Universidade Federal de Ouro Preto - UFOP

Instituto de Ciências Exatas e Biológicas - ICEB

Departamento de Computação - DECOM

MINERAÇÃO DE DADOS PARA PADRÕES DE SEQUENCIA

Aluna: Cecília Henriques Devêza
Matricula: 07.1.4121

Orientador: Luiz Henrique de Campos Merschmann

Ouro Preto
8 de dezembro de 2010

Universidade Federal de Ouro Preto - UFOP

Instituto de Ciências Exatas e Biológicas - ICEB

Departamento de Computação - DECOM

MINERAÇÃO DE DADOS PARA PADRÕES DE SEQUENCIA

Proposta de monografia apresentada ao curso de Bacharelado em Ciência da Computação, Universidade Federal de Ouro Preto, como requisito parcial para a conclusão da disciplina Monografia I (BCC390).

Aluna: Cecília Henriques Devêza
Matricula: 07.1.4121

Orientador: Luiz Henrique de Campos Merschmann

Ouro Preto
8 de dezembro de 2010

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O constante crescimento de informações decorrentes do desenvolvimento tecnológico tem trazido às organizações um número abundante de dados, aumentando a importância das ferramentas que tem por objetivo extrair informações úteis destes dados. O grande desafio dessas organizações é exatamente transformar os dados em conhecimento. Sendo uma organização comercial, este conhecimento oriundo dos dados históricos pode direcionar melhor campanhas de *marketing*, evidenciar formas mais lucrativas de exibição de produtos, bem como mostrar os clientes mais propícios à compra dos mesmos. Sendo uma organização acadêmica, o conhecimento pode esclarecer questões cujos resultados são impossíveis de serem observados "a olho nu" por pesquisadores frente a um volume tão grande de dados. Além do que, pode encontrar justificativas sobre dúvidas acerca da Medicina, Biologia ou qualquer outra ciência que tem a necessidade de prever resultados com base em dados previamente cadastrados. A técnica de extração de informação mais indicada para este tipo de processo, é a chamada Mineração de Dados.

Mineração de Dados é um ramo da computação que teve início nos anos 80, quando os profissionais das empresas e orgnaizações começaram a se preocupar com os grandes volumes de dados ifnormáticos estocados e inutilizados dentro da empresa. Nesta época, *Data Mining* consistia essencialmente em extrair informação de gigantescas bases de dados da maneira mais automatizada possível. Atualmente, *Data Mining* consiste sobretudo na análise dos dados após a extração, buscando-se por exemplo levantar as necessidades reais e hipotéticas de cada cliente para realizar companhas de *marketing*.[4].

A descoberta de conhecimento, entretanto, não pode ser resumida à Mineração de Dados. Esta, é apenas uma etapa de todo o processo, conhecido como KDD (*Knowledge Discovery in Database*). O mesmo pode ser dividido em:

- Pré-processamento de Dados
  Os dados muitas vezes precisam de uma limpeza antes de passar para a etapa de Mineração. Como as bases geralmente são muito grandes, é necessário remover delas tudo que não vai ser utilizado na próxima etapa, como dados redundantes ou inconsistentes. A qualidade dos dados é que determina a eficiência dos algoritmos de Mineração.

- Mineração de Dados
  É a técnica que será abordada neste trabalho, mais precisamente a área de Padrões de Sequência, onde se busca conhecimento em uma base de dados que segue uma ordem temporal, ou seja, onde os dados estão datatos. Essa base possibilita a descoberta de regras do tipo "Se um determinado usuário comprou um produto X, é provável que ele volte e compre o produto Y".

- Pós-Processamento de Resultados
  Após a extração das regras de conhecimento, é preciso verificar o que saiu como "novidade" dessas regras, ou seja, qual conhecimento estava escondido nos dados que alguém não seria capaz de descobrir sem passar pelo processo automatizado. É recomendado que esta etapa seja realizada por alguém que

conhece o processo sob os quais os dados foram extraídos e saiba diferenciar as regras que são ou não são aproveitáveis.

A figura a seguir ilustra as etapas do KDD:

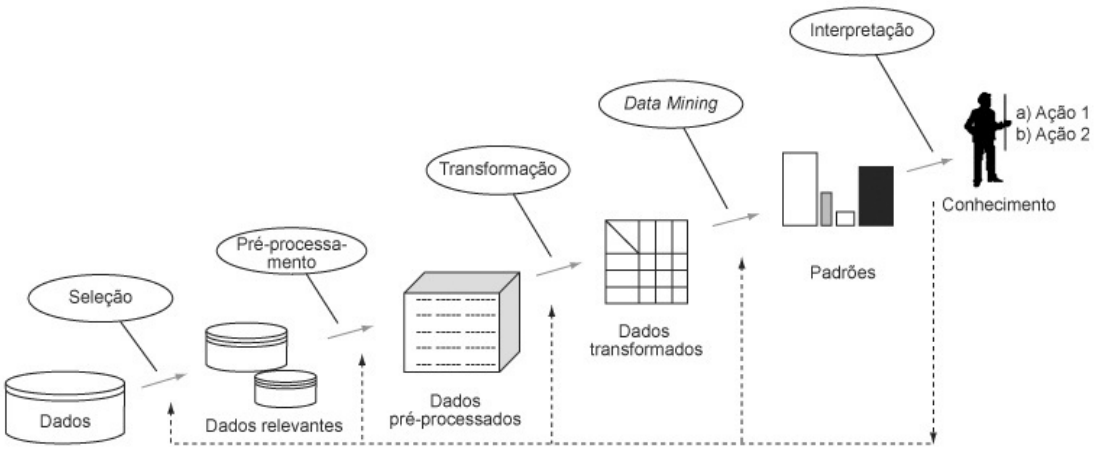

Figura 1: Etapas do KDD

Este projeto irá englobar muitas das etapas exibidas, desde o pré-processamento dos dados até a análise dos mesmos, tendo como ênfase a Extração de Regras de Associação, presente na etapa de *Data Mining*. O problema da exxtração de Padrões de Sequencia foi introduzido inicialmente em [2]. A definição de Padrões de Sequência é uma parte mais específica deste processo, que visa evidenciar sequências de ações/acontecimentos presentes em uma base de dados organizada de forma temporal, ou seja, onde os dados estão datados ou seguem um ordem cronológica. Além disso, é preciso que cada transação da base esteja relacionada a um agente para que as ligações entre as transações possa ser realizada de forma coerente.

A tabela a seguir exemplifica uma base de Dados que pode ser utilizada para descobrir padrões de sequência:

ID do Usuário	**Sequências de Itens**	**Data**
1	3, 6, 7, 1	06/01/2010
2	1, 2	10/01/2010
1	5, 9, 13	03/02/2010
3	2, 5	11/02/2010
3	8	23/03/2010

Tabela 1: Exemplo de Base de Dados.

2 Justificativa

A utilidade deste projeto justifica-se pela crescente necessidade de se encontrar informações úteis de grandes bases de dados. Com a constante automatização de processos das empresas, cresce também o armazenamento de dados enviados e recebidos nestes processos. Entretanto, este dados precisam de um tratamento para que sejam úteis à lucratividade das organizações.

Partindo desde princípio, a empresa GerenciaNet Tecnologia em Pagamentos ofereceu parte de sua base dados que estão relacionados à visualização de produtos em lojas virtuais, sob a condição de que estes dados sejam utilizados única e exclusivamente para fins acadêmicos. A fim de assegurar os direitos dos clientes pela empresa, não foram divulgados: nomes das lojas envolvidas; dados pessoais de clientes que visualizaram produtos nas lojas. Estas informações foram previamente criptografadas.

A base de dados recebida, possui o seguinte formato:

idTransação, idCliente, Produto Visualizado, Loja, Data
idTransação, idCliente, Produto Visualizado, Loja, Data
⋮
idTransação, idCliente, Produto Visualizado, Loja, Data

Onde IdTransação é um valor inteiro que cresce de um a cada tupla, começando por 1. IdCliente é um valor criptografado que representa o cliente que realizou a transação, este valor pode se repetir na base. Produto Visualizado é o nome do produto que foi visto pelo cliente naquela transação (cada tupla mostra a visualização de um único produto). E por fim, Data é o dia, mês e ano que o produto foi visualizado, ou seja, quando a transação ocorreu. A base completa dos dados possui 350MB.

O produto não foi criptografado inicialmente devido ao fato de que a análise das regras geradas ainda é realizada de forma manual. Isso ocorre porque muitas dessas regras podem pertencer a um conjunto de relações óbvias já conhecidas pelos pesquisadores. A ideia seria exatamente garimpar aquelas que são inéditas e utiliza-las a favor do *marketing* da empresa.

2.1 Proposta

A proposta deste projeto é construir um *software* capaz de receber uma base de dados como a descrita anteriormente, e gerar como saída as sequências de itens frequentes nessa base.

Primeiramente, a base deverá passar por um pré-processamento automático, que vai prepará-la para ser utilizada pelo algoritmo GSP (*Generalized Sequential Patterns*) implantado na ferramenta WEKA. A explicação sobre o funcionamento deste algoritmo pode ser vista em 5.3.Este pré-processamento deve gerar um arquivo no formato ARFF, onde os produtos e os identificadores dos clientes são mapeados para valores numéricos. Toda a conversão de dados para possibilitar a leitura pelo WEKA e posteriormente a tradução dos resultados gerados, caberão ao *software*. A geração dos candidatos frequentes a partir do arquivo ARFF caberá ao WEKA. Detalhes sobre o algoritmo GSP podem ser vistos em [3].

A escolha do algoritmo GSP foi realizada após um estudo sobre as principais implementaçãoes criadas para este fim. Após a criação do Apriori, surgiram diversos algoritmos com o objetivo de extrair especificamente regras de associação onde os itens estão ordenados sequencialmente. O algoritmo PrefixSpan é um exemplo. Este algoritmo possui pontos positivos e negativos em relação ao GSP (e ao AprioriAll, explicado na seção 5). Dentre os positivos, podemos citar o fato de que este não passa por uma fase de geração de candidatos, as novas sequencias são obtidas levando-se em conta o banco de dados que é projetado para cada padrão sequencial frequente. Deste modo, o espaço de busca de padrões no PrefixSpan é mais objetivo que os outros. Entretanto, seu ponto negativo está exatamente na construção de projeções do banco a cada padrão sequencial encontrado, este banco é recursivamente projetado em um conjunto de banco de dados menores e os padrões sequenciais frequentes são estendidos, aumentando seu tempo de execução quando se tem um número grande de padrões retornados. Detalhes do estudo de performance e escalabilidade do PrefixSpan podem ser encontrados em [5].

Outro algoritmo criado com a finalidade de se obter padrões de sequencia a partir de bases de dados, é o chamado SPADE. Este algoritmo utiliza propriedades combinatoriais para decompor o problema original em subproblemas, que são resolvidos de forma independente na memória principal, com técnicas eficientes de procura e operações de junção. Detalhes sobre este algoritmo podem ser vistos em [6].

Como a base é desconhecida, o GSP foi escolhido por mostrar uma eficiência que independe da instância, entretanto, como a fase da mineração em si será feita por um programa a parte - o WEKA - não exclui-se a possibilidade de testes da base gerada em outros algoritmos de padrões de sequência como o PrefixSpan, AprioriAll ou SPADE para fins de comparação, verificando-se o custo operacional e tempo de execução destes. Se alguns destes se mostrar bem mais eficiente que o escolhido, o *software* poderá receber uma otimização futura neste para melhoria do desempenho.

3 Objetivos

3.1 Objetivo geral

Este trabalho tem como principal objetivo, a obtenção de padrões de sequência a partir de dados reais de uma loja virtual, para melhor gerenciar campanhas de *marketing* e promocionais, utilizar de estratégias para fidelizar usuários, e outras ferramentas que visa a lucratividade de uma organização e satisfação de seus clientes.

3.2 Objetivos específicos

- Conhecer os algoritmos de Padrões de Sequência e Mineração de Dados em geral;
- Aprimorar o conhecimento da linguagem de programação utilizando diversos recursos para processamento de dados;
- Descobrir regras de sequência a partir de uma base de dados real.

4 Metodologia

Inicialmente será realizado um trabalho de levantamento e revisão de literatura, com o objetivo de fixar os conceitos de Mineração de Dados acerca da resolução de padrões de sequência. Um algoritmo bastante eficiente utilizado para este fim é o GSP (*Generalized Sequential Patterns*). A idéia é construir um *software* que utiliza este algoritmo na etapa de geração de regras, de forma que a base de entrada será aquela fornecida pela empresa, e a saída será o arquivo de regras de padrões de sequência contidas nesta base.

O arquivo de entrada, entretanto, precisa ser previamente processado. Existem neste arquivo informações desnecessárias à uma mesma pesquisa por padrões de sequência, já que visualizações de produtos de 5 lojas distintas estão misturados. Baseando-se nos produtos (que não encontram-se criptografados), é possível verificar que não existe uma correlação lógica entre essas lojas, portanto, é recomendável que sejam escolhidos os dados de uma loja por vez. Após a divisão de dados das 5 lojas, a sequência de uma delas será escolhida para os primeiros testes. É também parte deste processamento, a transformação tanto dos nomes dos produtos quanto dos identificadores de clientes em valores numéricos, visto que os dados precisam estar adaptados ao formato que o WEKA determina. Além disso, as datas não aparecerão de forma direta neste arquivo, mas é preciso ressaltar que os produtos listados no mesmo devem seguir a ordem cronológica definida por elas na base.

Feito o processamento, um arquivo ARFF é gerado pelo *software*. Este arquivo será lido pela ferramenta WEKA, que por sua vez, irá gerar um arquivo de saída com a lista de itens frequentes.

O WEKA é uma coleção de algoritmos de aprendizado de máquina para tarefas de Mineração de Dados. Os algoritmos podes ser aplicados diretamente a um conjunto de dados, ou chamados a partir do seu código Java. WEKA contém ferramentas para os dados de pré-processamento, classificação, regressão, clusterização, regras de associação e visualização. É também ideal para o desenvolvimento de novos modelos de aprendizagem de máquina [1].

Um mês será gasto apenas na realização de testes e possíveis correções do código. Neste passo, serão utilizadas bases de dados diferentes para comparar a análise de execução e dados de saída do algoritmo.

Após todas as modificações necessárias, será feita uma análise sobre os dados retornados. A regra de padrão de sequência, deve seguir o seguinte formato:

$$<\{5\}, \{2,7\}>$$

Onde os valores que estão entre chaves pertecem à uma mesma transação - ou seja, a um mesmo acesso à loja virtual. Cada transação separada por vírgula indica que foi realizada em uma data diferente, seguindo uma ordem cronológica da esquerda para a direita. Este exemplo indica que: "Um cliente que visualiza o produto 5, posteriormente retorna à loja e visualiza os produtos 2 e 7". Este conjunto de transações, por seguir uma ordem temporal, não é comutativo, ou seja, não é possível afirmar a partir do exemplo, que a regra $<\{2, 7\}, \{5\}>$ é verdadeira também.

Os dados nesta forma, entretanto, não trazem muita informação ainda, já que os produtos foram previamente mapeados em valores numéricos. O *software* deve ser capaz de fazer a tradução destes valores numéricos novamente para os nomes originais do produto, para então dar início à análise das regras.

Com as regras retornadas pelo algoritmo, será possível verificar os produtos que, se visualizados, geram uma probabilidade maior de outros determinados produtos serem visualizados também posteriormente, o que pode ser utilizado no *marketing* da empresa afim de direcionar propagandas a clientes específicos, como criar promoções de produtos Y que provavelmente são adquiridos após produtos X, para aqueles que já visualizaram este último. Enfim, este é o primeiro passo para uma melhor campanha publicitária de uma loja virtual, na qual a propaganda é realmente direcionada a clientes que a esperam, evitando desperdício de gastos da empresa, fidelizando clientes e evitando a imagem de *spammer* daqueles que não desejam receber determinadas promoções por não se interessarem pelo produto divulgado.

5 Atividades Desenvolvidas

Durante os primeiros meses do projeto, foram realizados estudos sobre algoritmos de extração de padrões de sequência. O algoritmo que será utilizado neste projeto será o GSP, entretanto, para entendê-lo, é necessário primeiramente entender o algoritmo AprioriAll, que por sua vez, é baseado na técnica Apriori.

5.1 Apriori

O Apriori é um algoritmo que encontra regras de associação, sem se preocupar com a ordem temporal dos itens destas regras - o que foi posteriormente resolvido no AprioriAll. Pra encontrar as regras, é realizado um procedimento iterativo, onde cada iteração executa duas ações: geração de candidatos possivelmente frequentes, e definição dos padrões frequentes. Ao final da execução, possuímos regras de itens frequentes na forma $X \Rightarrow Y$.

Para avaliar se uma regra deve ou não ser considerada, são utilizadas duas medidas: suporte e confiança. O suporte é a probabilidade do antecedente da regra estar presente na base em relação a quantidade total de registros na base. Ou seja:

$$\mathrm{S} = \frac{Xq}{T}$$

Onde Xq é a quantidade de vezes que o item X aparece na base, e T é o total de itens diferentes da base.

Já a confiança, é a probabilidade de o consequente e antecedente estarem presentes juntos relativo aos registros em que aparece o antecedente. Ou seja:

$$\mathrm{S} = \frac{XYq}{Xq}$$

Onde XYq é a quantidade de tuplas em que os itens X e Y aparecem na base, e Xq é a quantidade de vezes que o item X aparece.

Tendo estas duas medidas definidas, o algoritmo começa verificando os itens que atendem ao suporte mínimo. Ou seja, para um suporte mínimo de 0.5, os itens que aparecerem em pelo menos a metade das tuplas na base de dados, serão considerados candidatos frequentes e passarão para a próxima fase.

Propriedade Apriori
Sejam I e J dois itemsets tais que I está contido em J. Se J é frequente então I também é frequente.

Sendo assim, os itens que são "podados" em uma iteração, impedem que uma gama de candidatos certamente infrequentes sejam criados, o que determina que, para que um *itemset* seja frequente, é necessário que todos os *itemsets*s contidos nele também sejam. Se existir pelo menos um que não satisfaça ao suporte mínimo, então sabemos de antemão que qualquer candidato a partir dele não precisa ter suporte calculado, pois certamente não será frequente (evitando nova verredura no banco).

A Figura 2 ilustra o processo Apriori (considerando um suporte mínimo de 50%):

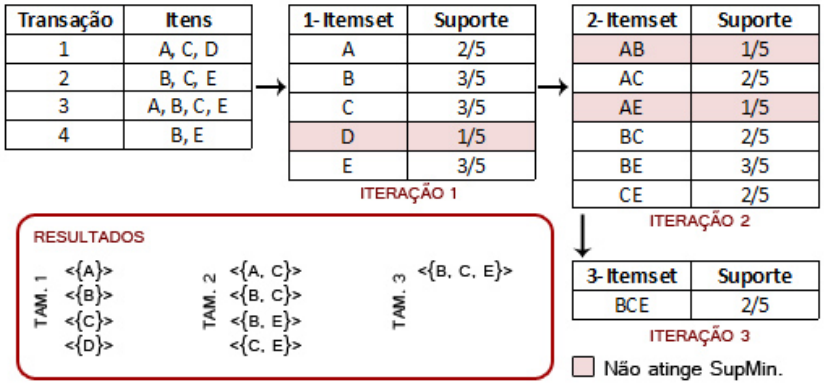

Figura 2: Algoritmo Apriori

Partindo de uma base de dados onde estão listadas algumas sequencias de itens em diferentes transações, cada iteração possui como candidatos conjuntos de itens (chamados *itemsets*) mesclados de acordo com os considerados frequentes na transação anterior. Aqueles que não atenderem ao suporte mínimo são podados e não geram canditados na próxima iteração.

Os resultados que possuem tamanho maior ou igual a dois, se tornam as regras de associação do algoritmo. No caso destes resultados apresentados, teríamos como regras:

- $A \rightarrow C$
- $B \rightarrow C$
- $B \rightarrow E$
- $C \rightarrow E$
- $B \rightarrow C, E$
- $C \rightarrow B, E$
- $E \rightarrow B, C$

Cada regra então passa por uma verificação de confiança. Aquelas que atenderem à confiança mínima, são as regras resultantes do processo. Entretando, esta verificação de confiança mínima não faz parte do algoritmo Apriori em si, o mesmo termina no momento que informa as sequencias de itens frequentes na base. Essa verificação é realizada posteriormente, a fim de garantir resultados mais fededignos.

5.2 AprioriAll

O algoritmo AprioriAll aproveita as premissas desenvolvidas no Apriori, com a diferença de que neste, a ordem com que os itens aparecem é de total importância. O objetivo aqui é encontrar os itens que costumam aparecer na base após o aparecimento de outros itens. Em nossa analogia sobre produtos em uma loja, gostaríamos de encontrar aqui os itens que tem grande probabilidade de serem adquiridos após a aquisão de outros itens.

A base de dados de um algoritmo deste tipo necessita - além da listagem dos itens - da correlação cliente X item, e a data determinante desta transação. Ou seja, se houve uma compra, é preciso saber quando e por quem. A data servirá para organizar as tuplas em ordem cronológica, e a associação com um cliente ajudará a determinar as sequencias de itens que os clientes costumam adquirir.

Para ilustrar o funcionamento deste algoritmo, o processo foi dividido em duas partes: A primeira está relacionada às transformações da base para prepará-la para o algoritmo. A segunda, é a própria execução do AprioriAll.

A Figura 3 ilustra a primeira parte do processo:

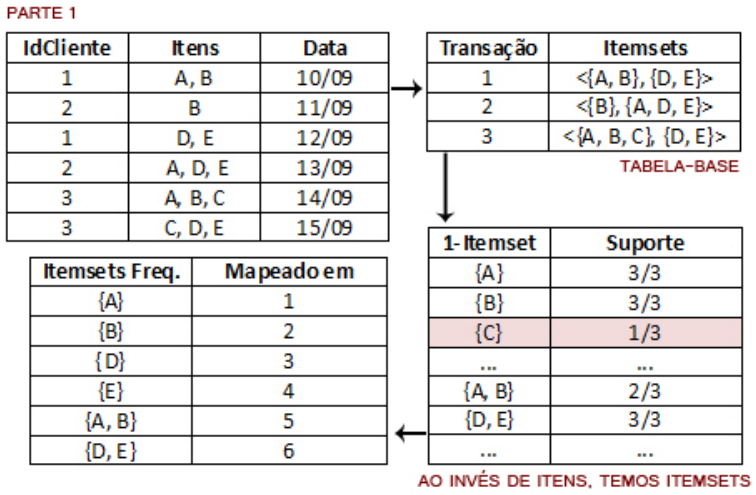

Figura 3: Algoritmo AprioriAll - Parte 1

Nesta etapa, a base de dados original sofre as primeiras transformações para ser preparada para a etapa da mineração em si. Os clientes têm suas transações agrupadas em uma só sequencia, na qual os *itemsets* aparecem seguindo uma ordem cronológica, começando do mais antigo, e terminando com o mais recente. Feito isso, listamos os *itemsets* separadamente para o cálculo do suporte. Os que não atingirem suporte mínimo são podados e não passam para a fase de mapeamento, onde todos os *itemsets*, únicos ou não, são mapeados em valores numéricos utilizados na próxima fase.

A Figura 4 ilustra a segunda parte do processo:

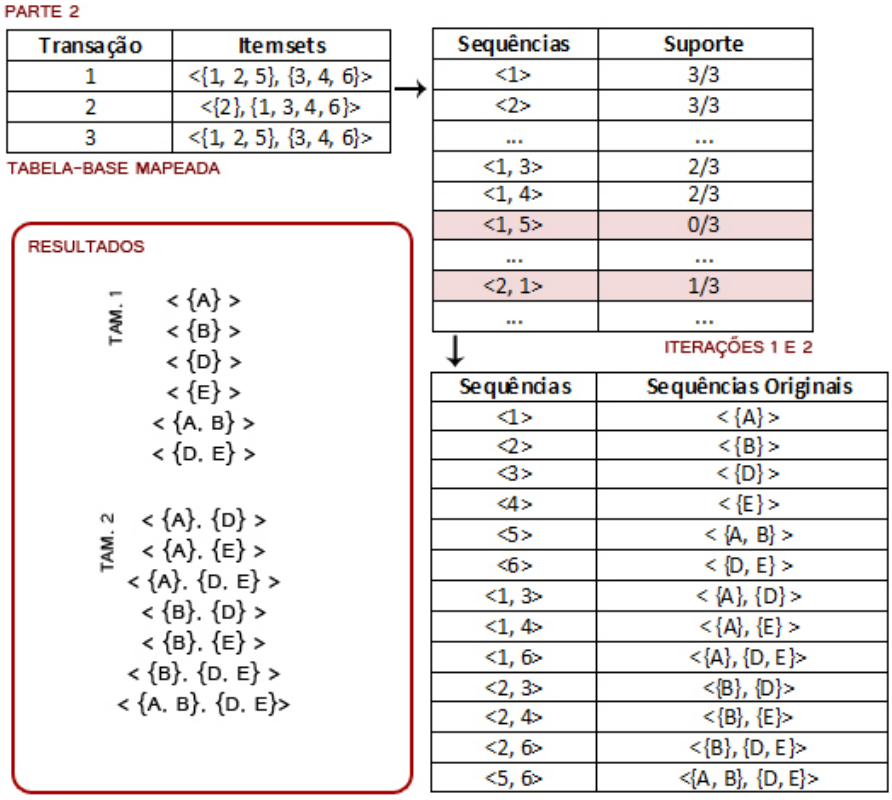

Figura 4: Algoritmo AprioriAll - Parte 2

Nesta fase, a base de dados é transformada de acordo com os *itemsets* que sofreram mapeamento. Neste passo, aqueles considerados infrequentes na etapa anterior serão retirados da base. Com a nova base, é necessário refazer o cálculo do suporte, visto que os *itemsets* agora sofreram modificações. Novas sequencias serão consideradas frequentes, e depois mapeadas novamente para seu valores original, retornando os padrões frequentes da base.

5.3 GSP

O algoritmo GSP se difere do AprioriAll principalmente nas etapas de criação de candidatos e poda de candidatos. Nesta última, são podados muito mais candidatos por iteração, devido à uma otimização na construção de seus pré-candidatos.

Na fase de geração de candidatos:

- No algoritmo AprioriAll, em cada iteração k, os conjuntos Lk e Ck (*Itemsets* frequentes e *itemsets* candidatos) são constituídos de sequências de k *itemsets*.
- No algoritmo GSP, em cada iteração k os conjuntos Lk e Ck (*Itemsets* frequentes e *itemsets* candidatos) são constituídos de sequencias de k itens.

Ou seja, os *itemsets* frequentes <A> e <B> dão origem, no AprioriAll, ao candidato <A, B>. Já no algoritmo GSP, os mesmos dão origem à dois candidatos: <A, B> e <A, B> ou seja, ao invés de derem origem a um candidato que possui dois *itemsets* (conjunto de itens), dá origem a dois candidatos que possuem dois itens, estejam eles em *itemsets* distintos ou não.

Neste projeto não será apresentado o detalhamento deste algoritmo devido à sua complexidade, entretanto, maiores informações sobre o mesmo podem ser encontradas em [3].

5.4 Pré-processamento dos dados

Para dar início aos primeiros testes com a base de dados das lojas, a base já passou por algumas transformações. Algumas procedures no bando de dados indicaram a loja A como sendo uma boa opção para os primeiros testes - Com 589df tuplas, a loja possui 865 itens diferentes, ou seja, um indíce de repetição considerável que pode trazer bons resultados.

Foi criada uma tabela em um servidor remoto, que recebeu todos as 3.525.926 tuplas da base. Uma segunda tabela, chamada lojaA, recebeu somente aqueles itens que foram visualizados na loja A, dando um total de 598.114 linhas (o restante são itens das lojas B, C, D ou E). Estes itens passaram por uma limpeza manual, que teve como objetivo reparar alguns erros de acentuação que estavam distorcendo a contagem, por exemplo:

Existíam na base da loja A, produtos com o nome "Essência de Laranjeira", produtos com o nome "Ess?ncia de Laranjeira" e produtos com o nome "Essncia de Laranjeira". Todas essas tuplas se referem ao mesmo produto, mas por algum motivo no momento da inserção no banco, ou da gravação do arquivo original, sofreram distorções de codificação. Após a limpeza, o valor total de itens diferentes na base caiu de 1501 para 864. Todos os produtos iguais mas com erro de codificação foram unificados em um só produto e tiveram seus nomes corrigidos.

A tabela lojaA estava portanto pronta para ser adaptada ao formato lido pelo WEKA. Cada identificador de cliente, recebeu um valor numérico único por cliente, este valor começa de 1, e é incrementado até o valor 449.059 (total de clientes diferentes da base). O mesmo foi feito com os produtos, entretanto neste, o valor começou de um e foi incrementado até o valor 864 (total de produtos diferentes na base). O arquivo ARFF já se encontra pronto para os primeiros testes.

6 Trabalhos Futuros

O próximo passo deste projeto é testar a base de dados gerada (arquivo ARFF) no algoritmo GSP da ferramenta WEKA. É preciso verificar se a ferramenta tem a capacidade de trabalhar com um volume de dados muito grande, ou se será necessário dividir essa base em pequenas partes a serem processadas. A partir do momento que tivermos um resultado sobre os itens frequentes da base da loja A, é preciso analisar cada uma das regras e verificar se foi extraída uma informação relevante desta base.

Ao término deste processo, daremos início à construção do *software* em JAVA capaz de realizar todos os passos descritos de forma automática. Este *software* terá como entrada uma base de dados como a descrita na seção 1 deste projeto, e deverá gerar um arquivo contendo nomes de produtos que são comumente visualizados / comprados após a visualização / compra de outros produtos da base.

Este *software* possibilitará à empresa, determinar itens com probabilidade de serem adquiridos através da análise e mineração do histórico de compras, o que pode trazer resultados positivos tanto aos donos quanto aos clientes das lojas.

7 Cronograma de atividades

A Tabela 2 mostra o cronograma com as atividades a serem realizadas, começando em Outubro de 2010, e terminando em Julho de 2011. A carga horária semanal prevista é de 20 horas.

Atividades	**Out**	**Nov**	**Dez**	**Jan**	**Fev**
- Estudo de Algoritmos de P.S.*	x				
- Levantamento de Referências		x	x		
- Pré-Processamento da Base de Dados			x	x	
- Implementação do *software*				x	x
Atividades	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**
- Implementação do *software*	x				
- Testes e Correções		x			
- Análise de Resultados			x	x	
- Escrita do Trabalho de Conclusão de Curso				x	x
- Defesa					x

Tabela 2: Cronograma de Atividades.

*Padrões de Sequência.

Referências

[1] Weka - data mining software in java.

[2] Srikant R. Agrawal, R. Mining sequential patterns. 1995.

[3] Srikant R. Agrawal, R. Mining sequential patterns : Generalizations and performance improvements. pages 3–17, 1996.

[4] Sandra de Amo. Curso introdutório de mineração de dados - compilação de notas de aulas. 2006.

[5] J. Pei et al. Mining sequential patterns by pattern-growth: The prefixspan approach. ieee transactions on knowledge and data engineering. 2004.

[6] M. Zaki. Spade: An eficient algorithm for mining frequent sequences. machine learning. 2001.