Relatório de Atividades

BCC390 - Monografia I

MINERAÇÃO DE DADOS PARA PADRÕES DE SEQUENCIA

Cecília Henriques Devêza

OURO PRETO, 27 DE NOVEMBRO DE 2010

Introdução

- O que são Padrões de Sequência
- O projeto
- Algoritmo GSP
- Atividades Desenvolvidas
- Trabalhos Futuros

Padrões de Sequência

- Coleção ordenada - por data - de *itemsets*.

- Objetivo: Responder à questão:

"Que sequencia de produtos são comprados por um mesmo cliente em momentos consecutivos?"

O projeto

- Criar um software capaz de realizar uma preparação de uma base de dados de forma automática, minerar os padrões de sequência da base, e repassar como saída os produtos traduzidos gerados pelo algoritmo GSP.

Algoritmo GSP

BASE DE DADOS INICIAL

Trans.	IdCliente	Produto	Loja	Data
1	223954afbfc851101040	Mitiflex	C	2010-04-05
2	af9a49394f5486385aeb	Pequenas Fontes...	B	2010-04-05
3	f7c191a0cd6a75c5cbe2	Surdez Processos...	C	2010-04-05
4	2aw23ew3983y4thi4438	CDI Digital DAFRA...	D	2010-04-05
5	t52kdi7486db5g1mnk6t8	Agulha Auricular C...	A	2010-04-06
...	64dsb8dbdf8bfdb6wd65	Zumbido Avaliação...	C	2010-04-07
3.525.926	d2f1dsg23sd1gs35ytj5ui	O Pastor Alemão	B	2010-04-07

Algoritmo GSP

- Base de Dados esperada:

IdCliente	Sequencia de Itemsets
1	<{2}, {5}, {87}, {323}>
2	<{79}, {141}, {456}>
3	<{324}, {68}>
4	<{23}, {1}, {908}, {8}, {3}>
5	<{6}, {79}, {34}, {34543}>
...	<{879}, {353}, {9}>
958.652.471	<{4546}>

MAIS ANTIGO MAIS RECENTE

Algoritmo GSP

- Geração de Pré-candidatos
- Poda de pré-candidatos
- Cálculo de Suporte
- Definição de itemsets frequentes
  - Definição de sequências ligáveis

Atividades Desenvolvidas

- Inserção dos dados em banco remoto
- Escolha da loja-teste
- Limpeza da base escolhida
  - Atualização de nomes que sofreram alterações no processo de inserção
  - Deleção de clientes e produtos que aparecem apenas uma vez na base

Trabalhos Futuros

- Teste da base gerada
- Implementação do software
  - Preparação da base automática
  - Tradução dos resultados gerados
- Análise dos resultados
- Escrita do Trabalho Final

Perguntas?

Anexos

- Detalhamento do GSP realizado sobre o artigo “Mining Sequencial Patterns: Generalizations and Performance Improvements” de Remakrishnan Srikant and Rakesh Agrawal

Definição: AprioriAll x GSP

Algoritmo de extração de padrões de sequência considerado mais eficiente que o AprioriAll, porque poda mais candidatos na fase de poda, devido à uma otimização na construção de seus pré-candidatos.

Na fase de geração de candidatos:

- No AprioriAll, em cada iteração k, os conjuntos L_k e C_k são constituídos de sequências de k itemsets.
- No GSP, em cada iteração k os conjuntos L_k e C_k são constituídos de sequencias de k itens.

Analogia – Cesta de Compras

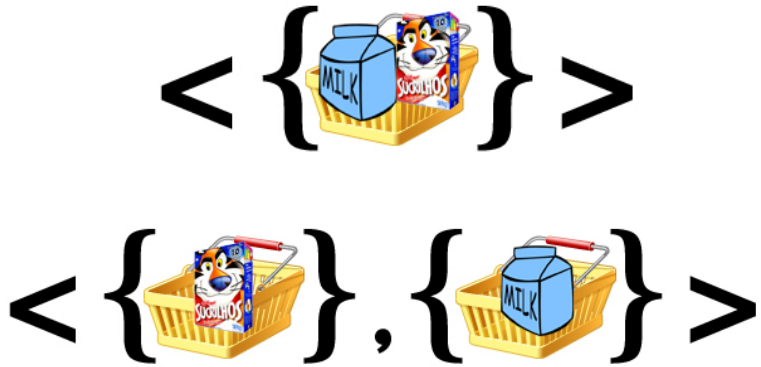
Iteração 1

ApriorиAll

GSP

Iteração 2

Ou seja: <{1}, {2}>



Ou seja: <{1, 2}>, <{1},{2}>

Transformação da Base

idUsuario	Produtos	Data
1	{2, 1, 4, 6}	06/06/2010
2	{2, 4, 5}	07/06/2010
1	{3, 7, 8}	07/06/2010
1	{5}	08/07/2010
3	{2, 4}	13/07/2010
2	{1, 6, 7}	17/07/2010
2	{3}	23/07/2010
3	{1, 5}	26/07/2010
4	{1, 3}	01/08/2010
1	{9}	25/08/2010
4	{2, 4, 5}	13/09/2010

TID	Produtos
1	<{2, 1, 4, 6}, {3, 7, 8}, {5}, {9}>
2	<{2, 4, 5}, {1, 6, 7}, {3}>
3	<{2, 4}, {1, 5}>
4	<{1, 3}, {2, 4, 5}>

Itemsets frequentes – Iteração 1

GERAÇÃO DE PRÉ-CANDIDATOS E CÁLCULO DE SUPORTE

TID	Produtos
1	<{2, 1, 4, 6}, {3, 7, 8}, {5}, {9}>
2	<{2, 4, 5}, {1, 6, 7}, {3}>
3	<{2, 4}, {1, 5}>
4	<{1, 3}, {2, 4, 5}>

Prods.	Suporte
<{1}>	4
<{2}>	4
<{3}>	3
<{4}>	4
<{5}>	4
<{6}>	2
<{7}>	2
<{8}>	1
<{9}>	1

Supondo suporte mínimo igual a 50%, todos os itens são considerados frequentes nesta iteração exceto o {8} e {9}.

Itemsets frequentes – Iteração 2

GERAÇÃO DE PRÉ-CANDIDATOS

(k=1)
<{1}>
<{2}>
<{3}>
<{4}>
<{5}>
<{6}>
<{7}>

→

L_1	X	L_1
<{1}>		<{1}>
<{2}>		<{2}>
<{3}>		<{3}>
<{4}>		<{4}>
<{5}>		<{5}>
<{6}>		<{6}>
<{7}>		<{7}>

→

TID	Produtos
1	<{1, 1}>, <{1}, {1}>
2	<{1, 2}>, <{1}, {2}>
3	<{1, 3}>, <{1}, {3}>
4	<{1, 4}>, <{1}, {4}>
5	<{1, 5}>, <{1}, {5}>
6	<{1, 6}>, <{1}, {6}>
7	<{1, 7}>, <{1}, {7}>
8	<{2, 1}>, <{2}, {1}>
9	<{2, 2}>, <{2}, {2}>
10	...

Na segunda iteração, são criados todos os conjuntos possíveis de 2 itens. Os conjuntos que possuem o mesmo elemento, na mesma transação, sempre possuem frequência 0. (Ex.: <{1, 1}>).

Itemsets frequentes – Iteração 2

CÁLCULO DE SUPORTE

Tendo todos os candidatos possíveis de 2 itens, é necessário verificar quais atendem ao suporte mínimo especificado. Esta verificação será mostrada posteriormente.

Itemsets frequentes – Iteração 3

A partir desta iteração, é preciso ficar atento para a seguinte regra:

Os candidatos só podem ser gerados a partir de duas **sequências ligáveis.**

Duas sequências s = < s1, s2, s3, s4, ..., sn > e t = < t1, t2, t3, t4, ..., tm > são ditas ligáveis se, retirando-se o primeiro item de s1 e o último item de tm, as sequências resultantes são iguais. Neste caso, s e t podem ser ligadas e produzir a sequência V, onde:

- Se tm não é unitário: v = <s1, s2, s3, ..., sn U t'>, onde t' é o último item de tm.

- Se tm é unitário: v = <s1, s2, s3, ..., sn, tm>

Itemsets frequentes – Iteração 3

GERAÇÃO DE PRÉ-CANDIDATOS

Suponha que as 2-sequências mostradas abaixo atingiram o suporte mínimo.

(k=2)
<{1}, {2}>
<{1}, {3}>
<{1}, {4}>
<{1, 5}>
<{2}, {6}>
<{2, 6}>
<{6}, {7}>
<{4},{6}>
<{3, 7}>
<{5, 6}>
<{4, 7}>

→

TID	Produtos
1	<{1}, {2}, {6}>
2	<{1}, {2,6}>
3	<{1}, {3, 7}>
4	<{1}, {4}, {6}>
5	<{1}, {4, 7}>
6	<{1, 5, 6}>
7	<{2}, {6}, {7}>
8	<{2, 6}, {7}>
9	<{4}, {6}, {7}>
10	<{5, 6}, {7}>

Itemsets frequentes – Iteração 3

FASE DE PODA DOS PRÉ-CANDIDATOS

Para uma sequencia S ser frequente, então toda subsequência de *S* deve se
frequente.

(k=2)
<{1}, {2}>
<{1}, {3}>
<{1}, {4}>
<{1, 5}>
<{2}, {6}>
<{1, 6}>
<{6}, {7}>
<{4},{6}>
<{3, 7}>
<{5, 6}>
<{4},{7}>

TID	Produtos
1	<{1}, {2}, {6}>
2	<{1}, {2,6}>
3	<{1}, {3, 7}>
4	<{1}, {4}, {6}>
5	<{1}, {4, 7}>
6	<{1, 5, 6}>
7	<{2}, {6}, {7}>
8	<{2, 6}, {7}>
9	<{4}, {6}, {7}>
10	<{5, 6}, {7}>

(K=3)
<{1, 5, 6}>
<{4}, {6}, {7}>

A subsequência <{1}, {6}> não pertence à $L_{k=2}$

Itemsets frequentes – Iteração 3

CÁLCULO DE SUPORTE

Novamente, é feito o cálculo do suporte para os candidatos que permaneceram após a poda. Este cálculo é mostrado a seguir.

Cálculo do Suporte

GERAÇÃO DA ÁRVORE-HASH

Suponha as seguintes 2-sequências candidatas:

(k=2)
<{1, 3}>
<{1}, {3}>
<{2}, {3}>
<{3}, {3}>
<{2, 3}>
<{1}, {4}>

Para construir a árvore hash, é preciso definir duas variáveis M e N, sendo:

- M : Número de sequências que cabem em um mesmo nó.
- N : Número de nós-filhos que um nó pode possuir.

Neste exemplo, usaremos M=3, N=2.

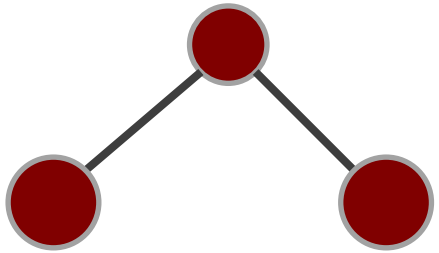
Cálculo do Suporte

Primeiramente, definimos as funções de hash para cada item que aparece na base:
h(1) = 1; h(2) = 2; h(3) = 1; h(4) = 2; h(5) = 1. (Variam de acordo com N).

Inserindo as 3 primeiras sequências na árvore:

<{1, 3}>
<{1}, {3}>
<{2}, {3}>

Na hora de inserir a quarta sequência <{3}, {3}>, o nó-raiz está cheio!

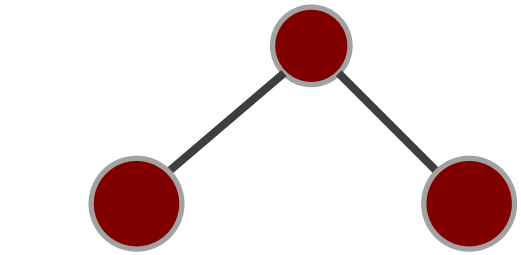


<{1, 3}>
<{1}, {3}>
<{3}, {3}>

<{2}, {3}>

Cálculo do Suporte

Inserindo a sequência <{2, 3}> na árvore:



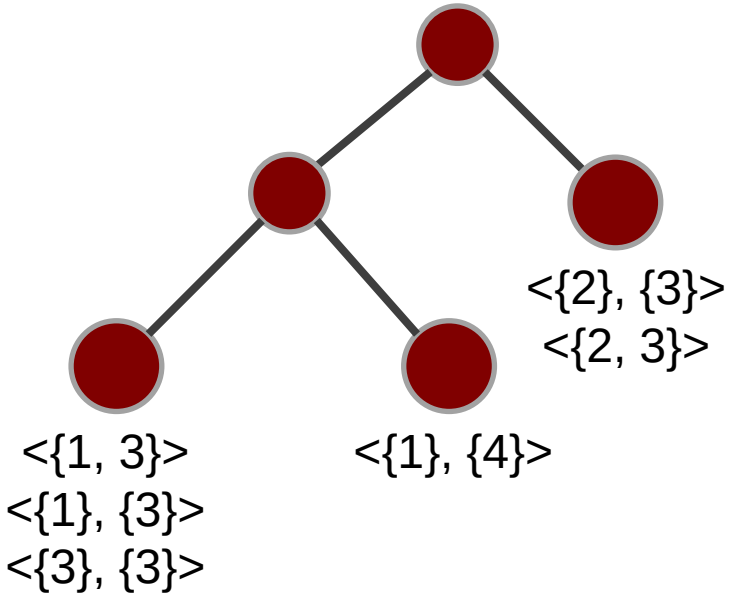
<{1, 3}>
<{1}, {3}>
<{3}, {3}>

<{2}, {3}>
<{2, 3}>

Na hora de inserir <{1}, {4}> seu nó de destino encontra-se cheio!

Cálculo do Suporte

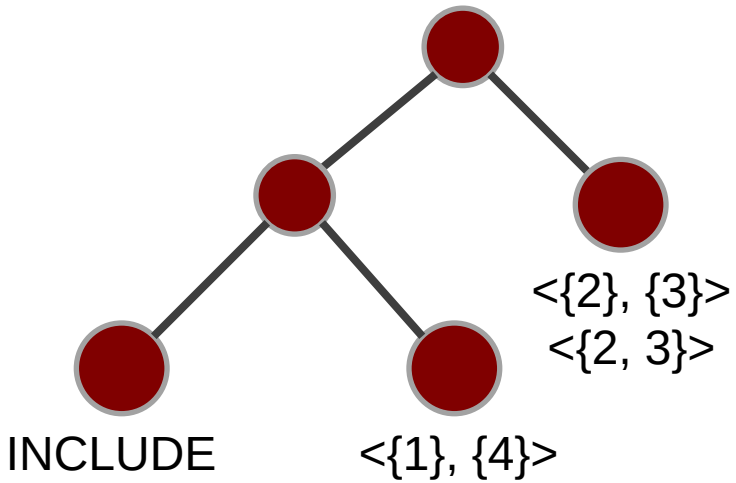
Inserindo a sequência <{1}, {4}> na árvore:



Cálculo do Suporte

CONTANDO AS FREQUÊNCIAS A PARTIR DA ÁRVORE HASH

Supondo a seguinte sequência de cliente: d = <{1, 5}, {1}, {3}>

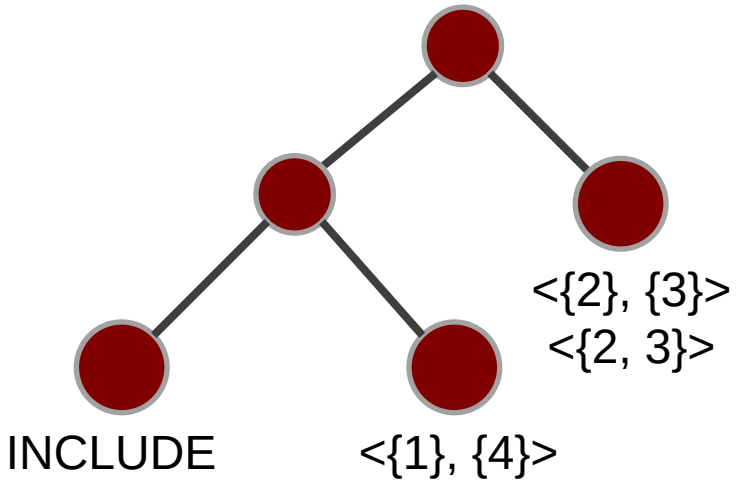


h(1) = 1; h(2) = 2; h(3) = 1; h(4) = 2; h(5) = 1

Cálculo do Suporte

CONTANDO AS FREQUÊNCIAS A PARTIR DA ÁRVORE HASH

Supondo a seguinte sequência de cliente: d = <{1, 5}, {1}, {3}>

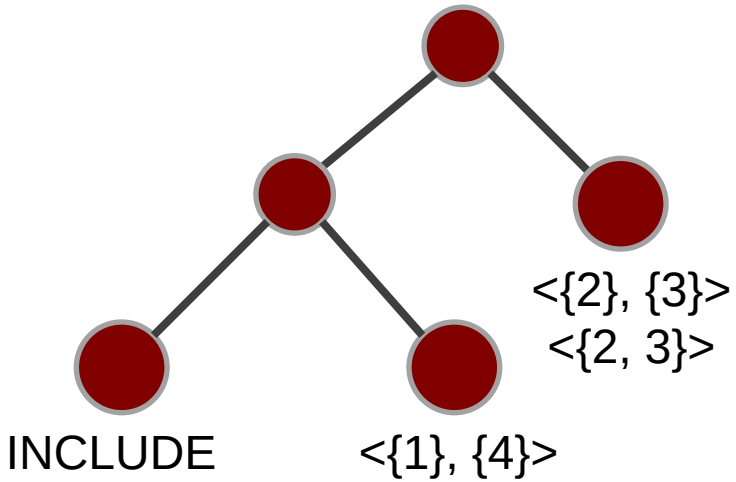


h(1) = 1; h(2) = 2; h(3) = 1; h(4) = 2; h(5) = 1

Cálculo do Suporte

CONTANDO AS FREQUÊNCIAS A PARTIR DA ÁRVORE HASH

Supondo a seguinte sequência de cliente: d = <{1, 5}, {1}, {3}>

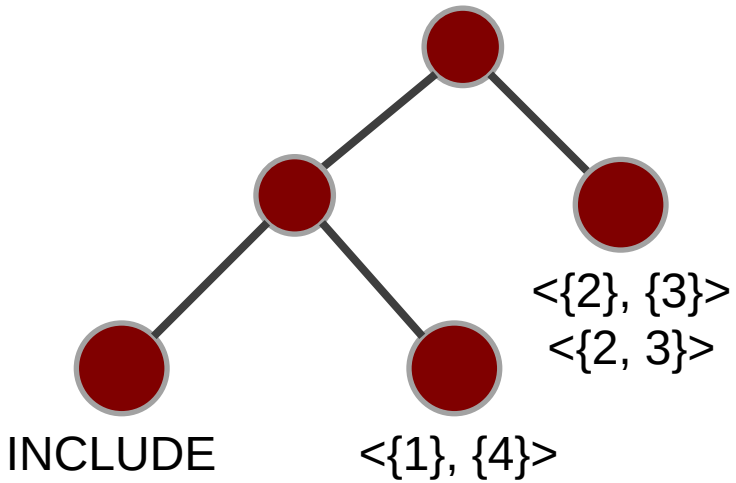


h(1) = 1; h(2) = 2; h(3) = 1; h(4) = 2; h(5) = 1

Cálculo do Suporte

CONTANDO AS FREQUÊNCIAS A PARTIR DA ÁRVORE HASH

Supondo a seguinte sequência de cliente: d = <{1, 5}, {1}, {3}>

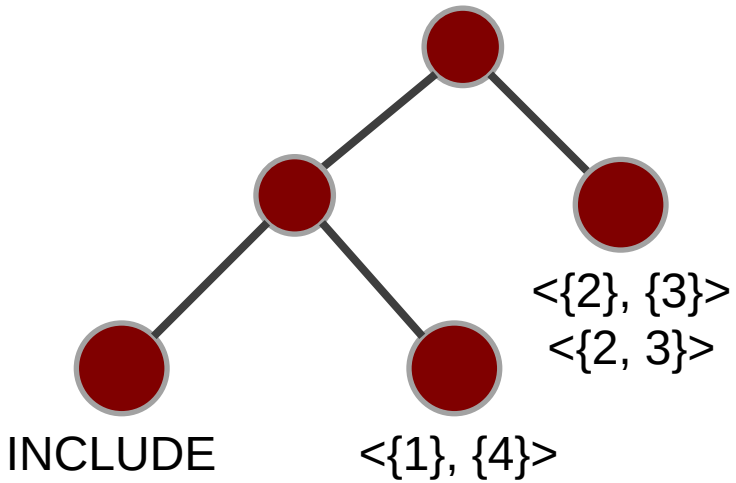


h(1) = 1; h(2) = 2; h(3) = 1; h(4) = 2; h(5) = 1

Cálculo do Suporte

CONTANDO AS FREQUÊNCIAS A PARTIR DA ÁRVORE HASH

Supondo a seguinte sequência de cliente: d = <{1, 5}, {1}, {3}>



h(1) = 1; h(2) = 2; h(3) = 1; h(4) = 2; h(5) = 1

Função INCLUDE

INCREMENTANDO O CONTADOR DOS NÓS FREQUENTES

Temos como entrada uma sequencia S (presente no Nó) e D (cliente)
S = <{2, 4}, {6, 7}>
D = <{1, 2}, {4, 6}, {3}, {1, 2}, {3}, {2, 4}, {6, 7}>

Em seguida, construímos a tabela de tempos:

Itens	Tempo
1	[1, 4]
2	[1, 4, 6]
3	[3, 5]
4	[2, 6]
5	[ ]
6	[2, 7]
7	[7]

Função INCLUDE

Para cada itemset de S, faz-se a verificação. No caso do nosso exemplo, Temos os itemsets {2, 4} e {6, 7}.

A lista de tempos para o primeiro itemset, é [1, 4, 6] e [2, 6].

Verifica-se então em quais momentos, e quais itens apareceram na mesma transação.

O processo é repetido para cada sequência de cliente.

Ao final do mesmo, temos uma árvore hash onde os nós folhas apresentam sequencias candidatas e seu respectivo valor de frequência, ou seja, quantas vezes aquela sequencia apareceu na base.
depois disso, basta percorrer a árvore e retirar as sequencias que atingem o suporte mínimo.

Esse processo de construção da arvore hash é feito a cada geração de candidatos Lk, até que sejam gerados mais candidatos ou nenhum deles atinja o suporte mínimo.