantitativos. No primeiro caso considerando taxonomias nebulosas, e no segundo caso utilizando termos lingüísticos associados a conjuntos nebulosos para obter regras de associação nebulosas. Na abordagem deste trabalho, exposta a partir da seção 6.2, também foram utilizados conceitos da lógica nebulosa para a mineração de dados, porém de forma totalmente diferente das anteriores presentes na bibliografia estudada.

Na mineração de dados multimídia (capítulo 5), as diversas abordagens existentes diferem basicamente quanto à forma de extração das características da mídia. Essa extração pode ser ou não ser automática e, independentemente da forma pela qual ela é realizada, tem um objetivo claro: obter metadados, para então minerá-los. Assim, na mineração de dados multimídia, os trabalhos assumem que os metadados constituem um material mais rico para a descoberta de novas informações na mineração de dados do que os dados brutos em si. Na abordagem deste trabalho, os metadados considerados são as informações semânticas, e essa característica lhe confere um diferencial em relação aos trabalhos apresentados no capítulo 5, já que dessa forma a semântica entre os dados é considerada no processamento das informações.

O objetivo da mineração de dados é a descoberta de conhecimento, seja qual for o tipo dos dados minerados. A análise do significado dos dados minerados (ou seja, de sua semântica) naturalmente contribui para que a qualidade das informações obtidas através da mineração seja maior e, conseqüentemente, melhores devem ser as decisões guiadas por essas informações.

Na seção seguinte é apresentada a contextualização dos conceitos utilizados na mineração de dados semanticamente similares.

6.2. CONTEXTUALIZAÇÃO

Os algoritmos que se propõem a minerar dados categóricos geralmente analisam se a base de dados é composta por atributos desse tipo, e então reconhecem esses itens e verificam a sua ocorrência ao longo de toda a base, assim como a sua associação com outros itens. A questão é como se dá esse reconhecimento, pois "aos olhos" desses algoritmos, cada item é simplesmente uma seqüência de caracteres (*string*). Assim, itens como *pão* e *baguete*, por exemplo, seriam considerados completamente diferentes por eles. De fato, as *strings* são totalmente distintas, mas os itens representados por elas constituem noções bastante similares. Afinal, *pão* e *baguete* são utilizados para um mesmo fim (alimentação), têm aspecto semelhante, possuem basicamente os mesmos ingredientes e geralmente estão disponíveis para a venda no mesmo local. Trata-se de uma similaridade semântica entre os itens, ignorada por algoritmos convencionais, que dessa forma podem desperdiçar informações importantes. Minerar os dados sem considerar a similaridade semântica entre os itens *pão* e *baguete*, por exemplo, leva a regras de associação que envolvem esses itens apenas individualmente. Uma análise adicional que considere associações entre esses itens similares poderia levar à descoberta de outras regras de associação, possivelmente relevantes também.

Seguindo essa idéia, a *mineração de dados semanticamente similares* estende a forma de mineração de regras de associação convencional, de maneira que os itens semanticamente similares sejam considerados como se a sua associação constituísse um único item. A similaridade semântica entre os dados é traduzida através de um grau de similaridade entre os itens, de acordo com a definição de relação de similaridade apresentada na seção 3.3. Assim, se o grau de similaridade entre os itens possuir valor igual a 1, isso significa que a similaridade entre os itens comparados é máxima. De acordo com a propriedade reflexiva das relações nebulosas

binárias, isso só pode ocorrer se um item for comparado a ele mesmo. Portanto, sempre que dois itens não idênticos forem comparados, o valor do grau de similaridade terá um valor maior ou igual a zero e menor do que 1. Pensando na mineração desses itens, considere novamente *pão* e *baguete* como exemplo e suponha que o grau de similaridade entre eles seja 0.8. Em uma avaliação simples, pode-se concluir que não seria adequado simplesmente somar os suportes obtidos por cada um dos itens, já que eles não são idênticos, mas similares. Para que o valor do suporte da associação entre eles reflita a influência de ambos, o grau de similaridade entre os itens precisa ser considerado no cálculo do suporte das regras obtidas.

Para realizar a mineração de dados semanticamente similares, foi criado o algoritmo *SSDM* (*Semantically Similar Data Miner*), descrito na próxima seção (6.3).

6.3. O ALGORITMO *SSDM*

O algoritmo *SSDM* teve como base o algoritmo *Apriori*, descrito na seção 2.3.1.

Sendo um algoritmo de mineração de regras de associação, ele necessita dos parâmetros de suporte mínimo e confiança mínima. Além disso, pela adição do uso de conceitos de lógica nebulosa, um parâmetro indicando o grau de similaridade mínimo desejado também se fez necessário e recebeu o nome de *minsim*.

Dessa forma, têm-se os seguintes parâmetros:

- *minsup*, que indica o suporte mínimo desejado;
- *minconf*, que representa a confiança mínima definida;
- *minsim,* que é o grau de similaridade mínimo necessário entre itens para que eles sejam considerados suficientemente similares e assim sejam associados durante a execução do algoritmo.

Esses parâmetros são fornecidos pelo usuário, sendo expressos por um valor real no intervalo [0,1].

As etapas realizadas pelo algoritmo podem ser divididas conforme mostra a Figura 6.1.

1. **Varredura da base: identificação dos itens e seus domínios**
2. **Determinação das similaridades entre itens em cada domínio**
3. **Identificação de itens similares**
4. Geração de candidatos
5. **Pesagem dos candidatos**
6. Avaliação dos candidatos
7. Geração de regras

Figura 6.1 – Etapas do algoritmo *SSDM*.

As etapas destacadas em negrito foram criadas para permitir a mineração de dados semanticamente similares. Para um melhor entendimento do algoritmo, sua apresentação será acompanhada por um exemplo, mostrando como o *SSDM* realiza a mineração de uma tabela de transações de compras em um supermercado (Tabela 6.1).

Tabela 6.1 – Transações de compras em um supermercado

Tid	Dom_1	Dom_2	Dom_3
10	Broa	Leite	Torta
20	Pão	Refrigerante	Sorvete
30	Baguete	Leite	Torta
40	Pão	Refrigerante	Sorvete
50	Broa	Suco	Torta
60	Broa	Suco	Sorvete
70	Broa	Refrigerante	Sorvete
80	Baguete	Suco	Biscoito
90	Broa	Refrigerante	Biscoito
100	Pão	Refrigerante	Sorvete

Note que na primeira coluna da tabela existe um identificador para cada transação (*Tid*) e que as demais colunas (Dom_1, Dom_2, Dom_3) contém itens comprados por clientes do supermercado.

Além disso, suponha que os valores dos parâmetros são os seguintes:

Suporte mínimo (*minsup*) = 0.45

Confiança mínima (*minconf*) = 0.3

Similaridade mínima (*minsim*) = 0.7

As etapas do *SSDM* são detalhadas nas seções seguintes (6.3.1 a 6.3.7), e nelas esse exemplo é sempre referido como “exemplo do supermercado”.

6.3.1. Varredura da base

Primeiramente é feita uma varredura na base de dados, para a identificação dos itens. O algoritmo identifica cada item da base, gerando *itemsets* de tamanho unitário. No *SSDM* cada item é associado a um domínio de acordo com a coluna em que aparece. Essa definição dos domínios é importante para que seja possível relacionar os itens quanto à sua similaridade somente quando isso for conveniente, ou seja, quando pertencerem a um mesmo domínio.

Assim, considerando o exemplo do supermercado, após a varredura os itens e seus domínios são identificados (Tabela 6.2). Nesse caso, a coluna à qual cada item pertence corresponde ao seu domínio.

Tabela 6.2 – Itens e domínios identificados na varredura

Itens	*Domínio*
Pão, Broa, Baguete	Dom_1
Suco, Refrigerante, Leite	Dom_2
Biscoito, Sorvete, Torta	Dom_3

Note que nesse exemplo o domínio Dom_1 contém itens de padaria, o domínio Dom_2 contém itens de bebidas e o domínio Dom_3 contém itens de sobremesas, tornando a identificação dos domínios coerente semanticamente.

O número de itens pertencentes ao domínio determina o seu tamanho. Assim, todos os três domínios que aparecem na Tabela 6.2 possuem tamanho 3.

6.3.2. Determinação das similaridades

Identificados os itens e seus domínios, é preciso determinar os valores das relações de similaridade a eles associados. Esses valores devem ser fornecidos por um especialista no domínio (geralmente o próprio usuário). Deve-se observar que esse passo compreende um dos

passos do *KDD*, anteriores ao passo de mineração de dados, mostrados na seção 2.1. Alternativamente, seria possível obter esses valores automaticamente, através de alguma regra ou método. Note-se, porém, que para que a semântica seja considerada na determinação dos valores de similaridade entre os itens, é necessário que se adote uma forma de se reproduzir, com bastante fidelidade, a capacidade que a mente humana tem de fazê-lo. Qualquer regra escolhida para se determinar esses valores automaticamente levará em conta fatores não semânticos, empobrecendo a análise realizada e dessa forma caminhando no sentido contrário ao do objetivo da mineração de dados semanticamente similares, que é enriquecer a análise e, por conseqüência, as informações obtidas a partir das regras.

Em cada domínio, os valores das similaridades entre itens obtidos são armazenados em uma matriz de similaridade. No exemplo do supermercado, três deles foram identificados, e as matrizes de similaridade correspondentes podem ser vistas na Figura 6.2.

Domínio: Dom_1 (Padaria)

	Pão	Broa	Baguete
Pão	1	0.9	0.8
Broa	0.9	1	0.7
Baguete	0.8	0.7	1

Domínio: Dom_2 (Bebidas)

	Suco	Leite	Refrigerante
Suco	1	0.4	0.3
Leite	0.4	1	0.2
Refrigerante	0.3	0.2	1

Domínio: *Dom3* (Sobremesas)

	Biscoito	Torta	Sorvete
Biscoito	1	0.6	0.4
Torta	0.6	1	0.3
Sorvete	0.4	0.3	1

Figura 6.2 – Domínios e suas respectivas matrizes de similaridade.

Cada matriz de similaridade é consultada durante a etapa de identificação de itens similares (seção 6.3.3).

6.3.3. Identificação de itens similares

Nesta etapa as matrizes de similaridade de cada domínio são analisadas, identificando os pares de itens cujo grau de similaridade seja maior ou igual ao mínimo esperado (*minsim*). Esses pares de itens formam *associações nebulosas* de tamanho 2. Essas associações são expressas no algoritmo através de *itens nebulosos*, que são representações onde o símbolo ~ é utilizado para indicar a relação entre os itens. Assim, supondo que os itens suficientemente similares são $item_1$ e

$item_2$, por exemplo, um item nebuloso da forma $item_1$~$item_2$ representa a associação nebulosa entre eles.

No exemplo do supermercado, são analisadas as matrizes da Figura 6.2 e, considerando o valor de *minsim* (0.7), são obtidas as associações exibidas na Tabela 6.3. Note-se que nos domínios Dom_2 e Dom_3 não são encontrados valores de similaridade maiores ou iguais a 0.7 e, dessa forma, somente no domínio Dom_1 são estabelecidas associações nebulosas.

Tabela 6.3 – Relações de similaridade que atendem ao valor de *minsim*

Domínio	*Valor*	*Relação de similaridade*[1]	*Item nebuloso equivalente*
Dom_1	0.9	sim(Pão, Broa)	Pão~Broa
Dom_1	0.8	sim(Pão, Baguete)	Pão~Baguete
Dom_1	0.7	sim(Broa, Baguete)	Broa~Baguete

Após a obtenção do conjunto de associações nebulosas de tamanho 2, é verificada a existência de *ciclos de similaridades*. Esses ciclos são associações nebulosas de tamanho maior do que 2, e para que eles existam, todos os itens que o compõem devem ser, dois a dois, suficientemente similares. Ou seja, de acordo com a teoria da intersecção entre conjuntos nebulosos (seção 3.2), o valor mínimo entre os graus de similaridade envolvidos no ciclo deve ser maior ou igual a *minsim*. É o que ocorre, no exemplo do supermercado, com o ciclo Pão~Broa~Baguete, mostrado na Figura 6.3. Nela, as setas representam as relações de similaridade entre os itens, e os valores que as acompanham expressam o valor dessas relações. Assim, pode-se constatar que nesse exemplo todos os itens são, dois a dois, suficientemente similares ($0.9 \geq 0.7$, $0.8 \geq 0.7$ e $0.7 \geq 0.7$). Ou, de outra forma, pode-se verificar que o mínimo entre os graus de similaridade envolvidos é maior ou igual a *minsim* ($\min(0.9, 0.8, 0.7) \geq 0.7$).

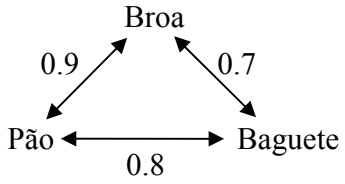

Figura 6.3 – Ciclo de similaridades.

[1] A notação sim($item_1$,$item_2$) representa a relação de similaridade entre $item_1$ e $item_2$.

O tamanho mínimo de um ciclo de similaridades é 3, e o seu tamanho máximo é o mesmo do domínio analisado. No exemplo do supermercado, o tamanho do domínio Dom_1 é 3, e por isso nele nenhuma associação nebulosa de tamanho maior do que 3 pode ser obtida. Porém, para domínios maiores, podem existir ciclos de tamanho maior. Por isso, em cada domínio, o *SSDM* verifica a existência dos ciclos de similaridades de forma iterativa, onde a cada passo k ($k \in$ N, $k \geq 3$) são analisados os conjuntos de associações de tamanho k-1, para obter associações de tamanho k. Essa estratégia é descrita no algoritmo da Figura 6.4, e a notação nele utilizada é apresentada na Tabela 6.4.

1	A_2 = {Conjunto das associações nebulosas de tamanho 2}
2	**para** (k = 3; k < tamanho(D_n) ; k++)
3	**comparação** das associações em A_{k-1}, duas a duas
4	**se** (prefixo(a_i) = prefixo(a_j), $i \neq j$)
5	//se a união dos sufixos é suficientemente similar
6	**se** ((sufixo(a_i) $\cup$ sufixo(a_j)) $\in A_2$)
7	$a_k = a_i \cup$ sufixo(a_j)
8	**fim se**
9	**fim se**
10	**fim da comparação**
11	A_k = {conjunto de todos os a_k}
12	**fim de para**
13	S_n = Reunião de todos os A_k

Figura 6.4 – Algoritmo para descobrir os ciclos de similaridades.

Tabela 6.4 - Notação utilizada na Figura 6.4.

A_k	Conjunto de associações nebulosas de tamanho k
D_n	Cada um dos n domínios analisados
a_k	Cada um das associações nebulosas do conjunto A_k
S_n	Conjunto dos itens similares obtidos no domínio n
tamanho(D_n)	Tamanho de D_n
prefixo(a_k)	Prefixo da associação nebulosa a_k
sufixo(a_k)	Sufixo da associação nebulosa a_k

Uma associação nebulosa a_k é da forma $\{s_1, s_2, ..., s_{k-1}, s_k\}$, onde $s_1, s_2, ..., s_{k-1}, s_k$ são os k itens que a compõem. O seu sufixo tem a forma $\{s_k\}$, enquanto que seu prefixo tem a forma $\{s_1, s_2, ...,s_{k-1}\}$.

Todos os A_k obtidos nessa etapa são reunidos em S_n. Assim termina a etapa de identificação de itens similares, e então outra parte iterativa do algoritmo tem início. Nela, para cada passo k ($k \in$ N), é feita a geração dos *itemsets* candidatos (seção 6.3.4) de tamanho k, a partir dos itens freqüentes do passo (k-1) anterior. Também é feita a pesagem dos candidatos (seção 6.3.5) de tamanho k e a avaliação desses candidatos (seção 6.3.6).

6.3.4. Geração de candidatos

A forma como são gerados os candidatos é similar à do *Apriori*. Porém, no *SSDM*, além dos itens identificados durante a varredura da base (seção 6.3.1), os itens nebulosos, que representam as associações nebulosas obtidas na etapa de identificação de itens similares (seção 6.3.3), também integram os candidatos gerados.

Ao final desta etapa, tem-se o conjunto de *itemsets* candidatos de tamanho k, que são submetidos à etapa de pesagem dos candidatos (seção 6.3.5).

6.3.5. Pesagem dos candidatos

Nessa etapa é computado o *peso* de cada um dos *itemsets* candidatos, que corresponde ao seu número de ocorrências na base de dados. No *SSDM*, diferentemente do que ocorre no *Apriori*, um *itemset* pode possuir itens nebulosos, e nesse caso, ele é chamado de *itemset nebuloso*. Lembrando a notação para itens nebulosos apresentada na seção 6.3.3, $item_1 \sim item_2$ guarda em si o seguinte significado: se $item_1$ e $item_2$ são muito similares, eles podem ser considerados praticamente iguais; assim, se forem encontradas ocorrências de $item_1$ ou $item_2$ na base de dados, elas serão associadas e, juntamente com o valor da relação de similaridade entre os itens, comporão uma *ocorrência nebulosa* de $item_1 \sim item_2$, em oposição às ocorrências exatas encontradas por algoritmos convencionais.

É necessário, portanto, identificar se o *itemset* é nebuloso ou não antes de realizar a sua pesagem: se o *itemset* não é nebuloso, a pesagem é feita de maneira convencional, contando-se suas ocorrências exatas; se o *itemset* é nebuloso, a pesagem deve considerar suas ocorrências nebulosas. Para entender como acontecem essas ocorrências, suponha que o valor da relação de similaridade entre $item_1$ e $item_2$ é de 0.8. Nesse caso, cada ocorrência de $item_2$ na base de dados pode ser considerada igual a 80% da ocorrência de $item_1$. Assim, a cada ocorrência de $item_1$ soma-se uma ocorrência de $item_1$ (naturalmente), e a cada ocorrência de $item_2$ soma-se 0.8 ocorrência de $item_1$ (Figura 6.5 - situação *A*).

Tid	Dom_1	
10	$item_1$	1.0
20	$item_1$	1.0
30	$item_2$	0.8

situação A

Tid	Dom_1	
10	$item_1$	0.8
20	$item_1$	0.8
30	$item_2$	1.0

situação B

Figura 6.5 – Ocorrências nebulosas.

O problema pode ser visto também de forma inversa, somando-se uma ocorrência de $item_2$ a cada ocorrência de $item_2$, e 0.8 a cada ocorrência de $item_1$ (Figura 6.5 - situação *B*). Note-se que para a situação *A* as ocorrências nebulosas totalizam o valor de 2.8 (1.0 + 1.0 + 0.8), enquanto que para a situação *B* elas somam o valor de 2.6 (0.8 + 0.8 + 1.0). Assim, dependendo da escolha da situação, o resultado obtido para os mesmos itens similares poderia ser diferente. Para evitar essa distorção, é necessário balancear essa contagem.

Isso pode ser feito da seguinte forma: considere *peso(item$_1$)* o número de ocorrências de *item$_1$*; *peso(item$_2$)* o número de ocorrências de *item$_2$*; e *sim(item$_1$,item$_2$)* o valor da relação de similaridade entre *item$_1$* e *item$_2$*. Assim, para a situação *A* da Figura 6.5, o número de ocorrências é dado pela expressão

$$peso(item_1) + peso(item_2) \times sim(item_1, item_2).$$

Analogamente, para a situação *B*, o número de ocorrências é obtido pela expressão

$$peso(item_1) \times sim(item_1, item_2) + peso(item_2).$$

Para o balanceamento da contagem das ocorrências nebulosas, adota-se a média aritmética entre as situações *A* e *B*, de acordo com a expressão abaixo.

$$\frac{\left[peso(item_1) \times sim(item_1, item_2) + peso(item_2)\right] + \left[peso(item_1) + peso(item_2) \times sim(item_1, item_2)\right]}{2}.$$

Analisando-se as possibilidades de contagem das ocorrências, verifica-se também que o valor mínimo possível seria obtido se fosse somado o valor da similaridade entre os itens (0.8) a cada ocorrência de *item$_1$* ou *item$_2$* no exemplo da Figura 6.5. Dessa forma, o peso obtido nesse exemplo seria de 2.4 (0.8 + 0.8 + 0.8). Já o valor máximo possível, por sua vez, seria obtido se a cada ocorrência de *item$_1$* ou *item$_2$* fosse somado 1.0 (uma ocorrência exata), obtendo dessa forma o valor de 3.0 (1.0 + 1.0 + 1.0) para o peso. Considerando esses valores extremos, parece bastante razoável que o valor do peso nebuloso corresponda à média entre eles, sendo calculada através da expressão

$$\frac{\left[peso(item_1) \times sim(item_1, item_2) + peso(item_2) \times sim(item_1, item_2)\right] + \left[peso(item_1) + peso(item_2)\right]}{2},$$

cujo valor resultante nesse caso é 2.7. Esse é o mesmo valor obtido através do cálculo da média aritmética entre os valores obtidos nas situações *A* e *B* (2.8 e 2.6 respectivamente), o que reforça a coerência da adoção dessa média para balancear o a contagem das ocorrências nebulosas.

Pode-se constatar ainda que as expressões apresentadas para a média das situações *A* e *B* e para a média dos valores máximo e mínimo possíveis são equivalentes, o que pode ser comprovado através de simples manipulação algébrica. Usando-se esse mesmo artifício, chega-se à Equação 6.1, equivalente às duas expressões, criada para calcular o número de ocorrências nebulosas. O resultado obtido através desse cálculo é chamado de *peso nebuloso*.

$$Peso\ nebuloso = \frac{\left[peso(item_1) + peso(item_2)\right]\left[1 + sim(item_1, item_2)\right]}{2}$$

Equação 6.1 – Cálculo do peso nebuloso de dois itens similares.

Note, porém, que através da Equação 6.1 é obtido o peso nebuloso de apenas 2 itens similares. Para associações nebulosas envolvendo mais itens, é necessária uma generalização dessa equação. Sendo assim, a fórmula genérica para o cálculo do peso nebuloso de associações nebulosas envolvendo *n* itens, $n \in \mathrm{N}$ e $n > 2$, é dada pela Equação 6.2. O fator nebuloso *f* que nela aparece é obtido através da Equação 6.3, e o seu valor corresponde ao mínimo entre as similaridades envolvidas na associação nebulosa, de acordo com a teoria da intersecção entre conjuntos nebulosos apresentada na seção 3.2. Observe ainda que, quando são envolvidos apenas dois itens similares (*item1* e *item2*, por exemplo), a Equação 6.3 assume a forma $f = \min\left(sim(item_1, item_2)\right)$, ou seja, neste caso *f* corresponde ao valor da relação de similaridade entre *item1* e *item2*, de acordo com o que já havia sido definido na Equação 6.1.

$$Peso\ nebuloso = \left[\sum_{i=1}^{n} peso(item_i)\right]\left[\frac{1+f}{2}\right]$$

Equação 6.2 – Cálculo genérico do peso nebuloso.

$$f = \min\left(sim(item_1, item_2), \ldots, sim(item_{n-1}, item_n)\right)$$

Equação 6.3 – Cálculo do fator nebuloso *f*.

Esclarecida a forma pela qual são obtidos os pesos dos *itemsets* candidatos, sendo eles nebulosos ou não, pode-se ampliar o foco para a etapa de pesagem dos candidatos como um todo. A cada passo do algoritmo, a base de dados é percorrida, e cada uma de suas linhas é confrontada com o conjunto de *itemsets* candidatos, uma após a outra. Se o *itemset* candidato sendo analisado não é nebuloso, toda vez em que é encontrada uma ocorrência desse *itemset* em uma linha, seu peso é incrementado de 1, já que suas ocorrências são exatas (ou há ocorrência do *itemset*, ou não há). Porém, se o *itemset* candidato for nebuloso, suas ocorrências também o serão, e o valor pelo qual o peso será incrementado precisa ser calculado através da Equação 6.2. Para isso será necessário identificar os seus itens nebulosos, e de posse dos valores de similaridade entre eles, pode-se calcular o peso nebuloso do *itemset*, e atualizá-lo devidamente. Assim, percorridas todas as linhas, a pesagem dos *itemsets* candidatos estará completa.

Obtidos os pesos dos *itemsets* na etapa de pesagem, passa-se à etapa de avaliação dos candidatos.

6.3.6. Avaliação dos candidatos

Nessa etapa, similar à existente no algoritmo *Apriori*, avalia-se o suporte de cada *itemset* candidato. O suporte corresponde ao peso dividido pelo número de linhas (ou total de transações) presente na base de dados (Equação 6.4). O cálculo ocorre de forma análoga se o peso considerado for nebuloso e, nesse caso, o valor obtido para o suporte também o será. Assim, o *suporte nebuloso* também é obtido através da Equação 6.4. De forma genérica, calcula-se o suporte de cada *itemset* a partir de seu peso, e verifica-se se o suporte do *itemset* é maior do que o suporte mínimo (*minsup*). Em caso negativo, o *itemset* é considerado não freqüente, e é então descartado. Em caso positivo, guarda-se esse *itemset* no conjunto de *itemsets* freqüentes.

$$Suporte = \frac{peso(itemset)}{\text{número de linhas da base de dados}}$$

Equação 6.4 – Cálculo do suporte de um *itemset.*

Avaliados todos os *itemsets* candidatos, chega ao fim a parte iterativa do algoritmo. Têm-se então todos os *itemsets* freqüentes reunidos em um conjunto, a partir do qual pode-se iniciar a etapa de geração de regras (seção 6.3.7).

6.3.7. Geração de regras

As regras de associação possuem *antecedentes* (os itens à esquerda da seta) e *conseqüentes* (os itens à direita da seta), conforme a Figura 6.6.

$$Antecedente \rightarrow Conseqüente$$

Figura 6.6 – Antecedente e conseqüente da regra.

Para cada *itemset* do conjunto de *itemsets* freqüentes, o algoritmo gerará todas as possibilidades de antecedentes e conseqüentes para as regras, usando como critério de escolha o cálculo da confiança. Se a confiança, que é obtida de acordo com a Equação 6.5, for maior do que o parâmetro de confiança mínima (*minconf*), a regra será válida.

$$Confiança = \frac{Suporte(regra)}{Suporte(antecedente)}$$

Equação 6.5 – Cálculo da confiança da regra gerada.

Na etapa de avaliação dos candidatos (6.3.6), foi mostrado que o suporte de *itemsets* nebulosos é calculado da mesma forma que para os *itemsets* convencionais. Analogamente, sejam os suportes da regra e do antecedente nebulosos ou não, o cálculo da confiança é obtido do mesmo modo.

Ao final do algoritmo, todas as regras válidas são exibidas, mostrando antecedente, conseqüente, suporte e confiança de cada regra, no formato mostrado na Figura 6.7.

Antecedente → Conseqüente sup = <valor suporte> conf = <valor confiança>

Figura 6.7 – Formato das regras de associação exibidas.

No *SSDM*, ao contrário do que ocorre nos algoritmos convencionais, antecedentes e conseqüentes da regra podem conter itens nebulosos, e os valores de suporte e confiança exibidos refletem a influência do grau de similaridade entre os itens no cálculo dessas medidas.

6.4. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o conceito de mineração de dados semanticamente similares. Para realizar essa mineração, foi criado o algoritmo *SSDM*, e cada uma de suas etapas foram detalhadas ao longo da seção 6.3. Para validar o trabalho realizado, no próximo capítulo será apresentado um estudo de caso, envolvendo a aplicação do *SSDM* nas informações semânticas de um banco de dados multimídia.

7. ESTUDO DE CASO

7.1. INTRODUÇÃO

Os dados utilizados para a mineração foram extraídos de uma aplicação multimídia criada no contexto do projeto *AMMO* (*Authoring and Manipulation of Multimedia Objects*) [VIEIRA *et al.*, 2002], o qual provê um ambiente para suporte à autoria, armazenamento e manipulação de aplicações multimídia. Este ambiente contém um Servidor de Objetos Multimídia (*SOMm*) que propicia o suporte e a interação entre diferentes ferramentas. Uma delas, denominada *MAE (Multimedia Authoring Environment)* [SANTOS, 2003], é utilizada na criação de aplicações multimídia. As aplicações são armazenadas em um banco de dados multimídia, cujo esquema incorpora informações semânticas das mídias armazenadas. Essas informações são adicionadas como metadados pelo usuário criador da aplicação. Dessa forma, as características extraídas a partir das mídias já estão disponíveis para a mineração, constituindo uma tarefa que compreende os passos de seleção e transformação dos dados no processo de descoberta de conhecimento, comentados na seção 2.1. No algoritmo *SSDM*, essa tarefa é realizada nas duas primeiras etapas (seções 6.3.1 e 6.3.2).

Para melhor compreensão deste estudo, na próxima seção é apresentada a parte do esquema do banco de dados multimídia do *AMMO* destinada ao armazenamento das informações semânticas das mídias.

7.2. INFORMAÇÕES SEMÂNTICAS NO *AMMO*

As informações semânticas caracterizam as mídias armazenadas e, através delas, podem ser efetuadas recuperações, ou "consultas baseadas no conteúdo". Parte do esquema, que permite o armazenamento dessas informações e conseqüentemente a recuperação das mídias, está ilustrada na Figura 7.1. Nela, a classe *SemanticInformation* agrega as mídias armazenadas e os dados que compõem suas informações semânticas. A informação semântica associada a cada mídia possui termos ou composições de termos, e cada um deles é composto por um tema e seus possíveis qualificadores. Cada termo é representado pela classe *Subject*, enquanto seus temas e qualificadores são representados pela classe *Element*. Um exemplo de termo seria "carro azul", onde "carro" é o tema (geralmente um substantivo) e "azul" é o qualificador (geralmente um

adjetivo). Pode-se ter também um termo sem qualificador; nesse caso, o termo teria apenas o tema (“carro”, por exemplo). Caso haja composição de termos, essa composição é representada na classe *Composition*. Um exemplo de composição de termos seria “árvore frondosa com flor amarela”, onde os termos “árvore frondosa” e “flor amarela” são compostos através da palavra “com”. A relação de similaridade (ou proximidade) entre elementos de um mesmo domínio (representado na classe *Domain*) é representada na classe *Similarity*. Através dos valores da matriz de similaridade, trata-se a imprecisão nas informações semânticas armazenadas.

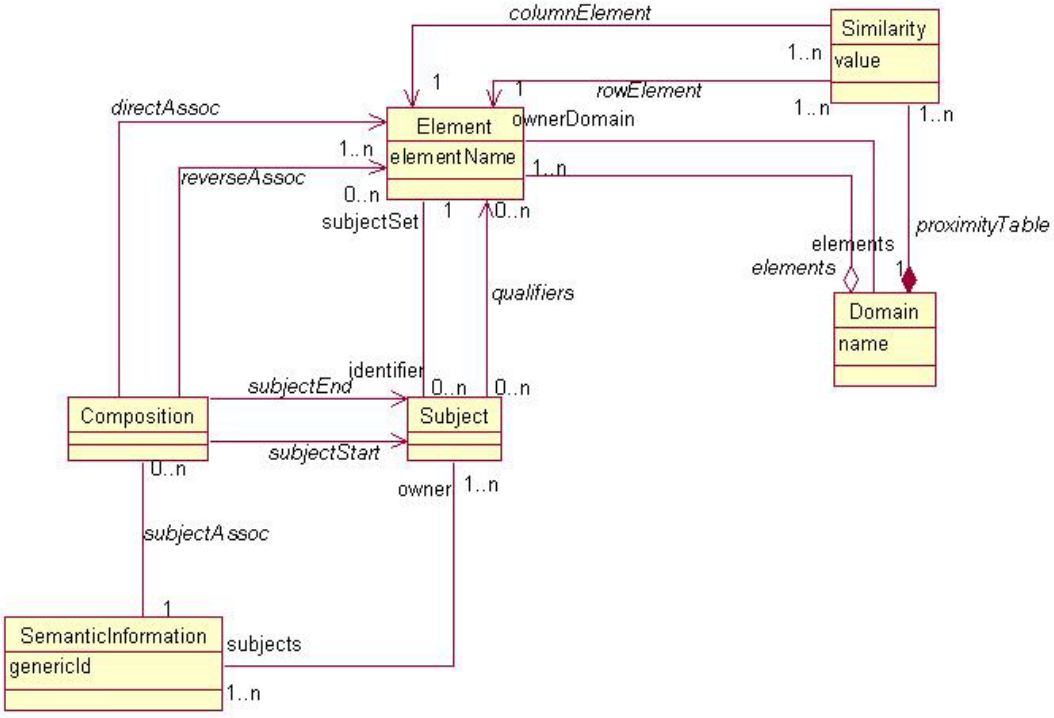

Figura 7.1 – Subesquema de informação semântica.

Assim, de acordo com o subesquema ilustrado acima, as aplicações multimídia têm suas mídias caracterizadas através de um conjunto de metadados, que representam as características que devem ser “extraídas” e utilizadas na tarefa de mineração de dados. Esse tipo de mineração, como foi descrito no capítulo 5, é denominado mineração de dados multimídia.

7.3. APLICAÇÃO DO ALGORITMO *SSDM*

A primeira fase do estudo de caso compreendeu a criação de uma aplicação multimídia através da ferramenta *MAE*. A aplicação utilizada corresponde a um sistema de agência de turismo, contendo imagens de pontos turísticos e paisagens, que podem ser consultadas pelos clientes da agência para, através delas, decidir qual o seu destino como turistas. No processo de criação, a informação semântica correspondente foi criada e, conseqüentemente, todos os metadados foram armazenados.

Em uma segunda fase, foi efetuada a transformação dos dados (uma das etapas de descoberta de conhecimento) para posterior aplicação do algoritmo. A transformação compreendeu a realização de consultas ao meta-esquema do banco para a extração das características das mídias a serem mineradas. Na transformação, as características foram automaticamente organizadas de forma que em uma mesma coluna, estejam informações de um mesmo domínio, atendendo a uma restrição do algoritmo *SSDM*. Note que, independentemente do tipo de mídia em questão, as características assumem a forma textual e estão prontas para o processo de mineração.

Considere a Tabela 7.1, a qual contém uma pequena amostra dos elementos que compõem as informações semânticas. Nela são mostradas instâncias do atributo *elementName*, da classe *Element*. Conforme foi apresentado na seção anterior, essa classe representa temas e qualificadores, que são associados a domínios. Assim, os temas e qualificadores aparecem nessa tabela junto a seus domínios correspondentes, representados na classe *Domain* através do atributo *name*.

Tabela 7.1 – Temas e qualificadores associados a seus domínios.

Element	*Domain*
elementName	*name*
lago	paisagem
montanha	paisagem
teatro	construção
novo	tempo
frio	temperatura
grande	tamanho
monumento	construção
quente	temperatura
praia	paisagem
velho	tempo

antigo	tempo
museu	construção
médio	tamanho
ponte	construção
gelado	temperatura
morro	paisagem
pequeno	tamanho
enorme	tamanho

Os elementos exibidos na Tabela 7.1 compõem termos que representam as informações semânticas das mídias. Eles são exibidos na Tabela 7.2. Para a aplicação do algoritmo *SSDM*, esses dados são selecionados e transformados, e o resultado desse processo pode ser visto na Tabela 7.3 e na Tabela 7.4.

Tabela 7.2 – Termos contendo temas e qualificadores.

	Termos		
t01	teatro	médio	antigo
t02	museu	pequeno	velho
t03	ponte	enorme	novo
t04	monumento	grande	novo
t05	lago	pequeno	frio
t06	montanha	enorme	gelado
t07	praia	grande	quente
t08	morro	médio	frio
t09	teatro	grande	novo
t10	museu	enorme	antigo
t11	ponte	pequeno	velho
t12	monumento	grande	antigo
t13	lago	médio	quente
t14	montanha	grande	frio
t15	praia	grande	quente
t16	teatro	médio	antigo
t17	museu	pequeno	velho
t18	ponte	enorme	novo
t19	monumento	grande	novo
t20	lago	pequeno	frio
t21	montanha	enorme	gelado
t22	praia	grande	quente
t23	morro	médio	frio
t24	teatro	grande	novo
t25	museu	enorme	novo
t26	ponte	pequeno	velho

t27	teatro	médio	velho
t28	museu	pequeno	velho
t29	ponte	grande	novo
t30	praia	médio	quente

Tabela 7.3 – Termos que envolvem temas do domínio construção.

	construção	*tamanho*	*tempo*
t01	teatro	médio	antigo
t02	museu	pequeno	velho
t03	ponte	enorme	novo
t04	monumento	grande	novo
t09	teatro	grande	novo
t10	museu	enorme	antigo
t11	ponte	pequeno	velho
t12	monumento	grande	antigo
t16	teatro	médio	antigo
t17	museu	pequeno	velho
t18	ponte	enorme	novo
t19	monumento	grande	novo
t24	teatro	grande	novo
t25	museu	enorme	novo
t26	ponte	pequeno	velho
t27	teatro	médio	velho
t28	museu	pequeno	velho
t29	ponte	grande	novo

Tabela 7.4 – Termos que envolvem temas do domínio paisagem.

	paisagem	*tamanho*	*temperatura*
t05	lago	pequeno	frio
t06	montanha	enorme	gelado
t07	praia	grande	quente
t08	morro	médio	frio
t13	lago	médio	quente
t14	montanha	grande	frio
t15	praia	grande	quente
t20	lago	pequeno	frio
t21	montanha	enorme	gelado
t22	praia	grande	quente
t23	morro	médio	frio
t30	praia	médio	quente

Além disso, para cada um dos 5 domínios identificados é obtida a matriz de similaridade correspondente. Essas matrizes são exibidas na Figura 7.2.

Domínio: c*onstrução*

	teatro	museu	monumento	ponte
Teatro	1	0.9	0.4	0.2
Museu	0.9	1	0.7	0.2
Monumento	0.4	0.7	1	0.3
Ponte	0.2	0.2	0.3	1

Domínio: *paisagem*

	lago	praia	montanha	morro
Lago	1	0.8	0.6	0.3
Praia	0.8	1	0.2	0.25
Montanha	0.6	0.2	1	0.85
Morro	0.3	0.25	0.85	1

Domínio: *tamanho*

	pequeno	médio	grande	enorme
Pequeno	1	0.6	0.35	0.1
Médio	0.6	1	0.25	0.2
Grande	0.35	0.25	1	0.85
Enorme	0.1	0.2	0.85	1

Domínio: *tempo*

	antigo	velho	novo
Antigo	1	0.95	0.4
Velho	0.95	1	0.1
Novo	0.4	0.1	1

Domínio: *temperatura*

	quente	gelado	frio
Quente	1	0.1	0.05
Frio	0.1	0.75	1
Gelado	0.05	1	0.75

Figura 7.2 – Matrizes de similaridade dos domínios identificados.

Considerando os valores de 25% para *minsup*, 50 % para *minconf*, e 80% para *minsim,* o algoritmo *SSDM* é aplicado, e assim são obtidas as regras de associação exibidas na Figura 7.3.

```
 1  grande->novo sup=0.2777778 conf=0.8333334
 2  novo->grande sup=0.2777778 conf=0.625
 3  grande~enorme->novo sup=0.41111112 conf=0.8
 4  novo->grande~enorme sup=0.41111112 conf=0.925
 5  pequeno->velho~antigo sup=0.27083334 conf=0.97499996
 6  velho~antigo->pequeno sup=0.27083334 conf=0.5
 7  pequeno->velho sup=0.2777778 conf=1.0
 8  velho->pequeno sup=0.2777778 conf=0.8333334
 9  grande->quente sup=0.25 conf=0.75
10  quente->grande sup=0.25 conf=0.6
11  praia~lago->quente sup=0.375 conf=0.71428573
12  quente->praia~lago sup=0.375 conf=0.90000004
13  praia->quente sup=0.33333334 conf=1.0
14  quente->praia sup=0.33333334 conf=0.8000001
15  praia,grande->quente sup=0.25 conf=1.0
16  quente,grande->praia sup=0.25 conf=1.0
17  quente,praia->grande sup=0.25 conf=0.75
18  grande->quente,praia sup=0.25 conf=0.75
19  praia->quente,grande sup=0.25 conf=0.75
20  quente->praia,grande sup=0.25 conf=0.6
21  grande->praia sup=0.25 conf=0.75
22  praia->grande sup=0.25 conf=0.75
```

Figura 7.3 – Regras de associação obtidas pelo *SSDM*.

Note que as regras 3, 4, 5, 6, 11 e 12 não seriam obtidas através da aplicação de um algoritmo convencional de mineração de regras de associação. A regra 4, por exemplo, mostra que as ocorrências de "novo" tendem a coincidir com as ocorrências dos itens similares "grande" e "enorme", com suporte (nebuloso) de 41,11% e confiança de 92,5%. Uma possível interpretação dessa regra, no contexto da aplicação da agência de turismo, é "entre as imagens de pontos turísticos, 41,11% mostram que o que é novo é grande ou enorme, e em 92,5% das imagens em que aparece uma atração turística grande ou enorme, ela é nova". O conhecimento representado por regras de associação que consideram as similaridades semânticas entre os dados, como no caso de "grande" e "enorme", passa a ser gerado quando o algoritmo *SSDM* é aplicado.

7.4. CONSIDERAÇÕES FINAIS

Esse estudo de caso demonstrou que através do algoritmo *SSDM* é possível obter regras de associação com informações contemplando a similaridade semântica dos dados, ignorada por algoritmos convencionais. Assim, a descoberta de conhecimento é favorecida, já que a análise do significado dos dados na mineração propicia a obtenção de regras de associação mais compreensíveis e ricas em informações.

8. CONCLUSÕES

8.1. RESULTADOS ALCANÇADOS

Com a criação e aplicação do algoritmo *SSDM,* tornou-se possível descobrir regras de associação que refletem a similaridade semântica que pode haver entre os dados. Com isso, o conhecimento fornecido por essas regras passa a ser mais rico, pois são reveladas associações entre os dados que algoritmos convencionais não são capazes de descobrir. O uso de conceitos de lógica nebulosa no *SSDM* contribuiu para que a representação e manipulação das informações estejam mais próximas da linguagem humana, tornando-as mais inteligíveis. Como o processo de descoberta de conhecimento (*KDD)* busca padrões potencialmente úteis e compreensíveis nos dados, quanto melhor for a compreensão do conhecimento obtido, maior será a sua utilidade. A mineração de dados semanticamente similares caminha nesse sentido.

A mineração de dados multimídia (capítulo 5) ainda tem que evoluir para que o processo todo seja totalmente automático. Estudos referentes ao estado da arte em mineração multimídia mostram que nela as características das mídias são extraídas e que, então, o processo de mineração se dá sobre os dados extraídos. O algoritmo *SSDM*, apresentado na seção 6.3, segue essa linha de implementação, e ao agregar maior semântica no processamento dos dados, contribui para a evolução do estado da arte. Essa contribuição fica particularmente evidente ao se analisar os tipos de metadados minerados nas abordagens estudadas. Enquanto nesses trabalhos são minerados metadados obtidos a partir de características como cores, padrões e formatos, o objeto da mineração de dados semanticamente similares são as informações semânticas.

8.2. CONTRIBUIÇÕES

Estão entre as contribuições deste trabalho:

- Um novo algoritmo (*SSDM*) que realiza mineração de dados semanticamente similares, além de ser genérico, uma vez que pode ser aplicado na mineração das informações semânticas de qualquer tipo de dados.
- A criação de um novo parâmetro (*minsim*) para a mineração de dados, que indica a similaridade mínima entre os itens desejada.

- A definição de associação nebulosa, no contexto do *SSDM*.
- A definição e a estratégia para identificação de ciclo de similaridades.
- As definições de item nebuloso e *itemset* nebuloso, no contexto do *SSDM*.
- A definição de ocorrência nebulosa, e a criação de fórmulas para o cálculo do peso nebuloso.
- A definição de suporte nebuloso, que permite manter inalterada a forma de cálculo da confiança.
- A possibilidade de existir itens nebulosos nas regras de associação exibidas, representando a associação entre itens suficientemente similares, e a possibilidade de os valores de suporte e confiança que aparecem nas regras serem calculados de forma nebulosa.

8.3. TRABALHOS FUTUROS

8.3.1. Avaliar o desempenho

Não foram realizados testes de desempenho com o algoritmo *SSDM*. Essa tarefa deverá ser realizada comparando-o com outros algoritmos existentes na literatura. Pretende-se efetuar dois tipos de comparação:

1) Comparar o *SSDM* com algoritmos convencionais. Por exemplo, comparar *Apriori* (seção 2.3.1) e *SSDM*, para avaliar a diferença de desempenho entre um algoritmo clássico que minera regras de associação, e um algoritmo que além disso também minera similaridades semânticas entre os dados.

2) Incorporar as novas funcionalidades presentes no *SSDM* em outros algoritmos existentes, e então compará-los com o *SSDM*. Isso poderia ser feito, por exemplo, em relação ao algoritmo *FP-growth* (seção 2.3.3). Note-se que, através da realização dessa tarefa, novos algoritmos serão criados.

8.3.2. Refinar o processo de expressão das regras

No algoritmo atual, as regras exibidas podem compreender itens nebulosos, e nesse caso os valores de suporte e confiança nelas exibidos refletirão a presença deles na regra. Porém, a forma como as regras são expressas, apresentada na Figura 6.7, segue a dos algoritmos convencionais. Assim, pode ser realizado um estudo para definir uma forma mais refinada de expressar os conceitos envolvidos na mineração de dados semanticamente similares. Afinal, as regras constituem a saída do algoritmo, e quanto maior for a qualidade da representação das informações, melhor será a interpretação das mesmas e conseqüentemente a capacidade de se descobrir conhecimento a partir das regras.

8.3.3. Realizar baterias de testes

Foram realizados testes sobre uma massa de dados capaz de validar o uso do algoritmo. Entretanto, deseja-se avaliar se o uso da similaridade pode trazer algum tipo de distorção no resultado, considerando-se que os valores de similaridade representam a semântica introduzida por um usuário e que por esse motivo não estão totalmente livres de problemas de interpretação. Para isso, é necessária a realização de baterias de testes envolvendo variedade e quantidade maiores de dados.

8.3.4. Substituição da estrutura *Hash-tree*

O *SSDM* utiliza uma estrutura de *Hash-tree*, adaptada daquela usada no algoritmo *Apriori*, para armazenar e recuperar os *itemsets* analisados durante a mineração. Além disso, o peso de um *itemset* também é atualizado na *Hash-tree*, à medida que novas ocorrências dos itens que o compõem são encontradas na base de dados. Na mineração de dados semanticamente similares, realizada pelo *SSDM*, os *itemsets* podem ser nebulosos, e por isso a construção de uma estrutura de dados que os manipule mais adequadamente pode ser avaliada. Uma possibilidade é a utilização de uma estrutura métrica, na qual os dados estejam armazenados em um nó conforme a similaridade ou dissimilaridade apresentada entre eles.

8.3.5. Permitir outras formas de definição dos domínios

Conforme foi comentado na seção 6.3.1, a definição dos domínios no *SSDM* é importante para que seja possível relacionar os itens quanto à sua similaridade somente quando isso for conveniente, ou seja, quando pertencerem a um mesmo domínio. Para possibilitar a aplicação do *SSDM* nas informações semânticas de qualquer tipo de dados, adotou-se a princípio como solução definir os domínios aos quais os itens pertencem de acordo com a coluna em que eles aparecem na base de dados. Existem, porém, aplicações onde esse não é um bom critério para a definição dos domínios, porque nelas seria interessante uma análise da similaridade semântica entre itens que fazem parte de uma mesma linha (portanto, de colunas diferentes) de uma base de dados. Assim, seria interessante permitir outras formas de definição de domínios, para que os itens sejam neles agrupados de acordo com a análise semântica desejada.

9. REFERÊNCIAS BIBLIOGRÁFICAS

AGRAWAL, R. e SRIKANT, R. **Fast Algorithms for Mining Association Rules**. In: 20th Conference on Very Large Data Bases (VLDB'94), 1994, Santiago, Chile. **Anais.** Santiago, Chile, 1994. p. 487-499.

ANTHONIE, M. L., ZAÏANE, O. R. e COMAN, A. **Application of Data Mining Techniques for Medical Image Classification**. In: 2nd International Workshop on Multimedia Data Mining (MDM/SIGKDD'2001), 2001, San Francisco, USA. **Anais.** San Francisco, USA, 2001. p. 94-101.

AU, W.-H. e CHAN, K. C. C. **FARM: A Data Mining System for Discovering Fuzzy Association Rules**. IEEE International Conference on Fuzzy Systems, 1999.

BIANCHI-BERTHOUZE, N. e HAYASHI, T. **Subjective interpretation of complex data: Requirements for supporting kansei mining process**. In: 3rd International Workshop on Multimedia Data Mining (MDM/SIGKDD'2002), 2002, Edmonton, Canada. **Anais.** Edmonton, Canada, 2002. p. 93-99.

CHAN, K. C. C. e AU, W.-H. **An Effective Algorithm for Discovering Fuzzy Rules in Relational Databases**. In: IEEE International Conference on Fuzzy Systems, 1998, Anchorage, Alaska. **Anais.** Anchorage, Alaska, 1998.

CHEN, G. e WEI, Q. **Fuzzy association rules and the extended mining algorithms**. Fuzzy Sets and Systems, v. 147, n. 1-4, p. 201-228, 2002.

CHEN, G., WEI, Q. e KERRE, E. E. Fuzzy Data Mining: Discovery of Fuzzy Generalized Association Rules. In: G. Bordogna e G. Pasi. **Recent Issues on Fuzzy Databases**. Physica-Verlag, 2000, p. 45-66.

DATCU, M. e SEIDEL, K. **An Innovative Concept for Image Information Mining**. In: 3rd International Workshop on Multimedia Data Mining (MDM/SIGKDD'2002), 2002, Edmonton, Canada. **Anais.** Edmonton, Canada, 2002. p. 11-18.

DEOGUN, J. S.*, et al.* Data Mining: Trends in Research and Development. In: T. Y. Lin e N. Cercone. **Rough Sets and Data Mining - Analysis of Imprecise Data**. Kluwer Academic Publishers, 1997, p. 9-24.

DJERABA, C. **Relationship extraction from large image databases**. In: 2nd International Workshop on Multimedia Data Mining (MDM/SIGKDD'2001), 2001, San Francisco, USA. **Anais.** San Francisco, USA, 2001. p. 44-49.

DUBOIS, D. e PRADE, H. Fuzzy Relations. In: **Fuzzy Sets and Systems: Theory and Applications**. New York: "Mathematics in Science and Engineering" series, Academic Press, 1980, p. 68-93.

DVORAK, J. e SEDA, M. **Comparison of fuzzy similarity values**. In: 3rd International Conference Aplimat, 2004, Bratislava, Slovak. **Anais.** Bratislava, Slovak, 2004. p. 387-392.

ELMASRI, R. e NAVATHE, S. B. Data Mining Concepts. In: **Fundamentals of Database Systems**. Addison-Wesley, 2004, p. 867-897.

FAYYAD, U., PIATETSKY-SHAPIRO, G. e SMYTH, P. **From Data Mining to Knowledge Discovery in Databases**. AI Magazine (AAAI), 1997.

HAN, J. e KAMBER, M. **Data Mining - Concepts and Techniques**. 1ª edição. Nova York: Morgan Kaufmann, 2001.

HAN, J., PEI, J. e YIN, Y. **Mining frequent patterns without candidate generation**. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data, 2000, Dallas, Texas, USA. **Anais.** Dallas, Texas, USA, 2000.

HIPP, J., GÜNTZER, U. e NAKHAEIZADEH, G. **Algorithms for Association Rule Mining - A General Survey and Comparison**. SIGKDD Explorations, v. 2, n. 1, p. 58-64, 2000.

HONG, T.-P., KUO, C.-S. e CHI, S.-C. **Mining association rules from quantitative data**. Intelligent Data Analysis, v. 3, n. 5, p. 363-376, 1999.

KLIR, G. J. e YUAN, B. **Fuzzy Sets and Fuzzy Logic - Theory and Applications**. Prentice Hall P T R, 1995.

KUOK, C. M., FU, A. e WONG, M. H. **Mining Fuzzy Association Rules in Databases**. ACM SIGMOD Record, v. 27, n. 1, p. 41-46, 1998.

LEE, J.-H. e LEE-KWANG, H. **An Extension of Association Rules Using Fuzzy Sets**. In: 7th International Fuzzy Systems Association World Congress, 1997, Prague, Czech. **Anais.** Prague, Czech, 1997. p. 399-402.

MANNILA, H. **Methods and problems in data mining**. In: International Conference on Database Theory, 1997, Delphi, Greece. **Anais.** Delphi, Greece, 1997.

MARCHIORI, E. **Data Mining**. Encyclopedia of Life Support Systems (EOLSS), 2000.

MITRA, S. e ACHARYA, T. **Data Mining: multimedia, soft computing, and bioinformatics**. Wiley-Interscience, 2003.

ORDONEZ, C. e OMIECINSKI, E. **Discovering Association Rules based on Image Content**. In: IEEE Advances in Digital Libraries Conference (ADL'99), 1999, Baltimore, Maryland (USA). **Anais.** Baltimore, Maryland (USA), 1999.

ORLANDO, S., PALMERINI, P. e PEREGO, R. **Enhancing the Apriori Algorithm for Frequent Set Counting**. Lecture Notes in Computer Science - Springer-Verlag, v. 2114, p. 71+, 2001.

PEDRYCZ, W. **Fuzzy set technology in knowledge discovery**. Fuzzy Sets and Systems, v. 98, n. 3, p. 279-290, 1998.

PEDRYCZ, W. e GOMIDE, F. Fuzzy Relations and Their Calculus. In: **An Introduction to Fuzzy Sets: Analysis and Design**. "A Bradford book." - The MIT Press, 1998, p. 85-126.

SANTOS, F. G. **Ferramenta MAE: Autoria Multimídia e Integração com Banco de Dados**. 2003. Dissertação (Mestrado) – Departamento de Computação (DC), UFSCar, São Carlos, 2003.

SAVARESE, A., OMIECINSKI, E. e NAVATHE, S. **An Efficient Algorithm for Mining Association Rules in Large Databases**. In: 21st Conference on Very Large Databases (VLDB'95), 1995, **Anais.** 1995.

SRIKANT, R. e AGRAWAL, R. **Mining Generalized Association Rules**. In: 21st VLDB Conference, 1995, Zurich, Switzerland. **Anais.** Zurich, Switzerland, 1995. p. 407-419.

_____ **Mining Quantitative Association Rules in Large Relational Tables**. In: ACM SIGMOD, 1996, Montreal, Canada. **Anais.** Montreal, Canada, 1996. p. 1-12.

VIEIRA, M. T. P.*, et al.* Content-Based Fuzzy Search in a Multimedia Web Database. In: **Intelligent Exploration of the Web**. "Studies in Fuzziness and Soft Computing" series, Springer-Verlag, 2002.

WOJCIECHOWSKI, M. e ZAKRZEWICZ, M. **On Efficiency of Dataset Filtering Implementations in Constraint-Based Discovery of Frequent Itemsets**. In: 2002 JCKBSE Conference, 2002, Maribor, Slovenia. **Anais.** Maribor, Slovenia, 2002.

ZADEH, L. A. Fuzzy Sets. In: R. R. Yager, S. Ovchinnikov*, et al*. **Fuzzy Sets and Applications: Select Papers by L. A. Zadeh**. Wiley-Interscience, 1987a, p. 29-44.

_____. Similarity Relations and Fuzzy Orderings. In: R. R. Yager, S. Ovchinnikov*, et al*. **Fuzzy Sets and Applications: Select Papers by L. A. Zadeh**. Wiley-Interscience, 1987b, p. 81-104.

ZAKI, M.*, et al.* **New algorithms for fast discovery of association rules**. In: Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, 1997, Newport Beach, CA, USA. **Anais.** Newport Beach, CA, USA, 1997.

ZHANG, J., HSU, W. e LEE, M. L. **Image Mining: Issues, Frameworks and Techniques**. In: 2nd International Workshop on Multimedia Data Mining (MDM/SIGKDD'2001), 2001, San Francisco, USA. **Anais.** San Francisco, USA, 2001. p. 13-20.