Universidade do Minho

Escola de Engenharia

Departamento de Informática

Incremental Mining Techniques

Fábio Torres Cavalcanti

Dissertação de Mestrado

2005

Incremental Mining Techniques

Fábio Torres Cavalcanti

Dissertação apresentada à Universidade do Minho para obtenção do grau de Mestre em Sistemas de Dados e Processamento Analítico, elaborada sob orientação do Professor Doutor Orlando Manuel de Oliveira Belo.

2005

Aos meus Pais.

Acknowledgments

Eu poderia escrever uma nova dissertação apenas com agradecimentos a todos que me ajudaram. Acho que todos nós aprendemos a viver e a ser quem somos através de conhecimentos e emoções provenientes de cada um que conhecemos em nossa vida. Assim, só tenho a agradecer todos que conviveram e convivem comigo por serem óptimas pessoas, tendo contribuído para eu chegar aonde cheguei e ser quem sou.

Para atingir este grande objectivo que eu vim buscar em Portugal, começo agradecendo especialmente ao meu chefe, orientador, incentivador e acima de tudo amigo, Orlando Belo. Foram dois anos de muita aprendizagem, trabalho e amizade. Uma pessoa que me ensinou muito não apenas na área académica e técnica, mas também na área humana, com verdadeiras lições de vida. Muito obrigado por tudo e espero que esta amizade continue.

Agradeço também aos funcionários e amigos do Departamento de Informática, em especial a Paula Anjo, Goretti, Helena, Cristina, João, Carla e Jaime.

Aos que fazem parte do grupo de base de dados, que me ajudaram muito nesta batalha com seus ensinamentos, opiniões e troca de ideias. São eles a Anália, e os dois maiores incentivadores e parceiros da área que escolhi para a dissertação, pai Ronnie e o "cara bicho" Pedro Gabriel. Obrigado também ao Nuno pela companhia nos almoços e troca de ideias.

Expresso minha gratidão também aos meus amigos do mestrado, pela receptividade desde o primeiro dia e pelo incentivo que sempre me deram. Aos amigos do projecto P2P, certamente importante

incentivadores e "partilhadores de conhecimento", Tavares, Lopes e José, e também a dona Severina, sempre resolvendo nossos problemas. Aos amigos do pólo aquático, muito obrigado pela convivência nestas duas horas por semana que foram certamente importantíssimas para tirar o stress e dar força e disposição para estudar.

Obrigado aos "brazucas" em Braga, pelo acolhimento e convivência, fazendo com que a saudade de nossa terra seja um pouco reduzida. Sem vocês seria muito mais complicado. Agradeço à família Ronnie, Ana e Pedrinho Miguel, à companhia do Marco e Adri, à Maria, e ao sempre brincalhão Ogi. Junto a esses, agradeço em especial aos que moram ou moraram comigo. Aos que fizeram a ponte para me trazer para o mestrado e me acolheram aqui em Portugal, meus ex-professores e agora grandes amigos, Gustavo e Alfrânio. Muito obrigado pela força, devo muito a vocês. Obrigado Tiago pelo incentivo e companhia desde a época da faculdade, vindo junto comigo para esta luta. Ao grande "oveia" Giovani, parceiro de guerra, obrigado pelo companheirismo "tchê". E finalmente Vitor, o último que chegou com alegria para renovar a força da galera.

Atravessando o oceano e chegando ao Brasil, gostaria agora de agradecer do fundo do coração a uma pessoa extremamente especial em minha vida… Cris. Agradecer é muito pouco comparado a tudo que você fez por mim. Foram dois anos muito difíceis, mas que você confiou, me entendeu e me deu forças para terminar este trabalho. Espero que tenhamos superado todas as dificuldades e que possamos viver com amor e felicidade. Muito obrigado.

Faço um agradecimento em geral aos meus amigos da Universidade Católica do Salvador (UCSal), do Itaigara Park - prédio em que moro no Brasil, e do grupo dos "sumidinhos".

Agradeço a meus tios, primos e avôs, em especial a minha tia, Telma Barros, quem sempre esteve presente, me apoiando e incentivando nesta luta, ao meu tio, José Carlos, também pelo grande apoio e incentivo e a minha avó, Zeza, quem sempre torceu e rezou por mim.

Por último, agradeço de todo coração à minha família. A meu irmão Leandro, sempre companheiro e incentivador, e aos meus pais, Paulo e Iara Cavalcanti. Estes sim são mestres e vitoriosos na arte de viver. Tenho muita sorte em ser seu filho. Sem o apoio, compreensão e amor que vocês têm por mim, eu nunca chegaria ao final deste objectivo. Muito obrigado.

Abstract

Incremental Mining Techniques

The increasing necessity of organizational data exploration and analysis, seeking new knowledge that may be implicit in their operational systems, has made the study of data mining techniques gain a huge impulse. This impulse can be clearly noticed in the e-commerce domain, where the analysis of client's past behaviours is extremely valuable and may, eventually, bring up important working instruments for determining his future behaviour. Therefore, it is possible to predict what a Web site visitor might be looking for, and thus restructuring the Web site to meet his needs. Thereby, the visitor keeps longer navigating in the Web site, what increases his probability of getting attracted by some product, leading to its purchase. To achieve this goal, Web site adaptation has to be fast enough to change while the visitor navigates, and has also to ensure that this adaptation is made according to the most recent visitors' navigation behaviour patterns, which requires a mining algorithm with a sufficiently good response time for frequently update the patterns.

Typical databases are continuously changing over the time, what can invalidate some patterns or introduce new ones. Thus, conventional data mining techniques were proved to be inefficient, as they needed to re-execute to update the mining results with the ones derived from the last database changes. Incremental mining techniques emerged to avoid algorithm re-execution and to update mining results when incremental data are added or old data are removed, ensuring a better performance in the data mining processes. In this work, we analyze some existing incremental mining strategies and models, giving a particular emphasis in their application on Web sites, in

order to develop models to discover Web user behaviour patterns and automatically generate some recommendations to restructure sites in useful time.

For accomplishing this task, we designed and implemented Spottrigger, a system responsible for the whole data life cycle in a Web site restructuring work. This life cycle includes tasks specially oriented to extract the raw data stored in Web servers, pass these data by intermediate phases of cleansing and preparation, perform an incremental data mining technique to extract users' navigation patterns and finally suggesting new locations of spots on the Web site according to the patterns found and the profile of the visitor.

We applied Spottrigger in our case study, which was based on data gathered from a real online newspaper. Our main goal was to collect, in a useful time, information about users that at a given moment are consulting the site and thus restructuring the Web site in a short term, delivering the scheduled advertisements, activated according to the user's profile. Basically, our idea is to have advertisements classified in levels and restructure the Web site to have the higher level advertisements in pages the visitor will most probably access. In order to do that, we construct a page ranking for the visitor, based on results obtained through the incremental mining technique. Since visitors' navigation behaviour may change during time, the incremental mining algorithm will be responsible for catching this behaviour changes and fast update the patterns. Using Spottrigger as a decision support system for advertisement, a newspaper company may significantly improve the merchandising of its publicity spots guaranteeing that a given advertisement will reach to a higher number of visitors, even if they change their behaviour when visiting pages that were usually not visited.

Keywords: Clickstreams; Algorithms and Strategies for Incremental Data Mining; Web Site Restructuring.

Resumo

Técnicas de Mineração Incremental

A crescente necessidade de exploração e análise dos dados, na procura de novo conhecimento sobre o negócio de uma organização nos seus sistemas operacionais, tem feito o estudo das técnicas de mineração de dados ganhar um grande impulso. Este pode ser notado claramente no domínio do comércio electrónico, no qual a análise do comportamento passado dos clientes é extremamente valiosa e pode, eventualmente, fazer emergir novos elementos de trabalho, bastante válidos, para a determinação do seu comportamento no futuro. Desta forma, é possível prever aquilo que um visitante de um sítio Web pode andar à procura e, então, preparar esse sítio para atender melhor as suas necessidades. Desta forma, consegue-se fazer com que o visitante permaneça mais tempo a navegar por esse sítio o que aumenta naturalmente a possibilidade dele ser atraído por novos produtos e proceder, eventualmente, à sua aquisição. Para que este objectivo possa ser alcançado, a adaptação do sítio tem de ser suficientemente rápida para que possa acompanhar a navegação do visitante, ao mesmo tempo que assegura os mais recentes padrões de comportamento de navegação dos visitantes. Isto requer um algoritmo de mineração de dados com um nível de desempenho suficientemente bom para que se possa actualizar os padrões frequentemente.

Com as constantes mudanças que ocorrem ao longo do tempo nas bases de dados, invalidando ou introduzindo novos padrões, as técnicas de mineração de dados convencionais provaram ser ineficientes, uma vez que necessitam de ser reexecutadas a fim de actualizar os resultados do

processo de mineração com os dados subjacentes às modificações ocorridas na base de dados. As técnicas de mineração incremental surgiram com o intuito de evitar essa reexecução do algoritmo para actualizar os resultados da mineração quando novos dados (incrementais) são adicionados ou dados antigos são removidos. Assim, consegue-se assegurar uma maior eficiência aos processos de mineração de dados.

Neste trabalho, analisamos algumas das diferentes estratégias e modelos para a mineração incremental de dados, dando-se particular ênfase à sua aplicação em sítios Web, visando desenvolver modelos para a descoberta de padrões de comportamento dos visitantes desses sítios e gerar automaticamente recomendações para a sua reestruturação em tempo útil. Para atingir esse objectivo projectámos e implementámos o sistema Spottrigger, que cobre todo o ciclo de vida do processo de reestruturação de um sítio Web. Este ciclo é composto, basicamente, por tarefas especialmente orientadas para a extracção de dados "crus" armazenados nos servidores Web, passar estes dados por fases intermédias de limpeza e preparação, executar uma técnica de mineração incremental para extrair padrões de navegação dos utilizadores e, finalmente, reestruturar o sítio Web de acordo com os padrões de navegação encontrados e com o perfil do próprio utilizador. Além disso, o sistema Spottrigger foi aplicado no nosso estudo de caso, o qual é baseado em dados reais provenientes de um jornal *online*. Nosso principal objectivo foi colectar, em tempo útil, alguma informação sobre o perfil dos utilizadores que num dado momento estão a consultar o sítio e, assim, fazer a reestruturação do sítio num período de tempo tão curto quanto o possível, exibindo os anúncios desejáveis, activados de acordo com o perfil do utilizador. Os anúncios do sistema estão classificados por níveis. Os sítios são reestruturados para que os anúncios de nível mais elevado sejam lançados nas páginas com maior probabilidade de serem visitadas. Nesse sentido, foi definida uma classificação das páginas para o utilizador, baseada nos padrões frequentes adquiridos através do processo de mineração incremental. Visto que o comportamento de navegação dos visitantes pode mudar ao longo do tempo, o algoritmo de mineração incremental será também responsável por capturar essas mudanças de comportamento e rapidamente actualizar os padrões.

Palavras-chave: Clickstreams; Algoritmos e Estratégias para a Mineração de Dados Incremental; Reestruturação de Sítios Web.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations and Acronyms

API	Application Program Interface
C_2	Candidate 2-itemsets
CI_SWF	Sliding-Window Filtering with Candidate Itemset
CLF	Common Log Format
CRM	Customer Relationship Management
DB	Database
DELI	Difference Estimation of Large Itemsets
ECLF	Extended Commom Log Format
ETL	Extract, Transform and Load
ExLF	Extended Log Format
FI_SWF	Sliding-Window Filtering with Frequent Itemset
FTP	File Transfer Protocol
FUP	Fast Update
I/O	Input/Output
IP	Internet Protocol
ISE	Incremental Sequence Extraction
ISL	Increment Sequence Lattice
ISM	Incremental Sequence Mining
ISPs	Internet Service Providers
J2ME	Java 2 Micro Edition
KDD	Knowledge Discovery in Databases
MFI	Maximal Frequent Itemsets

minconf	minimum confidence
minsup	minimum support
ODS	Operational Data Store
OLAP	Online Analytical Processing
ROI	Return Over Investment
SWF	Sliding Window Filtering
ULI	Update Large Itemsets
URL	Uniform Resource Locator
WCM	Web Content Mining
WSM	Web Structure Mining
WUM	Web Usage Mining

Chapter 1

Introduction

1.1 The Importance of Data Exploration

Daily, in extremely dynamic market sectors, companies face situations where information quality and knowledge over them play an important and vital role. For surviving in such competitive markets, they must be aware of what is making losses and profits inside them. Therefore, companies should know as much as they can about themselves and everything else that may somehow affect on the way they behave in market, such as inflation, country's economy growth and naturally market share. However, they must give especial attention on acquiring knowledge about their stakeholders – mainly their clients – because that is essentially where their survival comes from.

In the digital era we are living in, people are each day performing more of their daily activities through computers. Examples of such activities include shopping, working, reading newspapers, magazines, and many others. The average amount of time people spend on computers is growing at a rapid pace. As a direct consequence, Web sites are day by day more accessed, enlarging the amount of information stored in companies' databases. With all this information in mind, organizations began to realize the importance of exploring the huge amounts of data stored in their operational systems.

With the e-commerce emergence, many were the companies that promoted the creation of specialized sites for publishing their activities and commercialized products. Rapidly, some of them realized that this new way of being in commercial life, when properly assured and explored, could bring a great profit for their investments. However, to assure their survival in this new world, companies had to develop mechanisms, as in conventional markets, to allow them to know better their customers, or simply their Web visitors, and their behaviour in the sites they visit. Particularly, their interest is to know what are the main actions performed by visitors in the Web sites – searching or buying actions.

In order to publish their goods and products to customers, so they can be largely consumed, companies need to understand how customers behave in the market. Customer past behaviour analysis can provide valuable information for a possible determination of their future behaviour, contributing for enhancing the decision making process and for obtaining better decisions. When developing some type of data analysis process, the main goal is the discovery of client's behaviour patterns from data about these clients, usually stored in company's operational systems.

1.2 Knowledge Discovery through Data Mining

The increasing competitiveness among the organizations urged for developing ways to differentiate their offered services in order to gain market share. The companies, one faster than the others, discovered that they could have great advantages and acquire new knowledge about their business activities if they focused some efforts on the application of data mining techniques over their operational data systems. Data mining techniques can drill through enormous volume of data to discover frequent patterns or relationships that were implicit in the database but previously unknown by companies' management staff. This extracted information can be potentially useful in decision making processes.

According to Fayyad [Fayyad 1996], data mining is part of a larger process known as *Knowledge Discovery in Databases* (KDD). The KDD include several steps, which can be resumed as:

- Understand the end-user goals, the available relevant knowledge and the application domain.
- Choose the dataset from where we want to extract knowledge and submit data to a pre-processing phase, which intends to clean and prepare the data for the further processes.

- Select the data mining technique to be applied, which includes choosing what will be extracted from the mining process (association rules, sequential patterns, etc.), the algorithm itself and the mining parameters and models that will be used.

- Execute the mining technique in order to extract frequent patterns and models.

- Analyze the discovered knowledge through graphs, tables and other views available according to the tool being used, and perform data interpretation and consolidation.

The KDD process is iterative, meaning that after the execution of a task, adjustments might have to be made in preceding steps. Figure 1 illustrates the KDD main steps.

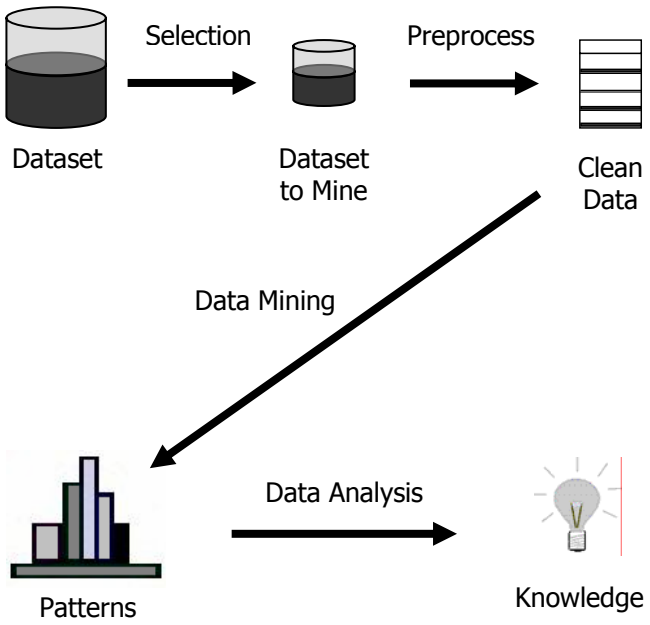


Figure 1: KDD steps

In particular, when performing data mining techniques over companies' repository of information about the navigation processes developed in their sites, fed through data derived from the site usage, we can obtain a valuable insight that can be used in customer relationship field. These data give us important information about the site access, namely, information about the pages, servers, objects or accessed hyperlinks. This information is typically stored in Web servers and specific log files, usually called clickstreams.

When applying data mining techniques over data derived from Web sites, we are dealing with a specific branch of study in the data mining field called Web Mining. Through Web Mining, we are able to acquire a better understanding of the Web and its visitors. From the Web sites, we can get some types of data according to the context we want to mine the Web data. The data extracted can be classified as follows [Srivastava et al. 2000]:

- Content data: data carried to the Web user, usually consisted of text and graphics, typically in form of HTML documents.
- Structure data: data about the organization of the Web sites and information systems.
- Usage data: data collected from user interactions with the Web site.

Therefore, the Web Mining field was divided into three more sub-areas, where each area of study is related to applying mining over one of the collected data types described above. Figure 2 shows the three sub-areas of the Web Mining domain.

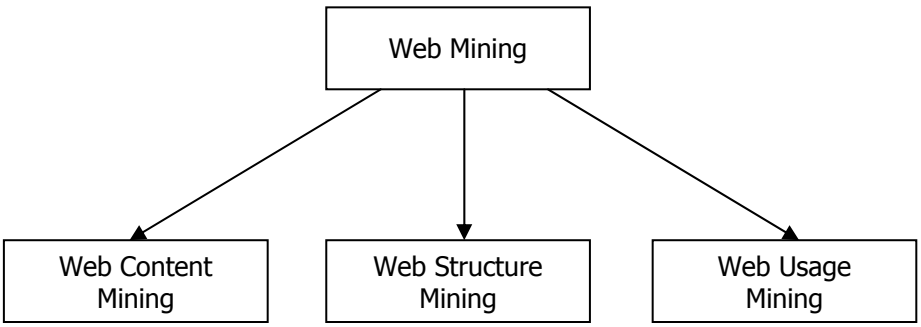


Figure 2: Web mining sub-areas

In this manner, the *Web Content Mining* (WCM) is related to the application of data mining techniques to extract knowledge from the content of the Web documents. The *Web Structure Mining* (WSM), in turn, uses the Web hyperlink structure as information source for the mining. Finally, the *Web Usage Mining* (WUM) uses data derived from clickstreams to study way sites have been used [Cooley 1997] [Srivastava et al. 2000].

We will focus our attention in the analysis of visitors' behaviour in a Web site, dealing therefore with the WUM domain. WUM techniques can be applied for several purposes. Their application helps in decision making processes and provides ways to better attend company's customers. For instance, we can personalize and modify the Web site according to each visitor profile. In addition, we can find bottlenecks or the most frequent accessed pages and services which might be an important characteristic when deciding where the company needs to perform optimizations in the system.

Moreover, stakeholders supporting a Web site generally want to know how much of company's business is going through the site. WUM techniques may answer several of their questions related to customers' behaviour within the Web site and provide statistics of site usage. As a result, the company is able to make decisions intending to achieve customer satisfaction and loyalty, and thereby increase its revenues and profits.

Therefore, the implementation of WUM services in a company has as its main goals the improvement of services quality and customer relationship. These objectives, when reached, will bring a greater profit from their investments on the Web sites, the capacity of attending several directions of their Web clients and the possibility of restructuring in advance their sites for turning them more useful, attractive and pleasant to their visitors.

1.3 Focus and Motivation

It is known that databases in the Web domain quickly change due to visitor navigation processes on the site. Initial data mining techniques suit well for static databases, but are not flexible enough regarding to the addition of new data. In other words, such developed techniques cannot update

their results, considering the new data incorporated into the data repositories, without having to re-execute the algorithm over the whole database.

Incremental mining techniques emerged to solve such problem. Some applications where these incremental techniques have been used are, for instance, to find frequent sequences among the alarms in a GSM network [Zheng et al. 2002] and recommend books for visitors of an electronic bookstore [Veloso et al. 2002a].

In this work, we will cover some of the existing incremental algorithms developed for two data mining types: association rules mining and sequential patterns mining. Firstly, we explain the processes of mining for association rules and sequential patterns. Afterwards, we give a brief explanation of some early developed incremental techniques. Then, we describe some recent incremental algorithms, which have as their main goal the reduction of the required memory and processing (I/O).

Due to the large amount of results that can be generated through mining techniques, we also explain some objective measures of pattern interestingness. Thus, just the most interesting patterns are selected from the mining process.

When mining data derived from Web site usage, we have to understand what kind of information we can acquire from this data and in what possible formats they might be stored. Therefore, we provide a detailed explanation on how clickstreams are stored by Web servers, according to the main existing formats. Along with that, we also point the main problems when handling clickstream data and the crucial preprocessing phase, which these data have to be submitted in order to be clean and put in a suitable format for the application of mining techniques. Further, we explain how is the process of mining data derived from clickstreams – WUM – and the main practical applications of the WUM in the real world domain, which are:

- personalization – profile and analyze customers behaviour to provide customized and personalized services for each Web site visitor [Baglioni et al. 2003] [Mobasher et al. 2000].

- site modification – dynamically restructure a Web site to attend visitors needs [Eirinaki & Vazirgiannis 2003] [Srivastava et al. 2000].

- business intelligence – help organization marketing analysts in decision making processes by providing answers to questions related to business coming into the company through Web site usage [Buchner et al. 1999] [Abraham 2003].

- usage characterization – foresee users future behaviour through the analysis of their navigational strategy when browsing the Web site [Catledge & Pitkow 1995] [Srivastava et al. 2000].

- system improvement – analyze performance of the provided services and Web traffic towards making improvements for turning the system faster, easier and, thus, more pleasant to the user [Srivastava et al. 2000] [Zaiane 2001].

Finally, we apply an incremental mining technique over real data derived from an online newspaper clickstream. The main question of our work is:

How a given company can advertise in an online newspaper having some guarantee that its published advertisement will be seen by most of the newspapers visitors?

It is known that readers of online newspapers may change their behaviour very fast with the occurrence of important events that are rapidly published on the pages. Therefore, in short periods of time, pages that were frequently accessed turn to be low accessed and some previously low accessed turn to be highly accessed. When this occurs, which is a common situation in the Web domain, companies that paid for advertising in frequently accessed pages and expected their advertisements to be often visualized by the newspaper visitors, see their investments going at the wrong way because the pages they advertised are not visited anymore.

Our main goal is providing means to dynamically change the advertisements on a newspaper in a useful time, in other words, reflecting very recent visitors' behaviour. Thereby, a newspaper company may merchandise its advertisements spots guaranteeing that a given advertisement will reach a high number of visitors, even if they change their behaviour by visiting pages that were

usually not visited. The predominantly exhibited advertisements may, for example, be the most expensive ones or ones that the organization has interest on it by some reason, like a partnership with the advertising company.
For this case study, we developed a system that was named by Spottrigger. The system is responsible to get the new data from the clickstream stored by a Web server, clean, filter and prepare these data through a preprocessing stage, and finally perform mining tasks. Due to the incremental nature of the technique employed, we store the results and some other attributes of the mining process to be used in the further mining iterations. By using these stored information, the results are fast updated, which allows us to execute the mining process at regular short intervals of time and guarantee that pages are restructured according to frequent patterns, taking into consideration the navigation behaviour of readers that recently accessed the site.

In short, the goal of this thesis is to study incremental mining approaches, their importance and application in a real practical case. This way, we implement the idea of re-structuring pages according to visitor's behaviour records in a useful time.

1.4 Thesis Organization

This thesis is organized into for more chapters, namely:

- chapter 2 that covers incremental mining techniques, explaining the main pattern interestingness measures and two types of mining: association rules and sequential patterns; then, early and recent incremental mining algorithms were characterized and compared among them.

- chapter 3 refers to clickstreams and how the mining processes are performed over them; we describe the main applications of mining Web data and how the incremental approach fits into this area.

- chapter 4 presents a practical case, aiming on replacing advertisements of an online newspaper pages, where the information to perform the site modification is based on the

results acquired from an incremental mining algorithm; it describes Spottrigger, the system we developed to get the data and perform all the necessary processes to come up with recommendations of what advertisements shall be put in what pages for a given visitor; we show our experiments results, emphasizing the importance of using an incremental mining approach.

- chapter 5 gives some conclusions and point some future works.

The organization followed covers all the aspects related to the development of a data mining system using incremental techniques. Each step involved is carefully detailed, also providing references to treatments or processes not pertaining to our main focus and therefore not comprised in this work.

Chapter 2

Incremental Mining

2.1 General Background and Problem Definition

The amount of information stored in companies' databases is rapidly growing. The experience tells us that the bigger the working data repository is, the bigger the processing time will be. This unsurprising situation reveals that we will have mining processes taking each time longer than before, since those repositories tend to increase in size as new data are added. This is a very common situation.

It is known that databases are continuously changing throughout the time. Thus, every attempt to discover new knowledge would require re-executing the mining processes over any database that had been somehow modified. It is evident that this situation is rather inefficient. So, what can be done? The task of guaranteeing a fast answering time in a mining process is very hard when the databases are dynamic, which means, they change throughout the time due to the insertion or deletion of data. Any change occurred in the database can invalidate some of the existing behaviour patterns, discovered in previous mining processes, or else, add new ones. Some examples of domains where we commonly find these time-changing databases are supermarket purchases, bank transactions, daily records of traffic and meteorology, e-commerce and many

others. In order to solve these problems, a new variant of data mining arouse: incremental data mining.

Therefore, the incremental data mining emerged with the necessity of mining large amounts of data in constantly changing databases, since the previous developed algorithms were inefficient once they needed to be re-executed over the entire database for updating its results. Those initial algorithms ignored the previously performed computation, leading to a duplicate work since most of it was already done. In many applications we might also want to mine the database according to a specific recent window of time, for instance, the last 30 days. Having a fixed-length time window means that transactions are daily added and removed from the target database. Consequently, to attend this requirement, most of the incremental mining algorithms allow not only the addition of new transactions but also the removal of transactions that currently belong to the mining but should not be considered anymore.

The problem of incremental mining, introduced in [Cheung et al. 1996], is defined as follows: Let LDB be a set of frequent itemsets, commonly known as large itemsets, in a transactional database DB. An itemset is a non-empty set of items. After some updates on the database, another set of transactions is added to DB (the problem also includes deletion), resulting in an updated database DB+. The objective is to discover LDB+, that is, the new set of frequent itemsets on the updated database, respecting the minimum support value. The support is the percentage of the itemset occurrence in DB. In [Cheung et al. 1996] was also specified that the maintenance of frequent itemsets involves the research of two itemset types: losers, which are previously frequent itemsets that, after adding the incremental data to the database, became infrequent, and winners. These are itemsets that were not frequent, but turned out to be after the database update.

A system designed with an incremental mining approach considers 2 main tasks. At a first stage, it needs to perform a general mining in the initial dataset. Then, when database changes come, in order to produce the updated results, it uses information derived from previous mining iterations and combines this information with the fresh data in order to find, in an efficient way, the new set of frequent itemsets. Figure 3 shows how an incremental mining process basically works. The dataset where the mining is performed keeps being updated. The results produced by the mining are stored in a database of patterns. According to the algorithm used, some other information might also be stored in the database. Then, the incremental mining uses these stored information for further mining iterations when new data arrives.

Some incremental mining techniques may store a lot of information for a faster update when new data arrives, while others may still have to make some processing in the original dataset. However, the more information is stored, the more memory will be necessary. This might be inappropriate when dealing with very large databases. Therefore, this is one important characteristic to consider when choosing or designing an incremental algorithm. Another important characteristic is the flexibility regarding to modifications in users parameters, such as minimum support. According to each case, it might be necessary to regularly make changes in the mining parameters, thus this flexibility may be a significant feature that one must be aware of when choosing the algorithm.

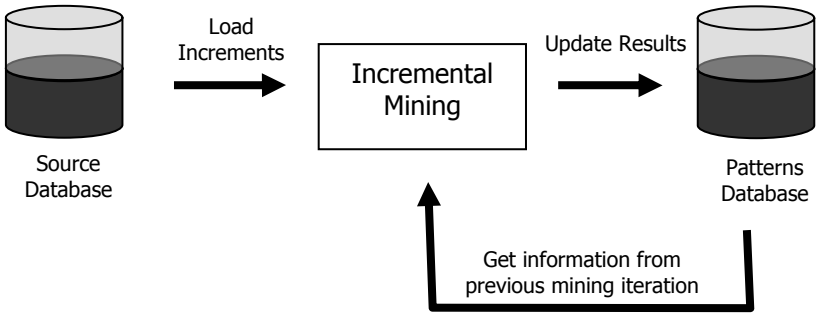


Figure 3: Incremental Data Mining

A relevant question one faces, when dealing with an incremental mining approach, is how often the algorithm shall be applied. If it is applied too frequently over a constantly updated database, with the extreme case of being applied on every new update, the overhead will be large, resulting in taking no benefit from the incremental approach. On the other hand, if the interval between each incremental execution is too long, there is a high risk of either missing important new rules, which were not caught because the last results available are already outdated, or the size of the update might be so big that the incremental approach may not be valuable when compared to running a traditional algorithm over the whole database. These situations show us the importance of knowing the frequency on which the database that will be mined is updated, so the algorithm can be executed in suitable interval of times. Some works have been published to address the problem of determining when to update [Lee & Cheung 1997] [Zheng et al. 2003]. Other efforts to improve the performance of the incremental mining algorithms include solutions to parallelize the algorithm [Parthasarathy et al. 2003] [Veloso et al. 2003].

The existing incremental techniques are essentially based in two data mining types: association rule mining and sequence pattern mining. The main goal of these techniques is to assure a greater efficiency, with resource optimization, mainly in means of memory and processing (I/O). Prior to giving detailed information of these two mining types, we will cover the most commonly used measures in the data mining field for filtering and analyzing attributes such as significance and interest of the rules.

2.2 Commonly Used Mining Measures

A data mining algorithm can potentially generate thousands or even millions of patterns [Han & Kamber 2001]. However, not all patterns generated are interesting. Therefore, in order to generate and filter patterns from a set of transactions containing items, some measures were created to allow the selection of just the most interesting patterns. These measures are statistical and are defined using probabilities to evaluate characteristics such as the frequency of the items, their significance and interest.

When a market analyst of a company is making a strategic decision, he may want to know, for instance, how much the interest of a client for a product B grows after the purchase of a product A? Hence, the measures were conceived to answer such questions and we will give an explanation of the most commonly used ones, namely: support, confidence, lift and conviction.

Support

Initially presented in [Agrawal et al. 1993], the support of an item A is the percentage of transactions in a dataset that contains A, being thus a statistical significance. It is commonly called a frequency constraint, since it is just a count of the transactions containing the given item, dividing it by the total number of transactions. Thus, if A → B has support 30%, it means that 30% of the total number of transactions contain A and B. When an itemset exceeds a minimum defined support threshold, this itemset is considered to be frequent, usually called a large itemset. The support measure has an important characteristic called the antimonotonicity property, which states that all subsets of a frequent set are also frequent. Analyzing this statement in another angle, no

super set of an infrequent set can be frequent. For instance, if an itemset "A, B" is found to be infrequent, we know by the antimonotonicity property that there is no need to analyze the itemset "A, B, C", since its subset "A, B" is already infrequent making it also infrequent. This characteristic is very important to prune the search space and thus increase the algorithm performance. The support main problem is that it misses possibly interesting and valuable rules that may occur in itemsets that are not frequent. The support calculation can be resumed in the following formula:

$$\text{supp}(A \to B) = \frac{\sum \{t \in D \mid (A \cup B) \subseteq t\}}{\sum \{t \in D\}}$$

Equation 1: Support

where *t* is a transaction and *D* is the set of transactions in the database.

Confidence

Also presented in [Agrawal et al. 1993], the confidence, also known as strength, is the probability of a given itemset occurrence (consequent) after the occurrence of another itemset (antecedent). For example, an itemset "C" has 80% of chance for occurring after the itemset "A, B" occurs. Analyzing the confidence attribute, we find that the confidence of A → B is different from the confidence of B → A. The confidence characteristic has the problem that it is sensitive to the frequency of the consequent in the database, meaning that the higher is the consequent support the higher will be the confidence even if there is no association between the items [WWW02]. The confidence of A → B can be calculated with the following formula:

$$\text{conf}(A \to B) = \frac{\sum \{t \in D \mid (A \cup B) \subseteq t\}}{\sum \{t \in D \mid A \subseteq t\}} = \frac{\text{supp}(A \to B)}{\text{supp}(A)}$$

Equation 2: Confidence

Lift

Firstly mentioned in [Brin et al. 1997] with its initial name "interest", the lift attribute denotes how much the probability of an item B occurrence will increase if an item A occurs. This measure is expressed as a ratio, for instance, when A occurs, B will be 4 times more likely to occur. In other words, it indicates the degree of dependency among the items. The lift property has the advantage of not being sensitive to the frequency of the items. Itemsets with low occurrence together in the database can produce very high lift values. Equation 3 shows how the lift measure is calculated.

$$\text{lift}(A \to B) = \text{lift}(B \to A) = \frac{\text{conf}(A \to B)}{\text{supp}(B)} = \frac{\text{conf}(B \to A)}{\text{supp}(A)}$$

Equation 3: Lift

Conviction

Also introduced in [Brin et al. 1997], the conviction measures the deviation of the implication A → B from the assumption that A and B occur independently. It was developed to be an alternative to confidence due to the fact that confidence does not capture direction of associations adequately. Conviction is a directed measure since it also uses the information of the absence of the consequent. In other words, it measures the effect of the right-hand-side not being true. A conviction value equals to 1 (one), means that the items are unrelated. If a rule holds 100% of the time, its conviction is the highest possible value of infinity. Thus, in contrast to lift, conviction is not symmetric and has no upper bound. The following formula is used to calculate conviction:

$$\text{conv}(A \to B) = \frac{1 - \text{supp}(B)}{1 - \text{conf}(A \to B)}$$

Equation 4: Conviction

When we have a high lift, it points out that there is an affinity between the considered items. Additionally, a high conviction can also indicate a strong implication amongst the items. Then, if we

have a high support and confidence, means that the itemset occurs enough so that it may be of high importance, possibly being very valuable in decision making processes. If this situation holds for items of itemsets corresponding to the purchase of products by customers, it may be worthwhile to make an offer involving those items.

There are also other measures which can be applied according to each case necessity. Those that were explained here are just some of the most important and commonly used ones.

2.3 Association Rules

The association rules are the most used models when one wants to discover the influence that one or more items may have over others. For instance, one can analyze that some product P may have a greater probability of being bought when products A and B are purchased.

As initially stated in [Agrawal et al. 1993], the problem of mining association rules comprises two sub-problems. The first is to find all large itemsets, that means, itemsets that are contained in a considerable number of transactions, respecting a minimum support threshold. The second sub-problem corresponds to the extraction of association rules from the large itemsets found, respecting the minimum confidence threshold. Having the large itemsets, the generation of association rules does not require much effort since it is made just by calculations over the large itemsets results, not requiring database scans. On the other hand, the first sub-problem consumes a lot more time, being computationally intensive, due to the huge search space. The search space for enumeration of all itemsets is 2^n, which is exponential in n, with n being the number of items [Veloso et al. 2002b]. Thus, most researches since then have been focused on efficiently finding large itemsets.

The association rules mining problem was formalized in [Agrawal & Srikant 1994] as follows:

> *Let $I = \{i_1, i_2, ..., i_n\}$ be a set of literals, called items; let D be a set of transactions where each transaction T is a set of items, so that $T \subseteq I$; to each transaction is associated a unique identifier, called TID (Transaction ID). We say that a transaction*

> *T contains X, a set of some items from I, if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = 0$. The rule $X \Rightarrow Y$ is valid in the transaction set D with confidence c if c% of the transactions in D that contains X, also contains Y. The rule $X \Rightarrow Y$ has support s in the transaction set D if s% of the transactions in D contains $X \cup Y$.*

Given a set of transactions D, the association rules mining problem is to generate all association rules that have support and confidence greater than a minimum user-defined (minsup and minconf). In a practical manner, when working with association rules, we have an expression of the form $A \Rightarrow B$ which attends the minsup and minconf, where A and B are itemsets, for example:

$$\{bread, milk\} \Rightarrow \{coffee\} (sup=20\%, conf=50\%)$$

The idea behind this is that people who buy bread and milk have a strong disposition to buy coffee. In the above example, this rule occurs in 20% of the total transaction set, and, when bread and milk are bought, the client has a probability of 50% for also buying coffee. In this case we have a common example, which might not be a surprise. However, the interestingness of association rules is to find unusual associations according to each database mined. For example, in a given supermarket we can find that milk goes with bread, but soy milk does not.

The area where the association rules have been more applied is in market basket analysis [Lee et al. 2001]. In this case, the items purchased by customers are stored in the database. Discovering the association rules occurred within this stored dataset, that is, the set of items that are frequently purchased together, may be a lot valuable when making promotions and discounts. It may also help on decisions of how the products may be better arranged in the shelves in order to catch the attention of the highest number possible of buyers. Other fields where the association rules have been used are in telecommunications, e.g. for finding associations in clients phone calls destination, in profiling clients accessing a Web site by finding patterns in their navigation within the site, and many others.

In [Agrawal & Srikant 1994] the first algorithm to solve such problem in static datasets was proposed, called Apriori. This algorithm finds large itemsets iteratively in a breadth-first search, firstly generating candidate itemsets on each iteration and then checking them by a database scan.

That means, for each set of candidates generated, one pass over all transactions in database is necessary. In addition, it uses some techniques to prune the search space for generating a small number of candidate itemsets. Since Apriori was introduced, there have been a lot of studies for improvements following its bottom-up, breadth-first nature, relying on taking advantages from the antimonotonicity property of the minimal support threshold [Goethals & Bussche 2000].

The association rule problem was also tackled in another manner, using a depth-first strategy. The first algorithm following this strategy was the FP-Growth, presented in [Han et al. 2000]. Despite the fact that this algorithm has an efficient counting mechanism, it presents the deficiencies of not taking advantage from the antimonotonicity property, resulting in much more candidate patterns, and its structure implicitly requires the database to reside in main-memory. FP-Growth works well in dense datasets, because it can construct a compressed tree. Dense datasets are the ones where the transactions tend to be similar, having approximately the same length and containing mostly the same items. However, the performance of the FP-Growth falls down when dealing with sparse datasets [Zheng et al. 2001].

One of the problems with the association rules is that a large dataset can lead to a very large number of rules, even if reasonable minimum support and confidence threshold values are used. The confidence by itself is not sufficient in some cases, for instance if all transactions contain B, then, any rule A → B will have a confidence of 100%. Thus, other metrics such as lift and conviction, explained in the previous section, can be used to better filter the rules according to each specific case.

Although the initially created algorithms to solve the association rules problem are somewhat efficient in static databases, they do not treat the problem of maintaining the association rules if database changes occur. The high computational costs involved due to the immense search space when finding large itemsets may be acceptable when dealing with static databases, but not with ever changing databases, since the time consuming itemset enumeration process would be frequently repeated.

Therefore, an incremental mining in association rules consists on the discovering of the item associations in evolving databases. The updates on the database can invalidate some rules,

because the existing rules might no more attend the required minimum support and confidence after the update, or there may be new rules attending the requirements.

2.4 Sequential Patterns

Advances in the way information is stored in databases, helped by technologies such as bar-codes and magnetic cards, have enabled companies' systems to store a very large amount of data in their databases. The records stored by a retail organization derived from customers purchases, usually contain the transaction date and the items bought in the transaction. In addition, they may also contain the customer-id, if it is possible to identify the clients, for example, via customer registration or acquiring the information from credit or loyalty cards. Aiming on analyzing the database to help decision support systems developed for mailing and increasing customer satisfaction, the sequential pattern problem emerged. Later, the problem was brought to other areas such as the scientific and business area [Parthasarathy et al. 1999].

The problem of sequential patterns mining, or sequence mining, essentially consists in finding temporal relations of frequent sub-sequences in a sequence database. Thus, it takes into consideration not only the association of the items according to their occurrence, but also the order in which they occur.

The problem was defined in [Agrawal & Srikant 1995] as follows. Let us consider a database D of clients' transactions. Each transaction is composed of the following fields: client id, transaction time and items purchased within the transaction. No client may have more than one transaction at the same time. The quantity of items purchased in a transaction is not considered. Each item is a binary variable representing when an item was bought or not. A sequence is a set of time-ordered items. A data sequence is composed of all transactions of a customer grouped by in an ascendant order. The support of a sequence *s* is, therefore, the fraction of the total data sequences that contain *s*. This way, the problem is to find sequences with the support greater then the user-defined. Each sequence attending the minimum support threshold is then called a sequential pattern or frequent sequence.

In order to better illustrate the problem, suppose we want to find the sequential patterns in an electronic store. Each data sequence would correspond to the equipment acquisitions of a customer, and each transaction to the equipments acquired by the customer in one order. An example of sequential pattern that could be found might be:

$$\{PC\} \Rightarrow \{DVD\ Player\} \Rightarrow \{Digital\ Photo\ Camera\}\ (sup=80\%)$$

The example above shows that 80% of the customers firstly buy a PC, next a DVD player and, later, a digital photo camera. Note that these purchases do not need to be consecutive as customers who buy other items in between are also fitted to this sequential pattern. Moreover, the items involved in a sequence do not need to be single items. For example, in a retail industry we may have the following sequential pattern:

$$\{Suit, Long\ Sleeve\ Shirt\} \Rightarrow \{Tie, French\ Perfume\}\ (sup=50\%)$$

The sequence above show that 50% of the people buy a suit with a long sleeve shirt, and then come back to buy a tie and a French perfume. In this case, the elements of the sequence are not single items. However, all the items in an element must be present in a single transaction for the data sequence to support the pattern.

Along with the sequential patterns mining problem definition, [Agrawal & Srikant 1995] also proposed three algorithms to solve the problem. The most famous of them, being a reference to many further works, was called AprioriAll. Later, in [Srikant & Agrawal 1996], the problem was extended by adding taxonomy (is-a hierarchy) on items and time constraints such as minimum and maximum gap between adjacent elements of a pattern. A new algorithm called GSP was also proposed in this work, which outperformed AprioriAll up to 20 times. This performance improvement is important specially when dealing with very large databases, where the discovery of all frequent sequences is computationally costly due to the huge search space that grows exponentially according to the length of the longest transaction sequence.

When dealing with an evolving database, as association rules can arise or be invalidated, new sequential patterns can also turn out to be frequent and some previous frequent patterns can become irrelevant with the database updates. That makes necessary the existence of efficient

algorithms for updating, maintaining, and managing the discovered information. The incremental approach should take advantage of previously discovered patterns and, thus, avoid the algorithm re-execution, whenever the data is updated. The problem of maintaining the sequential patterns is more complicated than maintaining the association rules, once the transaction cutting and sequence permutation must be taken into account [Masseglia et al. 2000].
Some distinct areas where the sequential patterns mining processes can be applied are, for example: in the Web domain one can analyze the Web database access, in the telecommunication field alarms can be generated based on the telecommunication network, in the health care industry it can be used in the treatment of diseases from sequences of symptoms and in the financial industry predictions of investments risks can be made based on past sequential stock market events.

2.5 Early Incremental Algorithms

The first incremental mining algorithm, called *Fast Update* (FUP), was presented in [Cheung et al. 1996]. It uses information from previous iterations to generate a smaller candidate set to be checked in the original database. In order to do that, it eliminates in an earlier stage the itemsets that, just by checking the increment, are known to be either frequent or infrequent. This way, it executes the update with a drastic performance improvement when compared to rerunning the algorithm in the whole updated database, as non-incremental techniques do. The first FUP algorithm developed only deals with changes regarding to the insertion of new transactions.

In the author's subsequent work, FUP_2 [Cheung et al. 1997], the algorithm was improved to also handle deletions. When compared to the original FUP, FUP_2 has good performance handling deletions but it is slower if only insertions are made. Although it is generally better than rerunning the algorithm on the whole updated database, it was found that the bigger is the difference between old and new transactional datasets, the bigger is the degradation of FUP_2 performance. If the size difference reaches around 40% of the original database, the incremental update is no longer advantageous.

Afterwards, Lee and Cheung proposed an algorithm called DELI (*Difference Estimation of Large Itemsets*) [Lee & Cheung 1997] which is an extension of FUP that decides whether an update is

necessary, thereby minimizing costs. The algorithm uses statistical sampling to determine how large the difference between the mined dataset and incremental dataset is. If the difference is found to be large enough so that it justifies the overhead of running the algorithm, the updating process is applied through the execution of an incremental algorithm. Otherwise, it states that the current mining results can be used because it represents a good approximation of the present real situation.

Later, the algorithm called ULI (*Update Large Itemsets*) was proposed in [Thomas et al. 1997], striving to reduce I/O cost for updating the set of frequent itemsets by maintaining previous frequent itemsets and the negative border along with their supports. The negative border in the context of sequential patterns mining is consisted of all sequences that are not frequent, which means that did not attend the required minimum support threshold, but both of whose generating subsequences are frequent. When mining for association rules, the negative border is consisted of all itemsets that were candidates, but lacked the minimum support. Maintaining the negative border, a full scan on the database is only necessary if the negative border expands. In the case of ULI algorithm, the database is scanned once, but the incremental portion has to be scanned as many times as the size of the longest frequent itemset.

In addition, some other incremental algorithms developed are PELICAN [Veloso et al. 2001] and MAAP [Zhou & Ezeife 2001]. These algorithms only focus on how to maintain maximal frequent itemsets when the database is updated, and thus they do not allow calculation of the count of non-maximal itemsets. A frequent itemset is considered maximal if it is not a subset of any other frequent itemset. Knowing the maximal frequent itemsets we can determine all sets that are frequent, but not their supports. However, the exact supports can be obtained by counting in the database all distinct subsets of the maximal sets. PELICAN makes its calculations based on vertical database format and lattice decomposition, scanning the search space for frequent itemsets. The efficiency of this algorithm was demonstrated by comparisons to non-incremental techniques, proving that it outperforms them. On the other hand, MAAP uses an Apriori-based framework to calculate the maximal frequent itemsets.

2.6 Recent Algorithms

Pushed by the necessity of urgent analyzing data derived from databases that are updated each time faster, mainly in the retailing industry and in the ever growing e-commerce field, new incremental mining techniques were developed. These techniques aim to reduce even more the I/O costs with lower processing and memory optimization. Early incremental mining algorithms based on the Apriori approach generally suffer from two main problems: a possible occurrence of an immense set of candidate itemsets and the necessity of multiple scans on the database. Therefore, the recent algorithms try to reduce such problems.

Sliding-Window Filtering, ZIGZAG, Incremental Sequence Mining and Incremental Sequence Extraction, are some references of algorithms used by the new incremental mining techniques already available for application in a real world domain. The first two mentioned algorithms are especially oriented for association rules discovering while the last two are for sequential patterns discovering.

Sliding-Window Filtering

The incremental association rules mining technique *Sliding-Window Filtering* (SWF), firstly presented in [Chen et al. 2001], is based on splitting the transactional database into several partitions. It filters unnecessary candidate itemsets by establishing a minimum support threshold on each partition. This way, it carries selectively the information from the previous partition to generate candidate itemsets in the subsequent partitions. Using cumulative filters and techniques to reduce the number of scans in the database, the algorithm minimizes the I/O and CPU costs. Thus, only one scan in the incremented dataset is necessary. Moreover, through the sliding-window partition, it controls memory usage more efficiently. The SWF is suitable for short-term mining over a transactional database, in other words, mining the database according to a specific time window, as it can easily "slide" the considered database by inserting a new incremental partition to it and removing an old one.

In SWF, each new set of candidate itemsets is consisted of the candidate itemsets carried from the set of candidate itemsets in the previous phase that remained candidates after considering the current partition, along with candidate itemsets that were not in the previous phase results but

turned to be selected after considering the current partition. The algorithm results in a cumulative filter composed of the progressive set of candidate itemsets, their occurrence count and related required partial support.

The algorithm first phase is to mine the original database, divided in partitions. The first partition is then sequentially scanned to generate the set of candidate 2-itemsets (C_2). A k-itemset is an itemset composed of k items. The C_2 set, containing the itemset, its occurrence count and the partition in which it started to be frequent, is the only information to be stored for using in the subsequent phase. Once having the candidate itemsets at the end of each partition processing, the set of candidate large itemsets can be generated using the scan reduction technique [Park et al. 1997]. Therefore, C_2 is used to use generate C_k (k = 3, 4 … n). Having the candidate itemsets, only one database scan is necessary to generate the large itemsets. Further, when updates come, the algorithm uses C_2 along with the new data to refresh the results. The removal of a partition from the data considered for the mining process is made just by subtracting the occurrence count of the itemsets in the partition being removed, obviously just considering the ones that were already frequent in that partition, from the last C_2 found.

One example of application were SWF can be efficiently used is on merchandising products in constant evolution, in other words, products having new releases within a short period of time, such as electronic products. New electronic products get in market monthly and almost all of them become obsolete in two or three years from their releases. Therefore, after this time frame they are not sold in the stores anymore, having new models taking their place. Then, the new models sale rapidly grows on its early stage, that is, before the release and growth of another newer one, which a very common situation in this domain. Hence, the SWF algorithm fits in this situation once it easily adapts to changes according to the time. There will be a moment that the old product is no more largely consumed, thus not attending the required minimum support within the partition, being removed from the set of candidate itemsets.

Later, two modifications to the SWF were proposed to increase its performance based on the incorporation of previously discovered information [Chang & Yang 2003]. The experience proved that these two are faster than the original SWF. We refer to:

- *SWF with Frequent Itemset* (FI_SWF), which reuses the frequent itemsets of previous mining processes to reduce the number of new candidate itemsets that have to be analyzed.

- *SWF with Candidate Itemset* (CI_SWF), which reuses the candidate itemsets from previous mining processes.

ZIGZAG

The *ZIGZAG* algorithm [Veloso et al. 2002a] main idea is maintaining only the *maximal frequent itemsets* (MFI), also referred as positive border, to build in an incremental way a frequent itemset grid. This algorithm uses the knowledge discovered in prior phases to reduce the frequent itemset updating cost. The maximal frequent itemsets are updated through a process of retroactive search, leaded by the results of previous mining iterations, being thus an efficient way of determining the frequent itemsets. The idea of a backtrack search for finding maximal frequent itemsets in databases is based in a previously developed non-incremental algorithm called GENMAX [Gouda & Zaki 2001].

The algorithm is also very efficient when the user modifies some parameters, for instance the minimum support. The computation of the support used in ZIGZAG is based on the itemset associativity [Veloso et al. 2003]. This algorithm also supports mining when old transactions are removed from the database and new transactions are added, keeping the set of association rules coherent with the most recent data. This way, it allows a high interactivity degree, being therefore also suitable for mining fixed-length time windows.

ZIGZAG is also appropriate to find reliable association rules by means of the stability property [Liu & Ma 2001]. Stable rules do not have great changes over the time, being thus more reliable. Thereby, it uses these associations to improve the performance of the incremental mining to accuracy. The algorithm also allows establishing a relaxation threshold to update only the frequent itemsets where the variation of popularity exceeds this threshold. The idea behind is that if there are few changes in the data related to an itemset, the rules from its subsets will probably stay very close to the actual rules. By not fully updating some maximal frequent itemsets, the algorithm increases its efficiency.

Incremental Sequence Mining

The incremental sequential pattern mining algorithm called *Incremental Sequence Mining* (ISM) [Parthasarathy et al. 1999] is based on the SPADE [Zaki 1998] algorithm. This algorithm can update the sequential patterns on a database when new transactions and new clients are added to the database. The algorithm efficiently manages the memory by indexing the database creating an *Increment Sequence Lattice* (ISL), which is composed of all frequent sequences and also all sequences located on the negative border on the original database. The ISL prunes the search space for potential new sequences. The support of each member is also stored in the nodes of the lattice, where the children of each node are their subsequences. The algorithm uses a vertical database layout partitioned into blocks, which can fit in memory, where the attributes are associated with the list of transactions in which they occur.

The main idea of the ISM algorithm is that when incremental data arrive, the incremental portion is scanned once to incorporate the new information in the lattice. The new data is combined with the frequent sequences and the negative border to determine the original database portions that need to be rescanned. In order to do this, there is a customer-id index that locates the block where a particular customer-id can be found. Then, a second index locates the items within the given partition. Afterwards, the algorithm makes a linear search for a given customer-id. Thereby, it can quickly jump to portions that will be affected by the update.

Incremental Sequence Extraction

The *Incremental Sequence Extraction* (ISE) [Masseglia et al. 2000] algorithm reduces the computational costs through the reuse of minimum information from old frequent sequences like, for example, the support of frequent sequences. The set of candidate sequences to be tested is, therefore, considerably reduced.

Being k the length of the longest sequence, the problem can be divided into finding all frequent sequences of size $j \leq (k+1)$ and finding all sequences of size $j > (k+1)$. In the former sub-problem, infrequent sequences in the database may become frequent with the update and new sequences that were not in the database may also become frequent. On the other hand, the latter sub-problem is much simpler and can be solved in a straightforward manner with a GSP-like

approach that uses k+1 sequences to generate candidate k+2 sequences and so forth, until it finds all frequent sequences.

For discovering frequent sequences of size $j \leq (k+1)$, the ISE algorithm executes iteratively. In order to better explain how the algorithm works, suppose we have an original database DB and an incremental set db+. First of all, the algorithm scans db+ pruning out infrequent sequences by combining with the frequent sequences in DB and generates the candidate 1-sequences L_1. From L_1 x L_1 it generates candidate 2-sequences, checking them in the updated database to find the frequent 2-sequences. At the same time, it also obtains the set of frequent subsequences preceding items of db+. From this set, associating items of db+ to subsequences of DB, it obtains a new set of frequent sequences called freqSeed. Afterwards, the algorithm generates candidate extensions from the frequent 2-sequences. Later, it uses these candidate extensions along with the freqSeed to generate candidate 3-sequences. Finally, the updated database is scanned for validating the candidate sequences, finding the set of frequent sequences. The process reiterates the generation of candidate extensions and candidate sequences until all candidate sequences of size $j \leq (k+1)$ are found.

2.7 Algorithms Comparison

If we analyze the incremental mining techniques explained in the previous section, we can notice some of their advantages and disadvantages. Looking on the algorithms proposed to find association rules, the SWF is very efficient when we want to mine according to a delimited time space, for instance, the last 6 months. As it works with partitions, the windows are moved throughout the time and only the patterns that meet the required support are passed over to the next updates. The database partition helps on efficiently managing the memory, which is very important when dealing with large databases. However, the algorithm is not so efficient when the support is changed, turning to be somewhat inflexible. The support can only be changed to a lower number without re-executing the mining over the whole database if the new support is only considered in the incremental partitions after its change [Cavalcanti & Belo 2005]. In other words, the algorithm cannot find the missing candidate itemsets, due to the consideration of a lower support, on the already mined partitions without re-executing. This does not occur with higher

support values because all the algorithm has to do is filter the candidates that previously met the support requirement but do not attend to the new minimum value.

On the other hand, the ZIGZAG is more flexible regarding to changes in support value and also performs efficiently when mining according to a fixed-length time window. Nevertheless, it has a drawback in terms of memory requirements since it needs to have its lattice structure in memory in order to perform its database retroactive search to update the frequent itemsets. Therefore, ZIGZAG potentially consumes a lot of memory when dealing with large databases and can turn out to be slower than SWF in this case.

In the case of the algorithms for mining sequential patterns, the ISM is very efficient for small databases and presents more flexibility since it allows changing the support value without having to rescan the original database. That occurs due to the storage of the negative border. However, keeping this negative border consumes a lot of memory, what makes it not very adaptable on large databases. It is obvious that the adaptability to large databases also depends on the hardware used.

The ISE, in turn, looses flexibility by not using the negative border, what increases the probability of having to rescan the database, for instance, when the support value is changed. The candidate set generated can also be huge, making its test phase slow. Another disadvantage is that its level-wise working manner requires multiple scans of the whole database, which is very costly when sequences are long. Although ISE may be some times costly in terms of processing, it generally consumes less memory than the ISM, what makes it more suitable when mining large databases.

We can therefore realize that for choosing an incremental mining technique, we foremost shall analyze some characteristics like the database size, the available hardware and eventual necessities of changes in the mining requirements. Thus, it is possible to choose a suitable incremental mining algorithm for each specific case, balancing important characteristics such as flexibility and fastness [Cavalcanti & Belo 2005].

Chapter 3

Incremental Mining on the Web

3.1 Clickstream Information Sources

As organizations realized the importance of discovering knowledge about their customers, they had to make their sites capture as much information as they could about them when using their sites. For instance, a site-restructure seeking to attend each visitor's characteristic and necessity is only possible if we have some knowledge about their profile. Consequently, Web sites began to adopt several strategies for identifying the profile of the users. Some of them initially asked the user to register, giving some personal data – a "passport" could be delivered – which would give him access to the contents of the site. Nevertheless, many users give up in visiting a given site when they are asked or invited to fulfil forms [Nakov 2000].

Therefore, when this fact was evidenced, some specialized mechanisms for automatic profile discovering were developed. Recent studies and applications added the automatic recording and tracking of users' activities while they navigate on the site. For instance, they store the visited pages, pressed buttons, followed hyperlinks, or the query submitted. Moreover, other information related to a specific user is also stored, such as the IP address, the Web browser used, or the date and time of the site access. All the collected information is stored according to some standard formats, in specific log files, usually called clickstreams. One visited page is, therefore, one of the

several data items stored in the site's clickstream belonging to a session. A session is composed of a sequence of clicks derived from the recording of the actions performed over a site. Thus, the site keeps a complete history of the users' activities, allowing an effective and objective profiling. Before any kind of analysis, it is important to pay attention on what kind of actions can be done and which information can be gathered from clickstreams to achieve effective actions.

Through the clickstreams, it is possible to extract a wide variety of information about the interaction processes between the client and the accessed site. The information captured can provide valuable insight through data mining. If the recording of these actions is performed in a fast way, they can be very important for helping an eventual decision-making process or a site restructuring, according to the visitor profile.

The clickstream data used for applying mining processes can be collected at server-level, client-level, proxy-level, or else acquired from an organization database. Client-level collection benefits from lowering the problems related to caching and session identification. However, it has the drawback that it is made by a remote agent, e.g. Javascripts or Java applets, or modifications on the Web browser, thus requiring authorization and cooperation from the user to enable functionalities or use a modified browser. Proxy-level collection, in turn, may show the real HTTP request from multiple clients to multiple servers, but these data might also not be available since the proxy may be outside of the organization. Finally, server-level collection is the most important one as it records the browsing behaviour of site visitors, not requiring their authorization and being always available since it resides at the same server where the Web pages are stored [Srivastava et al. 2000]. Thus, in this work, we will only consider the clickstream data stored at the server-level.

The Web servers usually follow specific standards to record data from user's navigation processes within the Web logs, probably supported with cookies. The two most popular formats are [WWW03]: the NCSA *Common Log Format* (CLF) and the NCSA *Extended Common Log Format* (ECLF) [Luotonen 1995]. The CLF is the oldest format and the one that contains fewer fields, being somewhat poor in terms of information stored. However, it has the advantage of being accepted by most of the Web servers, being also the default format for many of them. A log entry in the CLF is composed of the following fields: "remotehost", "ident", "authuser", "date", "request", "status" and "bytes".

The ECLF is a variant of the CLF where two more fields were added: "referrer" and "user-agent". Table 1 below explains the meaning of each one of these fields.

Table 1: Extended Common Log Format fields

Element	Description
Remotehost	Domain name of the client that made the request or IP address if the name is unavailable
Ident	Remote user identity information (name)
Authuser	Username as which the user has authenticated himself
Date	Date and time that the request reached the server
Request	The first request line from the client
Status	The three-digit HTTP status code returned to the client
Bytes	The content-length of the document transferred excluding HTTP headers
Referrer	URI of the source of the request
User-agent	Name and version of the client's operating system and browser used

An example of a log in the ECLF is given in figure 4 below, where we identified each one of its fields. Note that the fields related to user identification are not required, having a minus sign "–" character when not supplied.

Although the CLF and ECLF are the most used log formats, they do not provide the transfer time, domain name or cookie information, which disables the possibility of generating some reports. In order to support the necessities of servers, clients and proxies, another log format was developed by the World Wide Web Consortium, the W3C *Extended Log Format* (ExLF) [Hallam-Baker & Behlendorf 1996]. This type of log is self-identifying containing at the beginning of the file log a header with metadata directives about the file content. Therefore, the metadata header tells what information, with its data type, is recorded in the log. Thus, the ExLF is the most flexible among the log formats. Table 2 shows some directives present in ExLF. Among the directives, only version and fields are required ones, the others are optional.

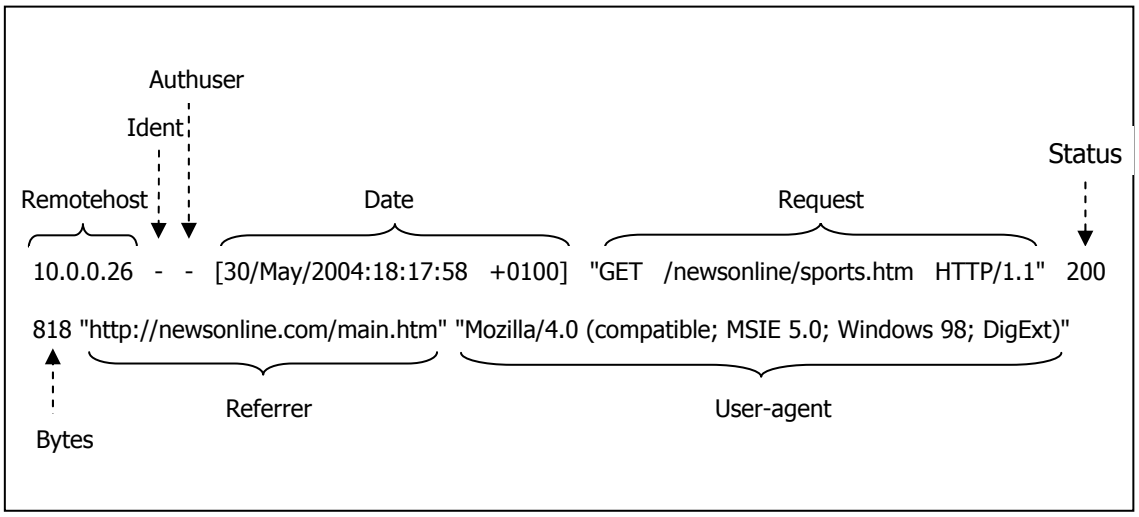


Figure 4: Example of log in ECLF

From the fields recorded by the Web server in the clickstream, we can also obtain extra information by analyzing them. For example, through the IP address it is possible to identify the individual users and user's origin. Moreover, we can also identify the user's sessions including the start page, end page, and all pages visited between those two.

Table 2: ExLF directives

Directives	Description
Version	The version of the ExLF
Fields	List of fields recorded in the log file
Software	Identification of the software that generated the log
Start-Date	Date and time of log creation
End-Date	Date and time that the log was finished
Date	Date an time that the entry was added
Remark	Comment information (shall be ignored when analyzing the log)

If we develop knowledge-discovering activities in clickstreams, we expect to get results that we can use, with significant advantages in the *Customer Relationship Management* (CRM) field, which has as its main goal maximizing customer satisfaction and loyalty. Consequently, the organization will have a greater *Return Over Investment* (ROI) on the Web, and will naturally increase its profits. However, in order to make the data extracted have business sense, so the analysis and decisions made from them are suitable to the company, it is important to merge the acquired information with information related to the Web site architecture, products and services the company is promoting and supplying on the site, and mainly the company's business and marketing objectives.

When we apply mining techniques over Web sites to discover knowledge based on data derived from customer navigation behaviour, we are studying the way the site is being used. In this situation, we are dealing with a specific branch of study in the data mining field called *Web Usage Mining* (WUM).

3.2 Mining Clickstreams

When performing WUM techniques over a site to analyze users' navigation behaviour we are essentially looking towards examining clickstream data. Due to the huge amount of generated data in frequently visited sites, clickstream analysis is very hard to do "by hand" [Ypma & Heskes 2002]. Actually, the sum of data collected each day quickly makes clickstream data difficult to manage and very costly to analyze. Thus, many efforts have been done lately in researches for WUM and the main focus of actions have been in analyzing, treating and mining clickstreams for efficient extraction of Web navigation patterns, as well as the discovery of sorting relations, the prediction of navigation behaviour, or the extraction of other characteristics that may be interesting [Banerjee & Ghosh 2001]. Using this knowledge, companies may identify customers segments to target offline, optimize market campaigns through personalization of Web pages, banners and links or else avoid content delivery issues such as redundant contents or pages too rich in graphics and animations for low bandwidth users.

From the application of mining techniques over clickstreams, one can extract statistics and interesting patterns in order to help in decision making processes. Within the overall process to achieve this goal, the data shall pass by three main phases:

- Preprocessing – clean and prepare data for the mining process.
- Pattern discovery – perform the mining process for knowledge discovering.
- Pattern analysis – analyze, interpret and understand the mining results in order to apply them in decision making processes.

Figure 5 shows the base architecture of a WUM system comprising these three stages.

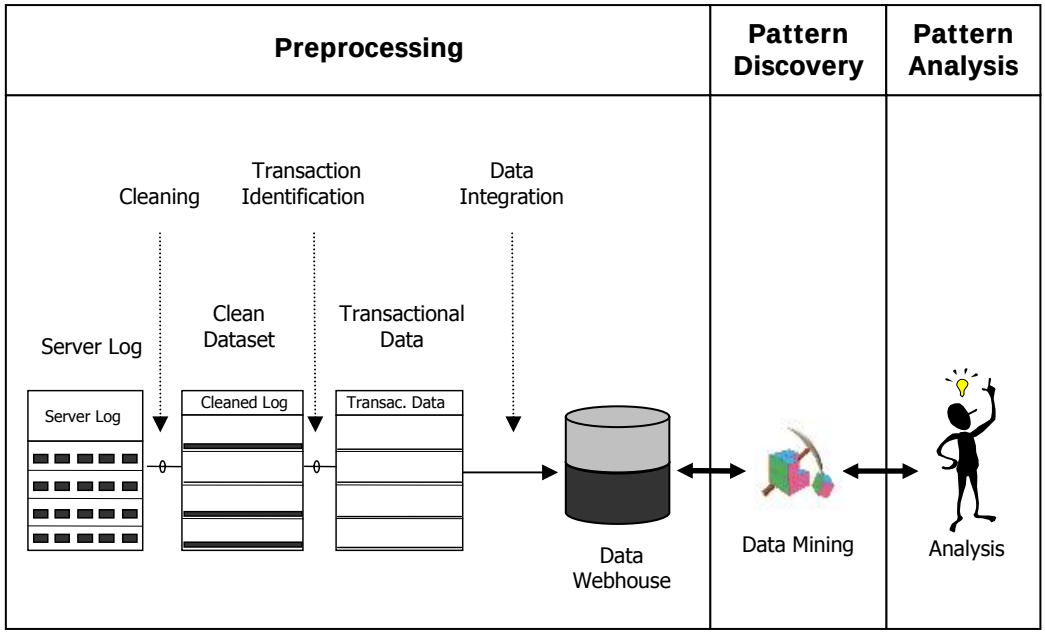


Figure 5: Base architecture of a WUM system

First of all, the Web server stores the data about client navigation in the clickstream. Once stored, the data shall pass through a cleaning process to, for example, eliminate null values or incomplete data, complete some of the incomplete data that may be important, and remove unnecessary data for the mining process. This preprocessing task is the most difficult among the WUM processes, many times due to the incomplete data [Srivastava et al. 2000].

In order to visualize one of the problems (the missing requests) in this phase, suppose that the site being analyzed can be viewed as a tree (figure 6). The page A has a connection to page B, and page B, in turn, links to pages D and E. There is no direct connection from page A to either D or E. We know that if the log file has only requests for A and D, a path completion needs to be made by adding a request for page B to the user. This situation generally occurs due to the caching of Web pages by clients or proxy servers aiming on reducing network traffic. When the page is already in cache, it is not requested to the Web server. Thus, the log record to that page will be missing, resulting in an incomplete path.

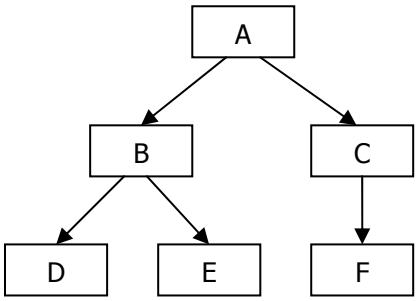


Figure 6: Example of a tree of linked pages in a Web site

Some other errors that commonly occur within clickstream data include, for instance, log entries pointing to invalid pages or mal formed logs originated from proxy access errors. Besides that, in order to have just the data required for the mining process, the cleaning procedure has to filter unnecessary data. When mining to discover user navigation behaviour patterns, we usually only want the requests representing pages of the site. Hence, we need to filter log entries related to requests for images, scripts, style sheets, and others. By removing the useless data, the clickstream size is reduced, what will consume less storage space and facilitate the upcoming tasks. However, whether to keep or not these requests for images depends on the purpose of the mining. For example, if the mining is used for Web caching, the requests for images and multimedia files from logs should not be removed, because these are typically bigger files than HTML documents, and therefore are the most important to cache.

Among the filtering process, another important task that has to be made is to eliminate the requests derived from the navigation of Web robots, also known as Web crawlers. Crawlers can significantly change the access frequency of a given page, since some of them navigate through all

pages they can find a link to. Therefore, low-accessed pages may appear to be frequently visited due to crawler accesses. This has a high chance of occurring if the page in question is referred by links in many other pages, thus robots can easily find it. Most robots can be identified by their IP address or user agent. However, some of them have hidden identity, complicating the filtering task. In order to identify these hidden robots, methods generally based on heuristics are used. Some works have been published addressing this problem [Tan & Kumar 2002].

Moreover, when the mining analysis requires user identification, one frequently found problem is how to identify the unique users and associate them with their access log entries. The easiest way for accomplishing this task is through user registration by asking him to fill out a questionnaire. This method has the advantage of collecting some data that are usually not available in server logs such as rich demographic information. However, many users may simply provide incomplete or false information, or else not go further if the site asks for registration, as referred before. Consequently, the study for the identification of users in clickstreams without requiring user cooperation gained impulse.

One of the difficulties found when identifying users in clickstreams is that single IP addresses may contain multiple sessions. This occurs when *Internet Service Providers* (ISPs) have proxy servers that users access the Web through them. Then, more than one user may access a given Web site at the same time through the same proxy server, having thus the same IP.

A single user may also access a Web site from different machines, having a different IP address in each session. This will complicate the task of keeping track of repeat visits from the same user. Therefore, user identification is a difficult task if the site does not have an authentication mechanism. Some studies proposed heuristics for accomplishing this task [Cooley et al. 1999] [Pirolli et al. 1996], like the identification based on IP, time and user-agent.

Once cleaned and, if necessary for the mining process, with the users identified, the transaction logs need to be identified with the click sequences separated into sessions. Extracting user sessions from Web server logs, especially in the lack of cookie information, is also a difficult work. There are some methods for performing this task but they are all based on heuristics, making us achieve only a relative accuracy. The simplest method for achieving this task is based on a timeout, where we define a time limit between page requests from a given user IP. Every time this

limit is surpassed, we consider that the next page requested belongs to a new session. The most common timeout limit used, which was defined based in several experiments, is 30 minutes [Cooley et al. 1999].

Then, the resulting clean data can finally be integrated in the database that will be mined, typically a Data Webhouse. The term Data Webhouse was firstly introduced in [Kimball & Merz 2000] and it basically refers to a Data Warehouse especially projected for storing data derived from clickstreams, aiming on helping the decision making process inside organizations that develop their activities in an e-commerce environment. The data are stored in a multidimensional form, which makes possible the use of more efficient data analysis techniques, for example, through OLAP systems or data mining.

With the integration of the cleaned and prepared data into the Data Webhouse, the preprocessing phase comes to its end. At this moment, the data is appropriate for the pattern discovery process which is generally made by a data mining technique. The patterns extracted can be, for instance, association rules or sequential patterns as explained in previous sections.

The data extracted from a WUM process can be used for several purposes. Figure 7 summarizes some of the main WUM applications areas, which include as shown: personalization, site modification, business intelligence, system improvement and usage characterization [Srivastava et al. 2000]. In the following sections, we will provide an explanation of these application areas and how they can be applied in the real world domain.

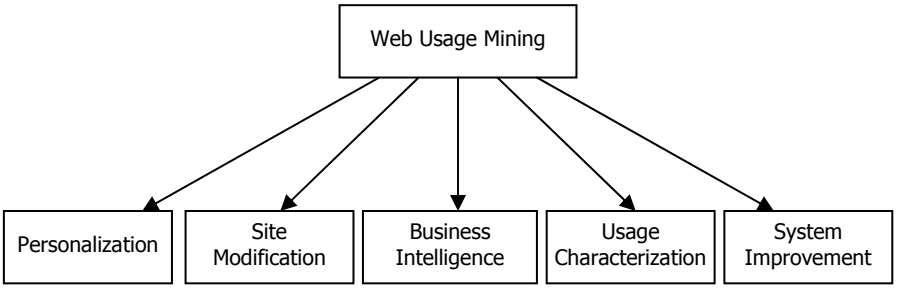


Figure 7: Web Usage Mining Application Areas

Once the data mining extracts the patterns, the third and last phase begins, which comprehends the analysis of the discovered patterns. Patterns analysis can be made by the market analyst or

the Web site administrator, usually through a specialized analysis tool. An analysis tool is extremely necessary if the mining process generates a high number of patterns. Nevertheless, it is also very important for a better understanding of the patterns generated, transforming patterns in easily understandable knowledge. In this phase, the interestingness measures previously explained in section 2.2 can also be used to filter the most interesting patterns. In addition, other characteristics may be taken into account, for instance, one can analyze separately only the new rules produced, that is, the ones that were not in prior mining results.

For analyzing the discovered patterns through an analysis tool, one can perform, for example, OLAP operations, ad-hoc queries (if the patterns are appropriately stored in a database), pattern visualization through graphs and other types of analysis that may be available. Some examples of these tools are WebMiner [Mobasher et al. 1996], which enable us to query the discovered knowledge (association rules or sequential patterns) through a SQL-like mechanism, WebViz [Pitkow & Bharat 1994] and Starfield Display [Hochheiser & Schneiderman 1999], which provide the visualization of traversal patterns through graphs.

3.3 Personalization and Site Modification

An interesting use of clickstream information is the personalization of Web sites to individual user desires [Baglioni et al. 2003]. For instance, one can deliver services and advertisements based on user interests and thereby improve the quality of user interaction leading to a higher customer loyalty. Therefore, Web site personalization is one of the most direct and potentially effective ways for winning the loyalty of the visitors. The Personalization Consortium [WWW04] provides the following definition:

> "Personalization is the use of technology and customer information to tailor electronic commerce interactions between a business and each individual customer"

Customer interests are generally based on what they have looked at on the Web site in the past and especially which sections of the site they visit and spend some time in. Through the clickstream it is possible to extract this information of where in the site customers probably want to

go, what information they seek, what services they utilize and what they buy, if the company in case sells products on the site. From their behaviour patterns, it is possible to determine what else they might be interested in. Thus, the company can profile their visitors in order to provide, for example, advertisements for desired products or links to their desired pages in dynamic customized and personalized Web pages for each visitor. Moreover, the clickstream data can be evaluated in order to measure how effective the customization and personalization of the given Web site was. Thereby, one can take better decisions about improvements on the Web site.

According to [Mobasher et al. 2000], the main elements of Web personalization are: the categorization and preprocessing of Web data, the extraction of correlations between and across different kinds of such data and the determination of the actions that should be recommended by such a personalization system. A personalization system may analyze data through different manners such as collaborative filtering, rule-based filtering, content-based filtering and Web Usage Mining [Eirinaki & Vazirgiannis 2003].

Collaborative filtering is based on asking the users to rate objects and tell their interests in order to predict what information should be of his interest, most of the times based on the interests of other users with similar choices. Rule-based filtering is slightly different since it asks the user to answer questions derived from a decision tree. After following the tree down to its leaves, it comes up to a final result, which should attend user's needs. Notwithstanding, as we previously mentioned, visitors usually do not like to answer questions, rate objects or fulfil forms.

On the other hand, content-based filtering relies on only following users behaviour in the past and recommending items that are similar to the ones they liked in the past. However, user behaviour changes throughout the time. Web Usage Mining will then extract statistics and patterns from user navigation behaviour and use them in order to properly personalize the site. The WUM personalization process usually has an offline part, which corresponds to the preprocessing and pattern discovery phases, and an online part, corresponding to the generation and publishing of personalized content which is based on the knowledge extracted from the offline component [Baraglia & Palmerini 2002] [Mobasher et al. 2000].

One example of personalization that can be made is as follows. Imagine that a visitor named Juan goes into the Sun Microsystems Web site and choose the Spanish language version of the site.

Then he follows the links to the Java programming language section and chooses to download the *Java 2 Micro Edition* (J2ME) toolkit for UNIX systems. At this time, without asking any explicit questions to the visitor, a Sun's software should be able to conclude by analyzing the clickstream that Juan prefers the Spanish language instead of English, he is a UNIX user and is interested in programming for mobile devices. Next time Juan accesses the site, it should be able to self adjust in order to serve the pages in Spanish, filter links to show the ones for UNIX first and give more information about news, available updates and other links regarding to application development for mobile devices. In order to achieve this goal, the recommendations are generally obtained by analyzing data through a data mining algorithm. Therefore, the main goal of personalization is to better serve the customer by anticipating his needs.

One of the most common uses of personalization is on the so-called recommender systems [Schafer et al. 2001]. They are mainly applied in e-commerce sites for making products recommendation and providing customer with information for helping them on deciding which product to buy. This can be particularly important if the site has many products advertised, making customers confused when choosing a product.

In order to process the Web site contents and classifies it in conceptual categories, a content management model can be used. Then, the information acquired from the use of WUM techniques along with knowledge derived from the content management can provide possible alternatives to restructure the site. A publishing mechanism would be responsible for performing this restructuring job and guaranteeing that each user navigates through a site structure personalized for him [Eirinaki & Vazirgiannis 2003].

Initially created Web sites have been designed to attend generic user desires. Personalization can be employed to develop what is called an adaptive Web site, which means that some modifications are made on the site to customize its content and interface to suit individual users or groups of users. An adaptive Web site may, for instance, add, remove or rearrange links, change reformat contents, show specific images or banners, etc. At the beginning, site changes used to be made by hand. As competitiveness among the companies grows at a fast rate, site modifications are turning to be automatically made. Thus, as soon as new frequent patterns are extracted from a WUM process, the restructuring task can be called and the site will provide information according to up to date user navigation behaviour.

3.4 Business Intelligence and Usage Characterization

Knowledge obtained through WUM is very valuable for a company in order to make decisions efficiently related to e-business. The information on how customers are using the Web site allows market analysts to draw strategies aiming mainly on improvements regarding to customer relationship. Customer relationship life cycle involves customer attraction, customer retention, cross sales, customer departure [Buchner et al. 1999]. The effectiveness of promotional campaigns is also another important assessment that can be made through WUM. From there, the company may decide if it is worthwhile making more promotions similar to the one that was made or if the promotion made was not successful. WUM can also provide information on what products were most bought and advertisement click-through rates.

When a company builds a Web site, it is evident that they are interested on how much of its business is coming into the company through the Web site. Organizations are generally concerned on attending the needs and interests of their stakeholders, because that is where their survival comes from. The most important stakeholders of a Web site include the company's customers, suppliers and employees who use the site. Usually, distinct groups of stakeholders have different interests. For instance, some stakeholders may be interested in business and marketing management, while others may be interested in product management and in the technology field. Some questions asked by stakeholders in respect to the company's Web site may be [Bosworth & Schiffman 2001]:

- What marketing campaigns are driving visitors to the site?
- What products are receiving a lot of interest, but they are not being bought at the end?
- Are visitors accessing the information the company is emphasizing?
- Are visitors looking mostly the pages that are easy to find?
- How long is the average sequence of clicks from a visitor since he enters the site to where they want to be, or to where the company wants him to be?

- What behaviours result in more business and what behaviours result in less?

- What does the behaviour of most customers suggest to company's marketing and sales departments?

All these questions are very common and important in the customer relationship field. The answers to them can be attained through WUM. As a result, if the company achieves customer satisfaction and loyalty, it will obviously increase its revenues and profits. The main goal is maximizing the advantages for both: the visitor and the company. Therefore, what will measure and drive the success of a Web site are mostly its visitor or customer experience and relationship with it.

Most of the stakeholders' questions are actually related to site usage characterization. By finding patterns through WUM we are studying how browsers are used and how user interaction happens with the browser interface. Therefore, the navigational strategy of the user browsing the company's Web site is analyzed. Knowing the navigational strategy of most users, one can predict users' future behaviour while he interacts with the Web site.

3.5 System Improvement

When users access a Web site and within it execute an action, performance and quality of the services provided are decisive factors to satisfy them. Through WUM techniques, it is possible to understand how the site is being used and thereby develop strategies for load balancing, network transmission, Web caching or data distribution [Srivastava et al. 2000] [Zaiane 2001]. In addition, it is possible to extract patterns for helping in intrusion and frauds detection, besides the discovering of other security problems.

Once having the frequent patterns in hand, one can realize that some part of the Web site is many times more visited than others, or else find out that a given service within the site is more required than the others. This knowledge allows the organization to take some decisions with a much higher level of confidence. For instance, the company may decide to put some pages or services in a

specific server to improve systems performance and thus provide a faster access to them. One can also realize the need of extra hardware equipments.

In addition, some frequently accessed pages may be redesigned to better attend the kind of users that, according to the behaviour patterns, access these pages of the site. However, if a page, or service, is highly accessed, and because of that it represents a system bottleneck, turning slow its access, the system can be redesigned to distribute it. For example, one can provide alternatives for achieving the same results of that service, or else other pages providing some information that used to be only available through the high accessed one. Knowing customer behaviour through WUM is therefore an important tool for helping in making improvements on a Web site, aiming on better attending its clients, maximizing their satisfaction and loyalty.

3.6 Incremental Web Mining

There are many possibilities for using WUM techniques inside a company. The advantages acquired from that are also very diverse and have, usually, a large impact, mainly on activities related to tracking visitors and site re-structuring. The WUM processes are performed over large dimension, usually static, data repositories. The execution time is usually very large when compared to the time the decision agent has to take any urgent decision.

Databases that hold data derived from clickstreams are always changing. For example, to enable the analysis of Web site visitor behaviour, each time one makes a click for buying a product or just for navigating on the site, the Web server stores the information on the clickstream. Thereby, the database got a new record, what means the early mining techniques would already have to re-execute the algorithm on the whole database in order to update the results for considering this new click.

From the example above, we can see that usage data collection on the Web is incremental in nature. Suppose now we want to mine data derived from a frequently accessed site. In this situation, the database rapidly grows. If we want to personalize such Web site to better attend the needs of visitors, we must be aware that their navigation behaviour may also rapidly change.

Consequently, the behaviour patterns extracted by a mining algorithm may also change very fast – which is a common situation in WUM. For example, important commercial dates, like Christmas, Valentine's Day and others may cause a drastic change in the search for some specific kinds of products or in the visiting frequency of some specific areas of the Web site. Re-executing the algorithm in the entire database would soon produce only undesired results, because when the mining algorithm finally finishes its execution, the behaviour patterns found might be no longer valid. Therefore, WUM algorithms should be scalable and accurate to be applicable to real-life Web sites with a high number of visitors.

Through this example we can visualize the importance of applying the incremental mining approach in the Web domain. The incremental process efficiently adjusts to the new behaviour patterns that may emerge or to the previously valid patterns that may turn out to be invalid in consequence of the updates. Most recent studies regarding to mining Web sites are being performed in this incremental approach.

Figure 8 extends the idea presented before in figure 5 by adding the incremental approach to the WUM architecture. The patterns discovered by the data mining system are now stored in a database. After updating the Data Webhouse with cleaned and prepared data coming from the preprocessing phase, the data mining system uses the already discovered patterns derived from previous mining processes, gathered with the data coming from the update, and refreshes the patterns through an incremental technique. The incremental mining techniques explained in the previous chapter are suitable for this type of application. Therefore, association rules or sequential patterns are examples of valuable information that can be extracted from visitors' navigation behaviour records in the Web site.

Using an incremental approach, the organization market analyst can efficiently get the result information from the system, what can be a very important help when he needs to take fast and effective decisions, in either a short or medium term.

The fast result update enables the company to track client trends and this way it can restructure in advance its Web site in order to make it more functional, attractive and pleasant to its clients. Thus, incremental WUM allows the site to reflect the most recent results by restructuring it with information that is guaranteed to be either the same or extremely close to the real-time frequent

patterns. Therefore, one can restructure the pages according to a given visitor profile taking into consideration the last customer behaviour records, while this visitor is navigating on the site. Site modifications that, with the increasing competitiveness among the companies, have already changed from a manual execution to an automatically way, can now be performed even faster with the mining results being updated much more often.

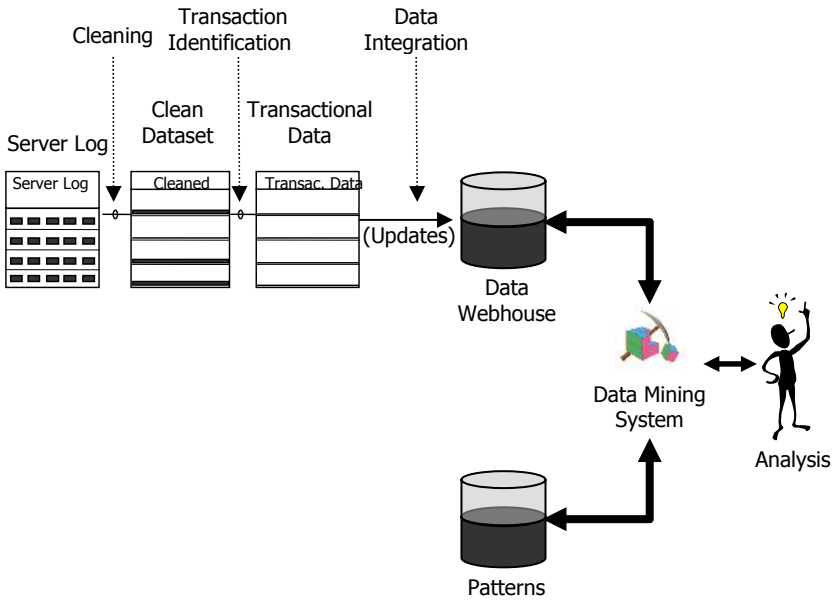


Figure 8: WUM system with an incremental approach

The incremental mining can also provide benefits in terms of system improvements. The incremental processes usually require much less memory and CPU processing for accomplishing the same tasks. Therefore, in some cases that the algorithm needed to be parallelized in several machines to have the results on a reasonable time, now the results can be achieved with a smaller number of machines.

Chapter 4

A Practical Application Case

4.1 The Case Study

The amount of time people are working, studying or just entertaining on computers has been growing very fast. This fact has made the Internet be one of the most important sources of information for many people. As we saw in previous chapters, the analysis of data derived from visitor navigation behaviour is crucial to predict their possible future behaviour and thereby take effective decisions for attending their needs. In this research, we worked on a real case study aiming on analyzing data derived from the navigation of visitors over an online newspaper.

Day by day, people are reading more online newspapers rather than buying printed ones. Hence, newspaper companies realized that if they wanted to keep their clients loyalty, they should turn their attention to this new way of being in commercial life.

Most of the online newspaper content is free, what makes the Web profit come basically from publishing other companies' advertisements. In order to attract companies to advertise on its Web pages, the newspaper company has, in turn, to guarantee that visitors will see the published advertisement, thereby this advertisement can possibly catch the attention of visitors. Organizations interested in advertising their products or services usually seek reaching the highest

number possible of visitors. However, how can the newspaper company guarantee that a given published advertisement will be seen by most of its visitors?

One of the possible solutions is to merchandise the advertisement spots according to the page access rate. Nevertheless, each client has his own interests and accesses different pages. Moreover, some expected or unexpected events such as political elections, tragedies, sport competitions or any other important happening may provoke a sudden change on visitor navigation behaviour. This change may turn some previously low-accessed pages into high-accessed pages and vice-versa.

As explained in chapter 3, when visitors navigate through a Web site, the Web server keeps track of their navigation, storing, among other information, the accessed pages. By applying mining techniques over this stored information, it is possible to extract navigation patterns and profile users. These patterns may be extremely valuable for taking decisions such as dynamically changing the Web site advertisements according to each visitor profile and thereby catch their attention.

Moreover, if an incremental approach is followed, the mining algorithm is able to perceive the behaviour changes and adapt the pages to have the desired advertisements in spots that were not frequently visited, but for some reason turned to be often visited. Thus, a newspaper company may merchandise its advertisements spots, charging for the publicity according to the frequency in which the pages are visited, what will be, consequently, the frequency in which the advertisement published will be seen. This frequency-oriented merchandising is possible due to the fact that we can guarantee through the incremental mining process that a given advertisement will reach a high number of visitors, even if the visitors change their navigation behaviour. For accomplishing this purpose, we suggest to classify the advertisements into levels, where the highest-level ones, which will be the most seen, may either be the most expensive ones or ones that are considered more important for some other reason, like a partnership between companies.

Figure 9 shows how our incremental mechanism works when changing the publicity banners. In short, the main idea is to have the advertisements classified into levels, according to their price and importance, and along with the mining results identify the most visited pages, building a page rank for the visitor's profile. The most visited pages have "n" spots for advertisements. The

available spots, in turn, are classified by the access frequency of the pages they belong and their location within the pages. In the example shown in figure 9, the most visited sections of the newspaper for the given visitor are, in sequence: Main (M), Technology (T), Business (B), Sports (S) and Health (H). Besides that, it shows the distribution of the advertisements according to their levels in the pages spots.

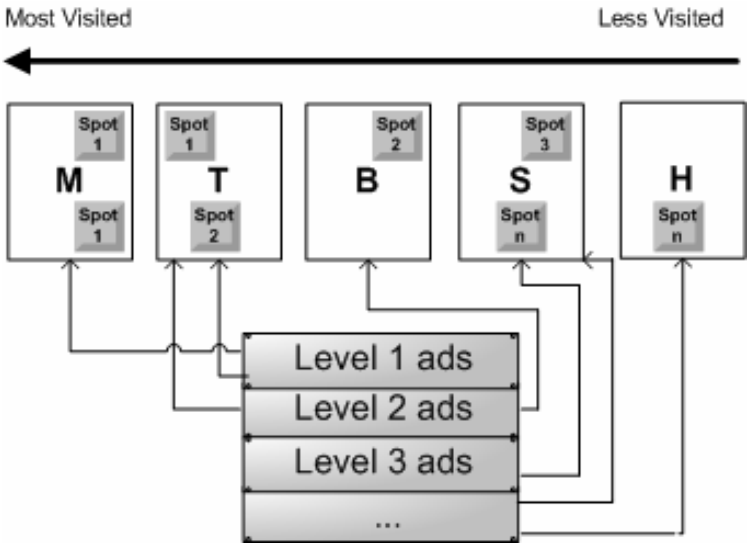


Figure 9: Incremental mechanism for changing page ads

Note that if the visitor starts by the main page and instead of going to the technology section, decides to go to the sports section, after accessing the sports section he will probably be fitted in another pattern with a different arrangement of the most visited sections. Therefore, we can have a pattern telling that people who accesses main and sports sections have a greater probability of also accessing health and not technology or business. Thus, the advertisements will be reorganized so that the top level ones will be now shown in the health section, leaving the lower level ones to technology and business. In this example, since the visitor navigation did not follow the most probable pattern stated at the begging, a lower level advertisement was initially shown in the sports section. However, the top level advertisements are yet guaranteed to be the most seen as this visitor will likely be fitted in another pattern and thus the section with the highest probability of being the next accessed according to this new pattern will be restructured to have the top level advertisements in the spots.

Basically, what we want to do is collect and process, in a useful time, the navigation information about the users that at a given time are consulting the organization's site. Thus, it would be possible to restructure the Web site in a short term and thereby deliver the desired advertisements, activated according to the user's profile.

The incremental mining algorithm combined with the advertisements level database act as a **trigger** to dynamically shoot advertisements in the pages **spots**. Based on this idea, we named our system as "**Spottrigger**". In the following sessions we will show how Spottrigger can dynamically update the Web site, changing advertisements in the pages spots.

4.2 Data Life Cycle

We devise a model through the Spottrigger system for the data analysis process that goes from the storage of information about visitor navigation to the Web site restructure. This analysis process has several intermediate phases. The complete data life cycle is illustrated in figure 10. First of all, we briefly describe each process corresponding to the numbers signed in the figure, indicating after the description its number involved in parenthesis. Later, we provide a better explanation of each one of the cycle processes. It is important to say that even though we show data being stored in databases in some of the processes, our system may also work only with flat files.

The cycle starts when a user is visiting the newspaper Web site. The pages he visits are requested to the Web server (1). The Web server stores visitor navigation information in log files, commonly called clickstream (2). The system regularly requests the clickstream new data and stores them in a Data Staging Area, where some cleansing and preparing processes will be performed. This way, the acquired data will have only the information desired and in suitable format, so they can be loaded into the Data Webhouse (4). Now, the data are ready to be analyzed, in our case, by an incremental mining algorithm.

The incremental mining approach is the chief section of the Spottrigger system, thus we will bring into focus, carefully detailing its steps in a further section. The incremental mining algorithm gets

data from the Data Webhouse and from results of previous mining iterations (5). By processing the incremental algorithm we obtain frequent patterns (6) which represent visitor navigation behaviour. These results are stored in a patterns database (7) for being used by the forthcoming processes responsible for replacing the advertisements on newspaper pages and by the next mining iteration.

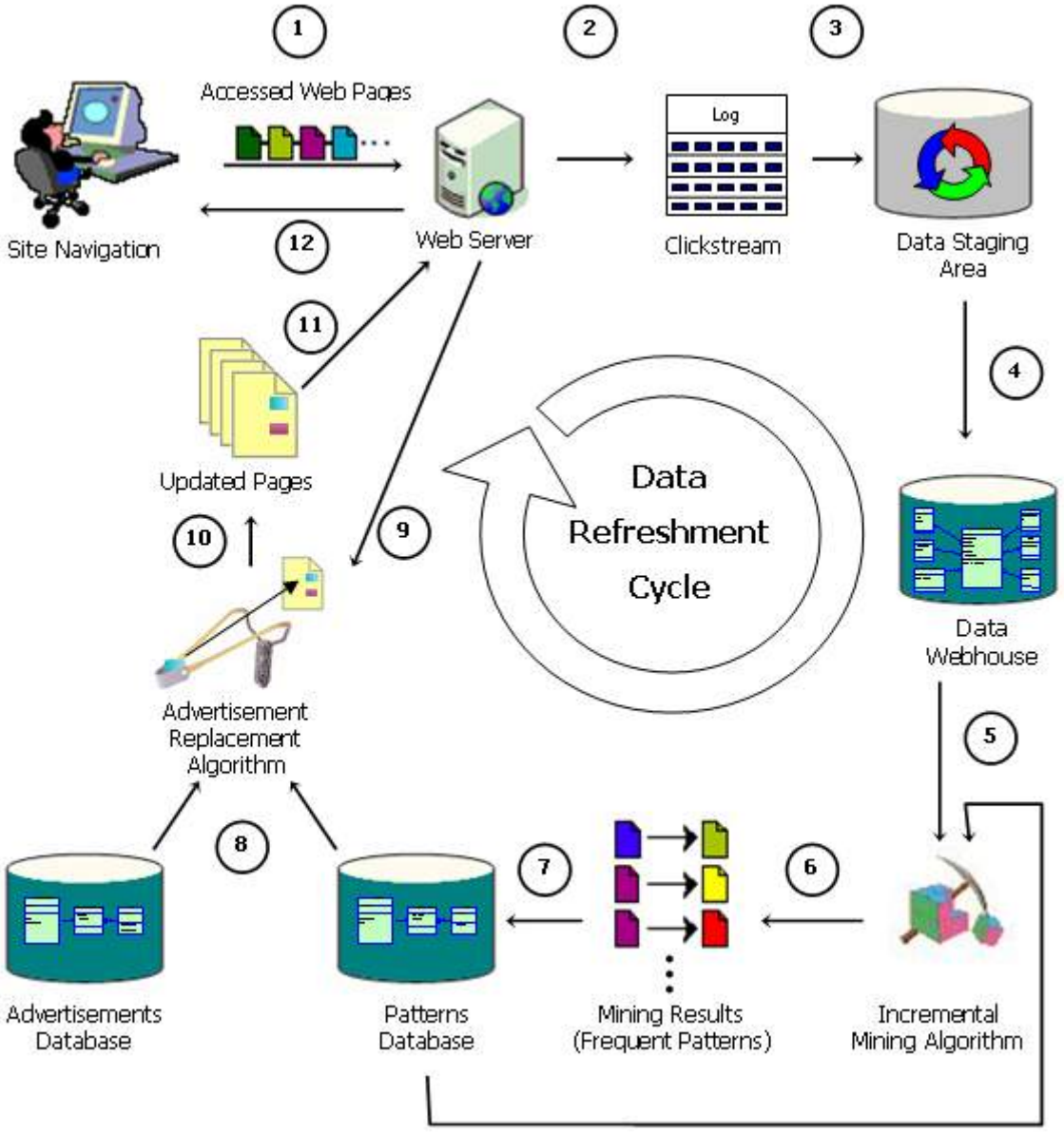


Figure 10: Spottrigger Data Life Cycle

Afterwards, we get to the other main pillar of our system, the advertisement replacement algorithm. As we shortly explained in the last section, the idea is to have the advertisements classified in levels and stored in a database. The advertisement replacement algorithm will process the information from both, patterns database and advertisements database (8), and matching with the data coming from current visitor navigation records (9) will update the advertisements on the page spots (10). These updated pages are, according to the frequent patterns, the ones this given user has the highest probability of visiting. Once updated, the pages are sent to the Web Server (11), so it can deliver the pages with the desired advertisements on the page spots for that visitor (12). From now on, the system cycle restarts.

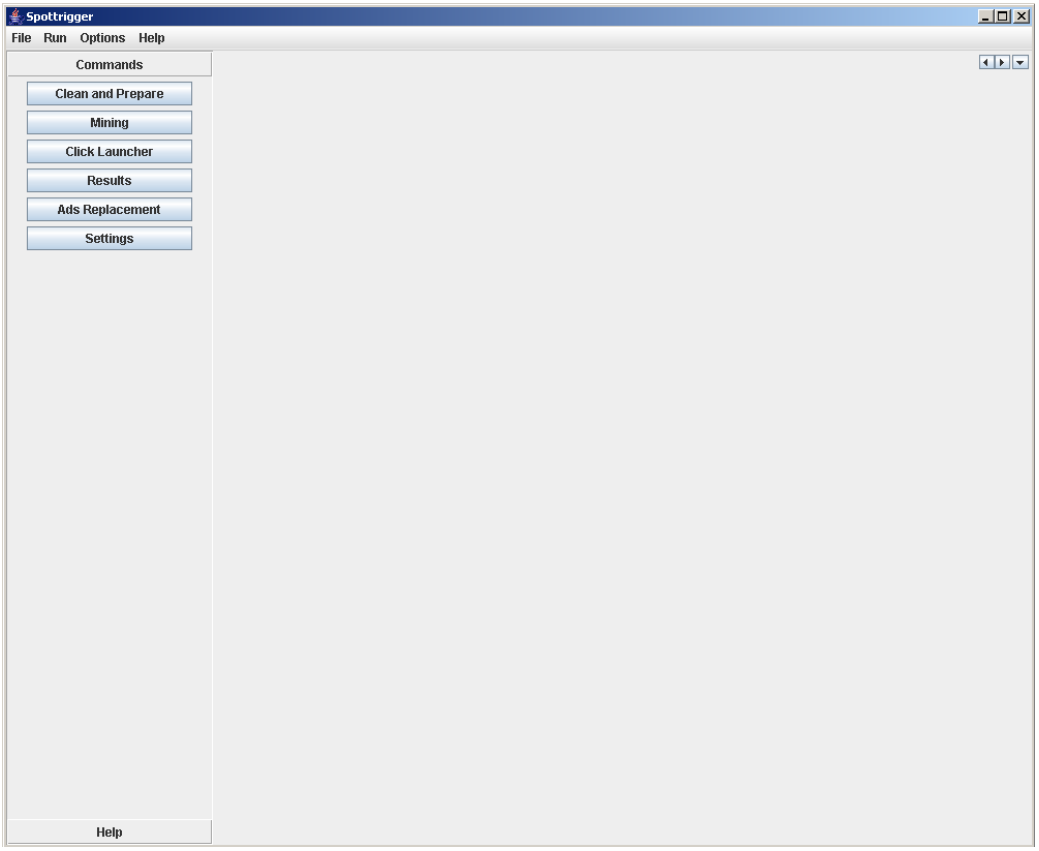


Figure 11: Spottrigger interface

We designed a simple interface in Java [WWW05] for our Spottrigger system to better perform our tests and visualize the results. The interface allow us to interact with the system by, for example, establishing the mining parameters, choosing the files to mine or visualizing the results through

tables that enable us to order the results according to any of the table columns. The initial view of the Spottrigger interface is illustrated in figure 11. In short, through the interface it is possible to choose a file to clean and prepare, perform full mining on the original dataset or an incremental mining if a previous mining has been made, change the mining settings, launch clicks as they were being performed at the moment for a simulation of what occurs in real world domain, visualize the results through tables and simulate a visitor navigation seeing the sections in sequence where the advertisements would be replaced.

In the following sections, we will detail the processes that compose Spottrigger's Life Cycle, explaining our experiments. For the steps possible to visualize through our interface, we will show pictures detailing how we perform them and the results obtained. We evaluated the main processes measuring the time they took to execute with a given amount of data. Our efforts were mainly focused on two particular processes of the system: the incremental mining and the advertisement replacement. It is important to emphasize that some processes on the cycle might include steps other than the ones described in the following sections, which we found uninteresting for this work as they are not part of our focus.

4.3 Storage, Extraction and Pre-processing

Initially, as users navigate through the newspaper's Web site, the pages accessed are requested to the Web server. The Web server, in turn, stores information obtained from every click on log files, the so-called clickstream. Therefore, the clickstream is constantly being updated with new data, a common situation in the Web domain.

The system's first step is to establish an FTP connection to the Web server, pass by the authentication process and make the request to extract the clickstream that will be analyzed by the mining process. This way, the clickstream extracted is firstly stored in the system's data staging area and may alternatively have an *Operational Data Store* (ODS). The ODS is a data structure appropriate for real-time or near real-time operational process support and for tactical decision support. It is usually non-persistent, storing only cleansed and consolidated data for short periods of time [Kimball & Ross 2002]. The clickstream extraction module is visually detailed in figure 12.

The clickstream may be extracted from multiple Web servers if the company has more than one server holding the Web site. When it gets to the data staging area, it contains raw data, which means there may be errors, incomplete data, data that are not in a proper format for the upcoming processes, or even unwanted data. Therefore, after being extracted from the Web server, the clickstream shall pass through a pre-processing phase, which consists in data cleansing and preparing, so it can further be successfully integrated in the decision support systems, typically in a Data Webhouse. This preprocessing work is performed within the Data Staging Area, where the renowned *Extract, Transform and Load* (ETL) processes take place.

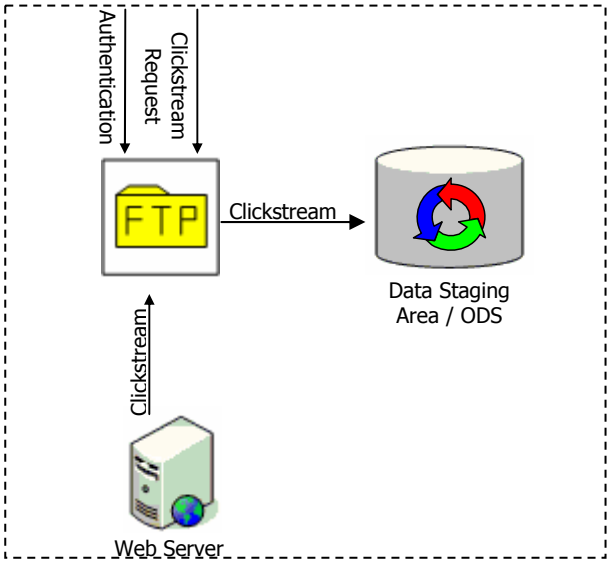


Figure 12: Clickstream Request Module

In our case study, the online newspaper's clickstream contained, for example, unwanted data such as requests for site images and scripts, data generated by the navigation of crawlers or Web robots, and logs with errors, for example, derived from a proxy access denied. Some fields found to be in an inappropriate format were date, time and page. From the page field, we extracted the page category and associated with an ID for applying the mining. We will soon describe how we extracted the page categories from the pages.

Other fields according to each specific case may also demand modifications. For instance, disparate codes for gender, marital status, and other fields may be required to be transformed into the Data

Webhouse standard, or else, companies that sell products or offer services on the internet may be required to change the product or service name by their IDs, since a given product or service name could have been written in different ways (with abbreviations, spelling errors, etc.).

The errors encountered and cleaned in this pre-processing phase are catalogued for a possible future analysis. If the error percentage represents a large part of the clickstream, it may be worthwhile to analyze what is the primary reason of these error occurrences and possibly make corrections on the storage process.

One common and important question regarding to this clickstream extraction process is how often the system should ask for the clickstream updates. In fact, there is no unique right answer for that question. It really depends on the average amount of information stored per minute or per hour, and on how often the upcoming processes require the data to be updated in order to not jeopardize the overall system performance. If the average information flow into the clickstream is low, the system can take longer to get the incremental information, because the overhead of the extraction process to acquire a small amount of information will likely not be compensated by the processing time gain it yields by loading this information sooner. In addition, if we process a small amount of incremental information the final results will likely be the same or have only minor modifications. If we indeed have a high flow of information, which in an online navigation matter means a frequently accessed site, the system has to acquire the clickstream more often as the amount of work to be done by the cleansing and preparing processes will increase and, therefore, the processing time gain will make up for the overhead. Moreover, in this situation, taking longer to acquire the information might signify that you may work with results that are not valid anymore, since the results may quickly change due to the amount of new data inserted within a short period of time.

With an approximately daily average of 800 to 1000 pages requested per hour, our system was designed to request the clickstream every 3 hours. We found that less than 3 hours is not worthwhile since at this value the information processing time is sufficiently low, the overhead caused is not high, the upcoming processes do not need the updated information faster and the amount of information in 3 hours is sufficient to make small changes in the result and not stay outdated, missing result changes, for a long time. In addition, when we reduce the time for performing the incremental mining process we are loosing some information. This occurs because

visitors generally spend a considerable amount of time reading a newspaper. Then, if one starts reading at a given moment and the information stored in the clickstream is acquired during his reading, his navigation will be interpreted as two distinct sessions because the preprocessing phase will not know that his navigation was not finished yet. Therefore, the more we reduce the time interval between the mining iterations, the more sessions will be cut, what may affect in the discovery of long navigation patterns. In order to solve this problem, we would require a mechanism of session controlling so only full sessions would be passed to the incremental dataset to be mined.

In an online newspaper, the pages change from day to day. For example, we can have on one day a page URL http://www.onlinenewspaper/sports/semifinaleuronews.html and a few days later this page might not be available anymore. Besides, a page URL http://www.onlinenewspaper/sports/finaleuronews.html that did not exist in the newspaper section may now exist and become frequently visited. Therefore, if we make our analysis considering the paths at the Web page level of resolution, paths tend to have very little similarity with one another. At such a high resolution, there are very few precise Web page matches between the paths [Alves et al. 2004]. Thus, Web pages can be firstly grouped into categories (that we also call concepts), in our case based on the newspaper sections. When we convert the raw paths to concept-based paths, the average size of the path reduces, and we get paths that can be easily understood. We also aggregate the concepts by merging successive concepts. This way, the mining results will be more concise and accurate making easier and faster the task of connecting page spot, the desired advertisement to publish and reader's navigation for an online advertisement replacement. Table 3 shows how we extract the concepts from the Web pages.

Furthermore, another important task that has to be performed after the cleansing process is the session identification. As explained in section 3.2, the most common method used to identify the sessions is based in assuming a timeout limit of 30 minutes between the page requests from a given user IP. Consequently, we adopted this 30 minutes timeout value in our cleansing algorithm for performing the session identification.

The preprocessing phase is generally the most expensive in terms of processing time, since it is the hardest one due to the amount of validations, corrections, problems with incomplete data and all kinds of processes that have to be performed in order to have accurate data, so we can trust

the analysis that will further be made from them. Although cleansing and preparing data is extremely important, this was not our main focus in this work. Therefore, the algorithm developed for accomplishing this task performs most of the necessary and common treatments but may also miss some advanced ones.

Table 3: Original paths converted to concept-based paths

Original Path	Concept-Based Path
main/home.html main/todaysnews.html sports/semifinaleuroresults.html business/stockmarketnewrecord.html	Main - Sports Business
main/home.html main/urgentnews.html politics/presidentfridaymeeting.html sports/nationalleaguecalendar.html sports/finaleuroresults.html main/todaysheadlines.html	Main - Politics Sports - Main

Through Spottrigger's interface, we can perform the data preprocessing task by clicking at the "Clean and Prepare" button. The cleaning tab will show up with a field for choosing the source file to be cleaned, as we can see in figure 13. Once the file is chosen, all we need to do is click in the "Execute" button. The dataset will be preprocessed and the file with clean data will be stored in a system default directory called "prepared-data". Later, at the mining process, the system is set to look into this directory for getting the file and mine it. After accomplishing the preprocessing task, the process log is shown revealing some statistics, such as the number of requests removed for each filtering procedure (images, crawlers, invalid pages and errors), the processing time and the number of sessions identified.

For our experiment, we firstly considered 15 days of log as being our original dataset. Figure 13 shows our system after preprocessing this raw clickstream, displaying the process statistics. Later, this preprocessing phase is performed in every incremental dataset arrived.

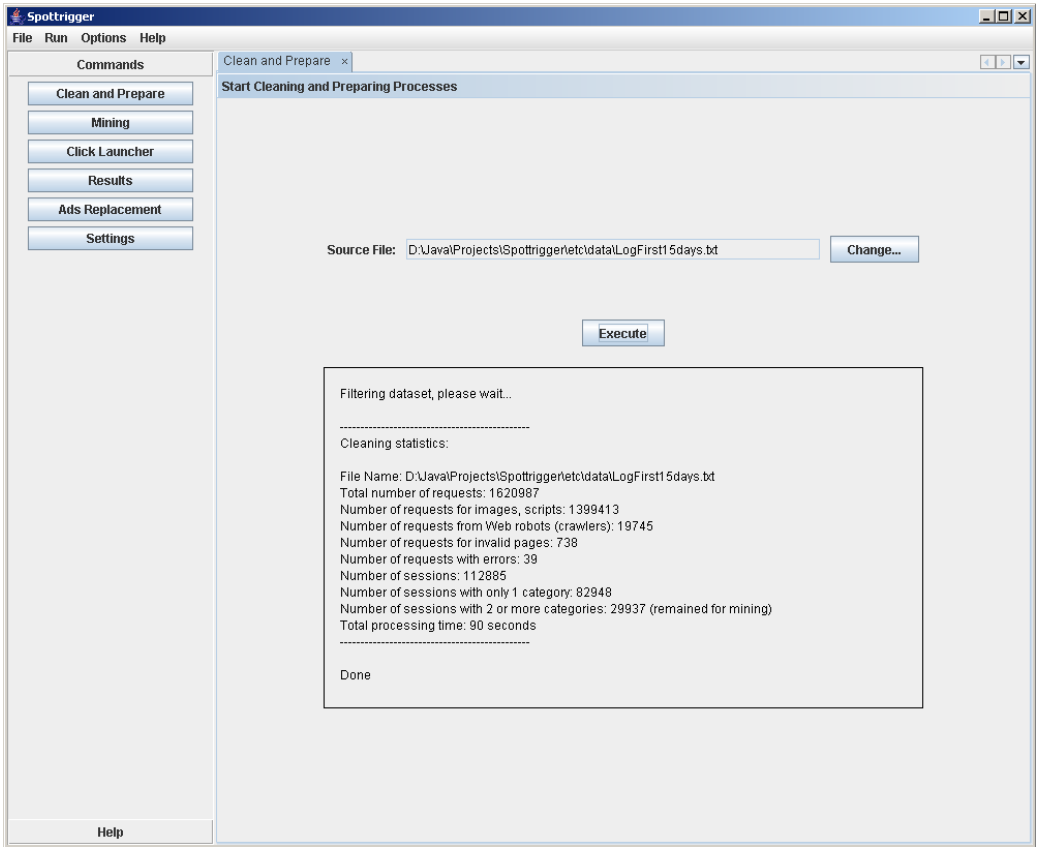


Figure 13: Clean and prepare process

The clickstream considered as our initial dataset – containing 15 days of log recording – has a size of 356 MB. Thus, through the statistics shown in the figure above, we can see that our experiments resulted in a time of 90 seconds for cleaning and preparing 356 MB of clickstream data. Within the processed clickstream, there were a total of 1,620,987 entries, from which we initially filtered 1,399,413 entries, which represents around 86% of it. This occurs because one isolated page request usually generates requests for many images, scripts, style sheets, and others. Therefore, the amount of data filtered is commonly a lot larger than what stays in the cleaned result, unless if our goal to achieve through the mining process claims for using these kind of data being filtered. We also removed a sum of 19,745 requests derived from crawler navigation. Moreover, there were 738 requests for invalid pages and 39 log requests that were recorded with errors.

After the filtering process, we identified 112,885 sessions. From those sessions, we counted 82,948 containing length equals to one page concept. Those sessions with length equals to one mostly point out crawlers, uninterested user or ones who just want to see only one section of the newspaper. It is very hard to deal with sessions like that as we cannot extract associations from only one concept. Therefore, we did not consider these sessions although it is important to know and analyze how much they represent of the total data. If they represent a high percentage of the clickstream, it may be valuable trying to discover the main reason of their occurrence, that is, if they are mostly crawlers that were not caught by the heuristics used in the preprocessing phase, or if the users are not sufficiently attracted to navigate on the site. After the preprocessing phase, we ended up with 29,937 sessions – which is around 26.5% of the total number of sessions – representing the visitors that navigated in at least two sections of the newspaper during the 15 days considered.

Since the preprocessing phase is not our main focus, we did not implement an SQL-like algorithm for preprocessing the clickstream in a Data Staging Area. Nevertheless, when building a Data Webhouse, a carefully designed Data Staging Area is highly recommended in order to have only reliable data. Through our preprocessing algorithm, we can choose to put the resulting clean data in either a file or a Data Webhouse.

4.4 Loading into the Data Webhouse

Once cleaned and in a suitable format, the data can be loaded into the Data Webhouse, which is a particular case of Data Warehouse specially oriented to shelter data derived from clickstreams [Kimball & Merz 2000]. This step is usually made through a bulk loader tool that can be provided by the Data Webhouse database or it can be a third-party one [Kimball & Ross 2002].

Although they are especially projected to hold clickstream data, Data Webhouses share the same basis but they may also have some distinct dimensions according to each specific case. For instance, there is no need for having a product dimension if the company does not sell products. An example of data mart for clickstream data that can fit in our online newspaper case is shown in figure 14. It is important to notice that even though the activity may be the same, differences may

still occur in the Data Webhouse. In our example, we do not have a client dimension as the site we worked on does not require authentication and we did not implement any mechanism for client identification. However, there may be another newspaper Web site requiring user authentication, for instance, if its contents are only for registered users. Hence, depending on how the Web site is designed and what is recorded by the Web server, it may be possible or not to have the values for a client dimension.

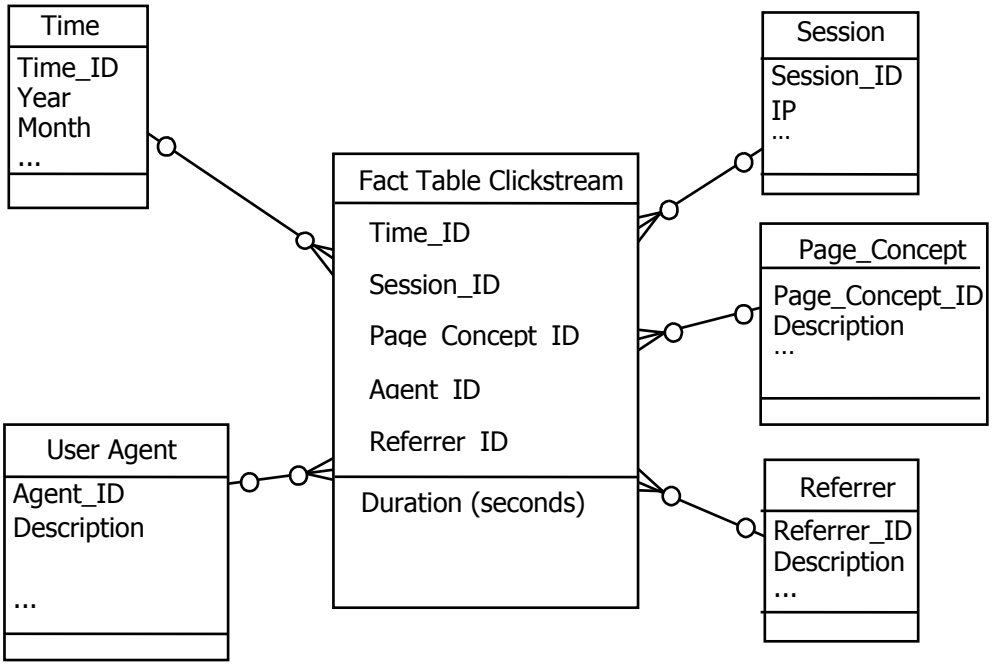


Figure 14: A partial schema of a data mart for clickstream data

Although our developed algorithms are completely prepared for databases, since we implemented a database handler capable of making the connection and read/write data on it, our final experiments were mainly based on storing the information in flat files. In our initial tests, we were reading and writing data in a MySQL database. However, we found that the overhead of connecting to a database for reading data and writing the results was not advantageous when taking into consideration the amount of data we processed.

4.5 Incremental Mining over Newspaper Data

Continuing the data cycle, the subsequent step is to apply the incremental mining algorithm, which uses data obtained from the Data Webhouse along with the results from previous mining iterations stored in a patterns database to update the frequent patterns. This incremental mining step is the heart of the Spottrigger system. Through the incremental approach we can guarantee a fast update on the mining results and, hence, ensure that the current navigation behaviours are taken into account when choosing the advertisements to put in the page spots. Therefore, along with the advertisement replacement algorithm, it is where the system relies to come up with fast, accurate and updated information for changing the advertisements in a useful time, in other words, in real-time or in the closest it can get to real-time.

Once the mining algorithm is applied, the results produced are frequent patterns, which in our case are association rules, representing the navigation behaviour of the site visitors. One example of association rule in an online newspaper may be: when a reader accesses the Sport section, he has a high probability of accessing the Business section. With this knowledge in hands, when one accesses the Sport section, it is possible to previously update the Business section spots with the desired advertisement according to this user profile.

It is important to highlight that the Spottrigger system needs an incremental mining algorithm, but does not requires it to be a specific one. The incremental mining step is like a black box that receives clean data from a Webhouse (or a flat file), computes along with the results from previous mining iterations, and updates the frequent patterns. Therefore, any incremental algorithm that meets the requirement of generating frequent patterns can be used.

In order to simplify and clarify our explanation, we will firstly describe how the mining process works, assuming a small dataset containing only part of the newspaper sections. Later, we will give more details of our implementation and the results obtained when processing our real data in which we had 15 days of log for the initial dataset and some incremental datasets that we added containing 3 hours of log.

Remember that we take into account the page concepts and not the complete page URL. Thus, for this short example, we consider the following available page concepts: Main, Sports, Business,

Technology and Health, which we will identify in our example by their first letter. For these pages, suppose we have the following initial set composed of six transactions:

Initial dataset = {(M, S, H), (M, H), (M, S, T), (M, S, B), (S, H, T), (S, H, B)}

Through this short set of transactions, we can notice that a visitor may enter on the site by any page, not necessarily the newspaper's main page. This occurs because many newspaper readers usually bookmark the pages they frequently visit or know the URL to directly access them.

We use association rules to analyze the pages accessed by the Web site visitors, in order to find possible associations among them and, thereby, allow a better publishing on the site. This publishing can be, for instance, through links, publicity banners, or pop-up windows. Our choice for association rules was mainly due to the fact that the site we worked on does not require client authentication and does not have any mechanism for client identification. Thus, it is hard to keep track of sequences of items from a client that occur with a large interval of time between them.

In order to discover the association rules derived from the given initial transaction set, we used for our example the SWF algorithm with a minimum support of 40% and minimum confidence of 70%. It is important to emphasize that the minimum support and confidence chosen are only an assumption for our example. In the real world, their values have to be determined by analyzing the dataset. Thus, there are no right values, but the ones that better fit in each specific case. Appropriate support and confidence values will balance number of results and processing time.

The first step of the SWF mining process consists in splitting the original database into partitions. The number of partitions is a user-defined parameter where the desired number is one that makes each partition's size approximated to the size of the incremental partitions that might be added later to the database. For this example, we divide the original database into two partitions, each one having 3 transactions. The partition transactions are scanned for the generation of candidate 2-itemsets. This way, for the first partition, composed of the first three transactions {(M, S, H), (M, H), (M, S, T)}, the minimum number of times one itemset has to appear to be considered frequent is: $\lceil 3 * 0.4 \rceil = 2$, where 3 is the number of transactions in the partition and 0.4 is the support established.

Table 4 shows the candidate 2-itemsets derived from the transactions of the first partition, giving also the number of the partition it first appeared (Start Partition) and the number of times it occurred until the current partition processing (Count). Notice that, only the candidate itemsets indicated with the symbol "•" on the right are considered frequent, that is, attended the minimum support. Thus, after each partition processing, only the candidate 2-itemsets considered frequent are stored in what is called the "cumulative filter" and will be carried over to the further phases. The other candidate k-itemsets (k = 3, 4, 5 … n) are found in a later stage and are not required for the processing of each partition.

Table 4: SWF mining of the 1st partition

Candidate 2-itemsets	Start Partition	Count
M, S •	1	2
M, H •	1	2
S, H	1	1
M, T	1	1
S, T	1	1

Afterwards, we consider the occurrence of the next three transactions that compose the second partition: {(M, S, B), (S, H, T), (S, H, B)}. Therewith, the minimum support for the itemsets coming from the previous mining process is based on 6 (3+3) transactions, resulting in a minimum support of: $\lceil (3+3) * 0.4 \rceil = 3$. However, for the new candidate itemsets identified, the filter is: $\lceil 3 * 0.4 \rceil = 2$. In table 5 we can observe that the itemset "M, H", even though it makes part of the frequent candidates set in the prior phase, it does not meet the required support when considering the cumulative data and, therefore, will not make part of the set of frequent candidate itemsets in a further phase. On the other hand, two other itemsets "S, H" and "S, B" meet the required minimum support and will be carried to the forthcoming phases.

After finding all candidate itemsets, the large itemsets, which means the itemsets that are actually frequent, can be found using the scan reduction technique [Park et al. 1997]. Only one scan of the time-variant database is necessary for this.

Table 5: SWF mining after 2nd partition

Candidate 2-itemsets	Start Partition	Count
M, S •	1	3
M, H	1	2
M, B	2	1
S, H •	2	2
S, T	2	1
H, T	2	1
S, B •	2	2
H, B	2	1

Furthermore, we consider an incremental partition to the database. The SWF incremental step consists in removing an old partition from the mined data and adding a new one, updating its final results. We assume the occurrence of the following transactions in the incremental partition:

Incremental Partition = {(M, S, T), (M, T, B), (S, T, B)}

Firstly, we remove the first partition subtracting the final candidate itemsets that have as their start partition the one being removed, by their occurrence count on the first partition, which are stored in the cumulative filter. The start partition number of the remaining items that initially pertained to the first partition is incremented by one, so they now pertain to the following partition after the one being removed.

Table 6: SWF incremental mining – 1st step

Candidate 2-itemsets	Start Partition	Count
M, S	2	1
S, H •	2	2
S, B •	2	2

In our example, the "M, S" itemset was the only one left in the last cumulative filter that started in the first partition being removed. Thus, its start partition now turned to be 2 and its occurrence count was subtracted by its occurrence count in the first partition. Table 6 shows the result of this operation.

We can notice that this "M, S" itemset did not meet the requirements after the subtraction and, therefore, will not be carried to the next step. Later, the incremental transactions are added, following the same operations of any partition in the initial step. Table 7 shows the results after adding the incremental partition.

Table 7: SWF incremental mining – 2nd step

Candidate 2-itemsets	Start Partition	Count
S, H	2	2
S, B •	2	3
M, S	3	1
M, T •	3	2
S, T •	3	2
M, B	3	1
T, B •	3	2

One clear advantage is that it was not necessary to scan again the whole database to generate the candidate itemsets. This way, we can obtain the updated association rules in a much shorter time, when compared to obtaining these updated association rules through a non-incremental mining technique. Now, for finding the actual frequent itemsets, the scan reduction technique is applied. The scan reduction technique matches the itemsets having the same length k that contain at least k-1 items in common, resulting in a candidate of length k+1, which is composed of the k-1 items that matched plus the two items that differed the two itemsets being combined. In addition, the candidate 1-itemsets are all the distinct items of the dataset. The 1-itemsets obviously cannot be association rules but their calculation is particularly important for later calculating some of the pattern interestingness measures. If we get the candidate 2-itemsets from table 7 and apply the scan reduction technique, we will find the following candidates:

{"S", "B", "T", "M", "H", "S, B", "M, T", "S, T", "T, B", "S, B, T", "M, T, B", "M, T, S", "S, M, B, T"}

Then, scanning the database only once to count the itemsets, we will find the following large itemsets meeting the required support: {"S", "B", "T", "M", "S, B", "S, T"}. Therefore, we can see that the candidates "S, B" and "S, T" are truthfully frequent itemsets.

In order to verify if these itemsets are association rules, we now have to check for confidence. Calculating their confidence by the formula explained previously in section 2.2, we found the following results:

- Conf (S → B) = 60%
- Conf (B → S) = 75%
- Conf (S → T) = 60%
- Conf (T → S) = 75%

As we can see, "B → S" and "T → S" meet the minimum required confidence of 70% being thus considered association rules. With this knowledge, we know that when a page related to business or technology is visited, there is a strong probability that this visitor, according to what occurred in the past, will also be interested in the sports sections.

Knowing a set of association rules, we can restructure the Web site having some conviction for publishing the desired advertisements in the most visited pages according to the user behaviour. Thus, one can charge for advertisements according to the frequency of pages visited without having problems with changes on the navigation behaviour of visitors, which may occur due to external factors. For example, if an important sport event is occurring, visitors will likely visit more times the sports section. The incremental algorithm will catch this behaviour change and may dynamically adapt the pages to have the most important advertisements in spots that were not much visited but now turned to be often visited.

In case of incremental mining of sequential patterns with the given algorithms, the process is somehow similar. However, in this case, as its own name indicates, the sequence of the pages visited is considered in pattern extraction processes.

In our real case study, we had an initial dataset of 29,937 cleaned and prepared transaction entries, or sessions, that we considered as our original dataset to be mined. Additionally, average increments of around 250 sessions were inserted, each increment representing nearly 3 hours of navigation storage. It is important to emphasize that this average of 250 sessions are related just to the clean sessions containing two or more concepts within each session. The actual raw log, before the cleansing, session identification and one-concept sessions filtering processes, contains in average over 10,000 entries.

Based on the average increment size containing 3 hours of log, we decided to split the initial dataset into 120 partitions for the first mining iteration. As we earlier explained, although this is not extremely necessary, for a better performance of the SWF algorithm, the number of partitions in which the initial dataset should be divided is a number that makes the resulting partitions have approximately the size of the increments. In our case, 15 days divided into 120 partitions results in each partition having the size close to the average size of a three-hour clean log.

The SWF incremental mining algorithm for the system was implemented in Python [WWW01]. We created object classes for handling each one of the important concepts involved when performing an incremental mining process using the SWF algorithm. In other words, we have objects representing the items, transactions, itemsets, partitions, and the cumulative filter. Besides that, we have a class containing the processes that are used by both, full and incremental mining, from which we extended one class having the specific operations for performing the initial full mining and another one for the incremental process of the SWF algorithm.

Additionally, we created a file handling class, which is responsible for performing all the reading and writing processes. Through this file handling class, the algorithm can be configured to get the information from either a Data Webhouse or a flat file, and it can also write its results in a flat file or in a database.

The class for performing the full mining process requires the following parameters from the user or system process that calls it:

- name of the file containing the initial dataset
- number of partitions this dataset will be split

- minimum support value

- minimum confidence value

In our experiments, we executed the algorithm giving 120 for the number of partitions, 0.01 as the minimum support and 0.2 as the minimum confidence. We came up with these minimum support and confidence values after analyzing a smaller amount of the dataset and verifying that, with this number, the relation of processing time and number of results was satisfactory.

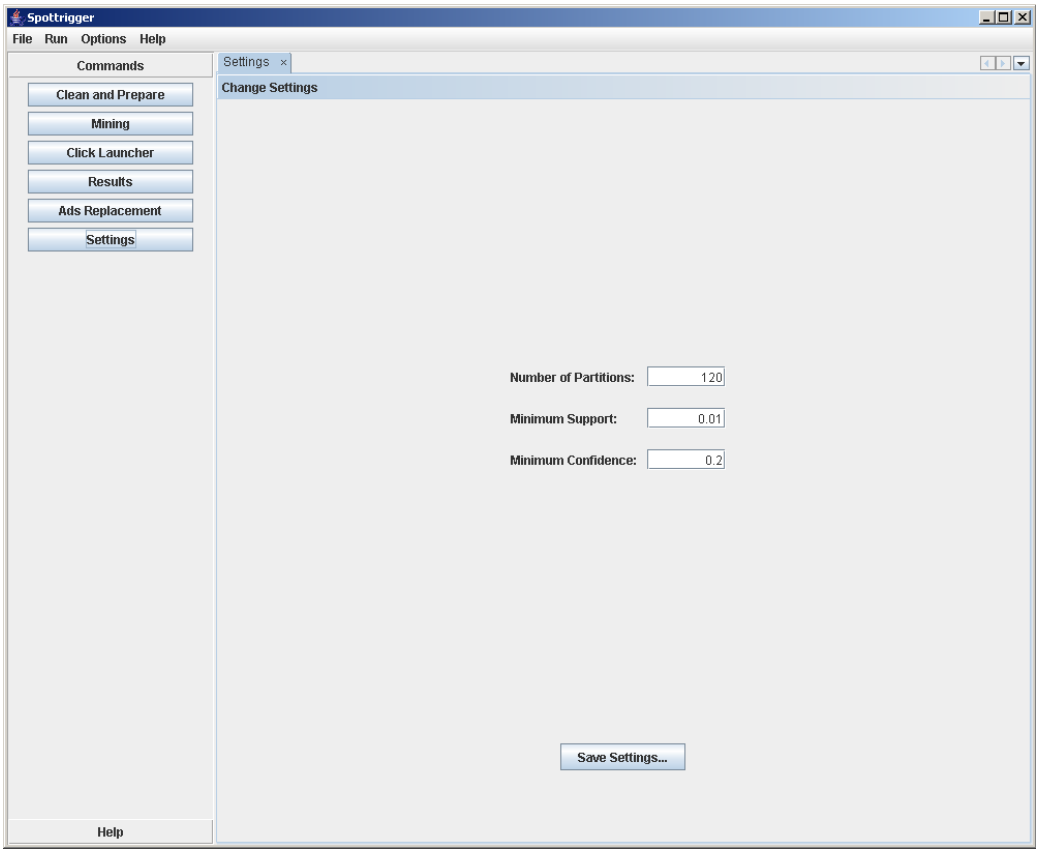


Figure 15: Setting up the mining parameters

For setting up the parameters through the Spottrigger's system interface, all we have to do is click on the "Settings" button on the left-side menu. A settings tab will be shown as demonstrated in figure 15, allowing us to set values for the number of partitions we want to split our initial dataset, minimum support, and minimum confidence. When performing incremental mining, the number of partitions is ignored and the algorithm considers the incremental portion as one partition. By

clicking at the "Save Settings…" button, the configured information is stored on disk for being used in further mining processes.

After setting up the mining parameters, and having the clean original dataset to be mined, the initial full mining process can be finally performed. For performing mining operations in Spottrigger, we shall click at the "Mining" button on the menu, so the mining tab will appear as illustrated in figure 16. Since we have already established the mining parameters, for accomplishing this task the system just asks the location of the dataset to be mined and the mining type, if it is, respectively, a full mining or an incremental one.

Thus, for mining our initial 15 days dataset containing 29,937 sessions with the configuration we mentioned above, we must choose the dataset file, mark the full mining option and click on the "Run" button. The algorithm took 307 seconds (5 minutes and 7 seconds) to perform the full mining, as we can see through the mining statistics window in figure 16.

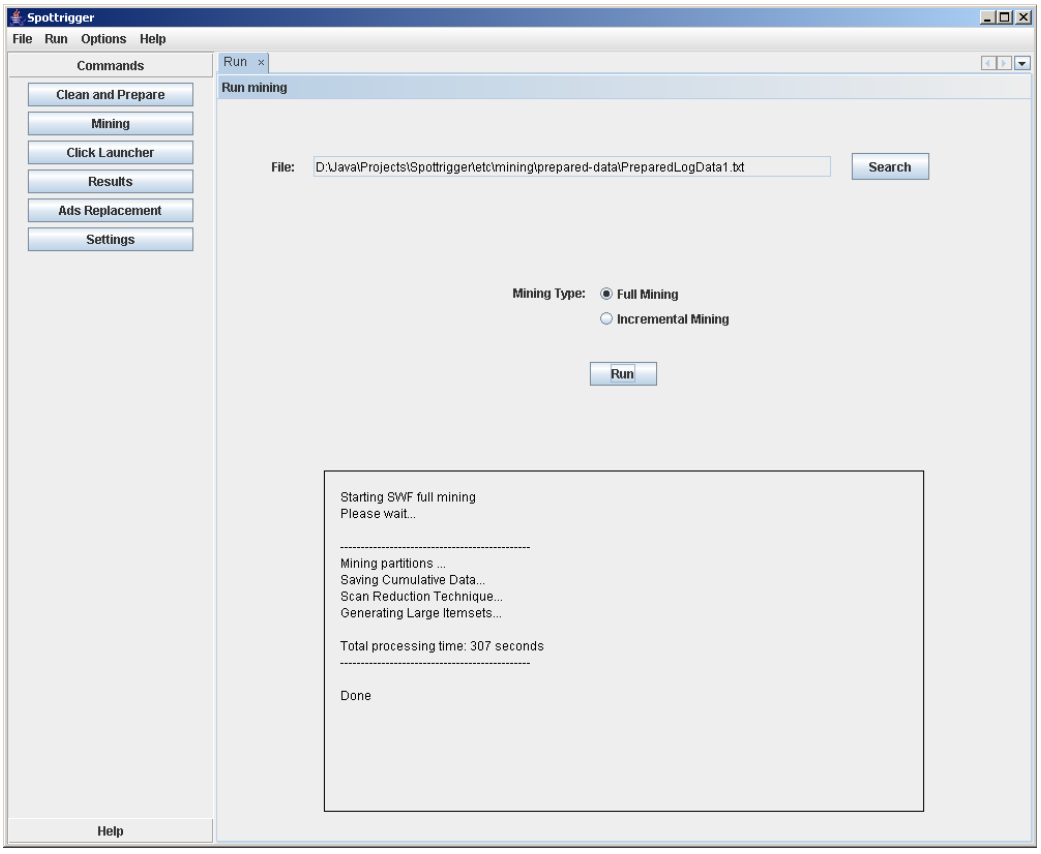
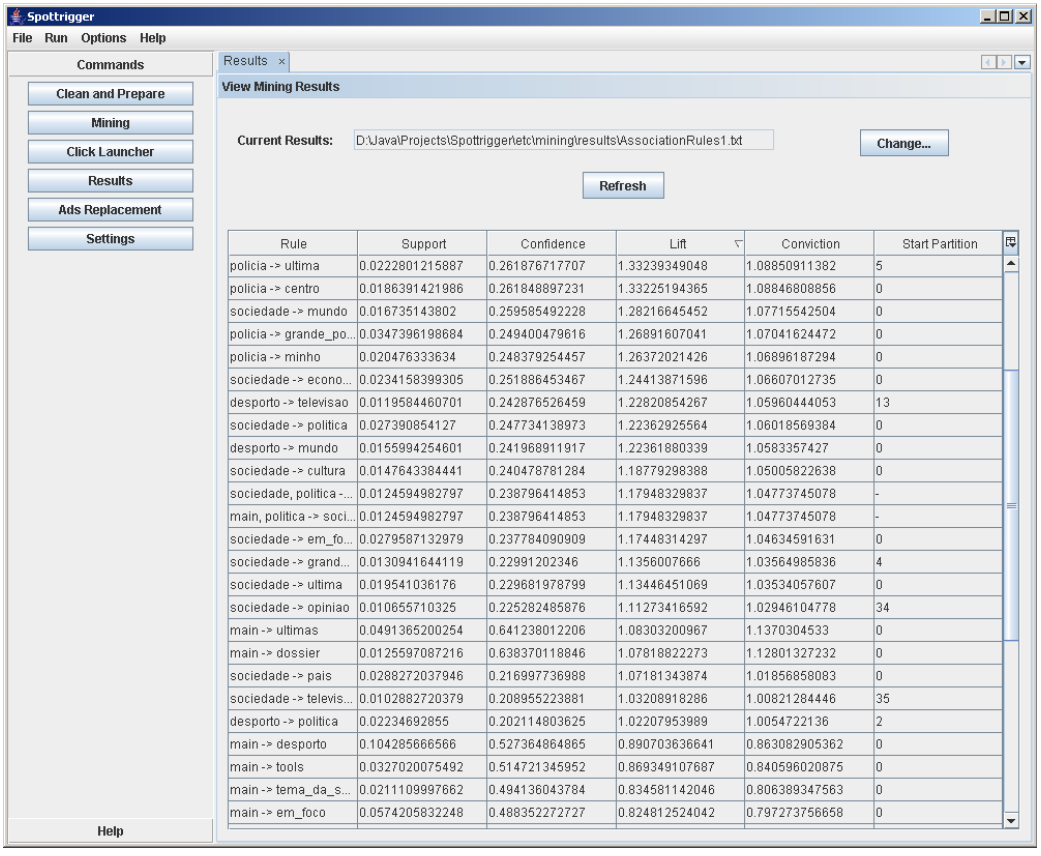


Figure 16: Full mining the initial dataset

For better visualizing the results obtained, we designed in the Spottrigger system a result panel, accessible through the "Results" button on the menu, which contains a table that can be ordered by any of the main measures of pattern interestingness, namely, support, confidence, lift and conviction. These measures were previously explained in section 2.2. For visualizing the results of our initial dataset mining, all we have to do is choose the result file and click on the "Refresh" button as shown in figure 17. As we can see, it is shown the rules, the values for each of the pattern interestingness measures and the start partition where each one of the rules first appeared. We can rearrange any column of the table in ascendant and descendant orders just by clicking at the column label. These operations allow us to better analyze the rules acquired. In figure 17, we descendant ordered the results by the lift measure.



Rule	Support	Confidence	Lift	Conviction	Start Partition
policia -> ultima	0.0222801215887	0.261876717707	1.33239349048	1.08850911382	5
policia -> centro	0.0186391421986	0.261848897231	1.33225194365	1.08846808856	0
sociedade -> mundo	0.016735143802	0.259585492228	1.28216645452	1.07715542504	0
policia -> grande_po...	0.0347396198684	0.249400479616	1.26891607041	1.07041624472	0
policia -> minho	0.020476333634	0.248379254457	1.26372021426	1.06896187294	0
sociedade -> econo...	0.0234158399305	0.251886453467	1.24413871596	1.06607012735	0
desporto -> televisao	0.0119584460701	0.242876526459	1.22820854267	1.05960444053	13
sociedade -> politica	0.027390854127	0.247734138973	1.22362925564	1.06018569384	0
desporto -> mundo	0.0155994254601	0.241968911917	1.22361880339	1.0583357427	0
sociedade -> cultura	0.0147643384441	0.240478781284	1.18779298388	1.05005822638	0
sociedade, politica -...	0.0124594982797	0.238796414853	1.17948329837	1.04773745078	-
main, politica -> soci...	0.0124594982797	0.238796414853	1.17948329837	1.04773745078	-
sociedade -> em_fo...	0.0279587132979	0.237784090909	1.17448314297	1.04634591631	0
sociedade -> grand...	0.0130941644119	0.22991202346	1.1356007666	1.03564985836	4
sociedade -> ultima	0.019541036176	0.229681978799	1.13446451069	1.03534057607	0
sociedade -> opiniao	0.010655710325	0.225282485876	1.11273416592	1.02946104778	34
main -> ultimas	0.0491365200254	0.641238012206	1.08303200967	1.1370304533	0
main -> dossier	0.0125597087216	0.638370118846	1.07818822273	1.12801327232	0
sociedade -> pais	0.0288272037946	0.216997736988	1.07181343874	1.01856858083	0
sociedade -> televis...	0.0102882720379	0.208955223881	1.03208918286	1.00821284446	35
desporto -> politica	0.02234692855	0.202114803625	1.02207953989	1.0054722136	2
main -> desporto	0.104285666566	0.527364864865	0.890703636641	0.863082905362	0
main -> tools	0.0327020075492	0.514721345952	0.869349107687	0.840596020875	0
main -> tema_da_s...	0.0211109997662	0.494136043784	0.834581142046	0.806389347563	0
main -> em_foco	0.0574205832248	0.488352272727	0.824812524042	0.797273756658	0

Figure 17: Initial dataset mining results

After accomplishing the initial mining iteration, we introduced an incremental dataset holding three hours of log. This incremental set initially had a size of 3 MB, composed of 13,620 log entries. After

passing through the preprocessing stage, 274 sessions containing two or more concepts were identified. The incremental process uses the cumulative filter information from the previous mining iteration for processing this incremental dataset and updating the association rules results.

In our system, the incremental mining operation can only be executed if a full mining had been previously performed. The SWF incremental algorithm developed needs the following parameters for being executed: the name of the file containing the incremental dataset, and the minimum support and confidence values. We maintained the same configuration for mining the incremental dataset.

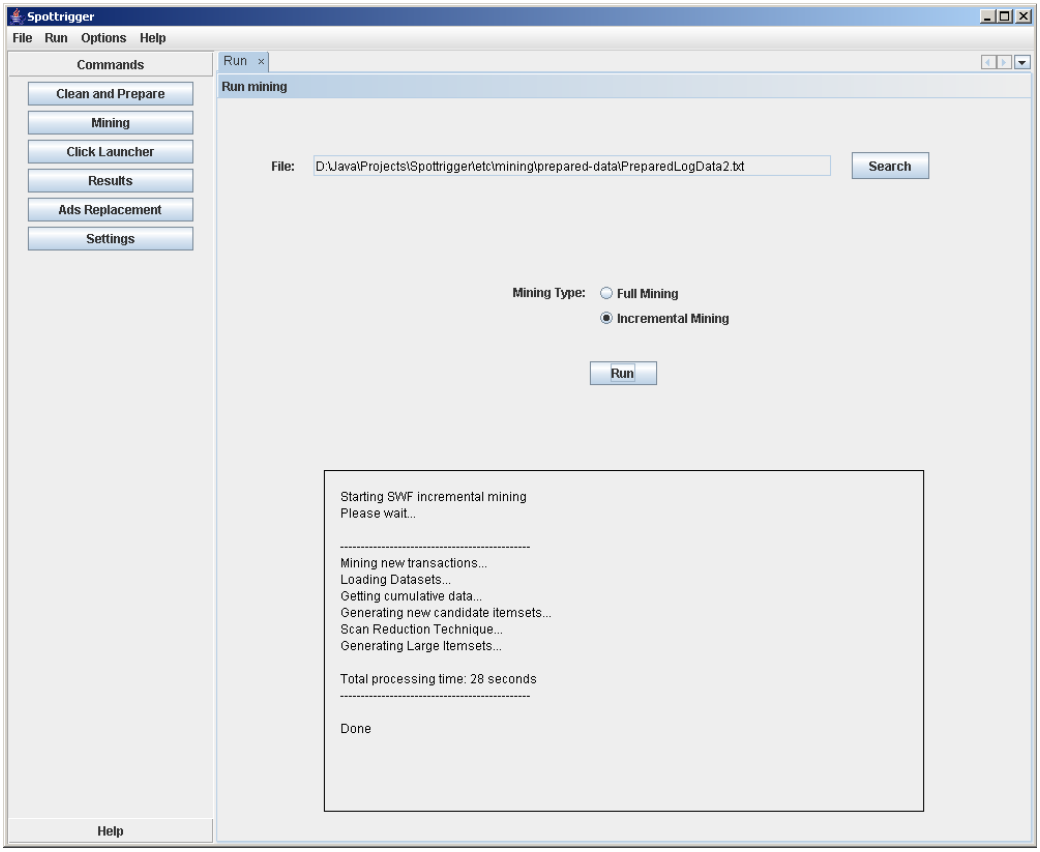
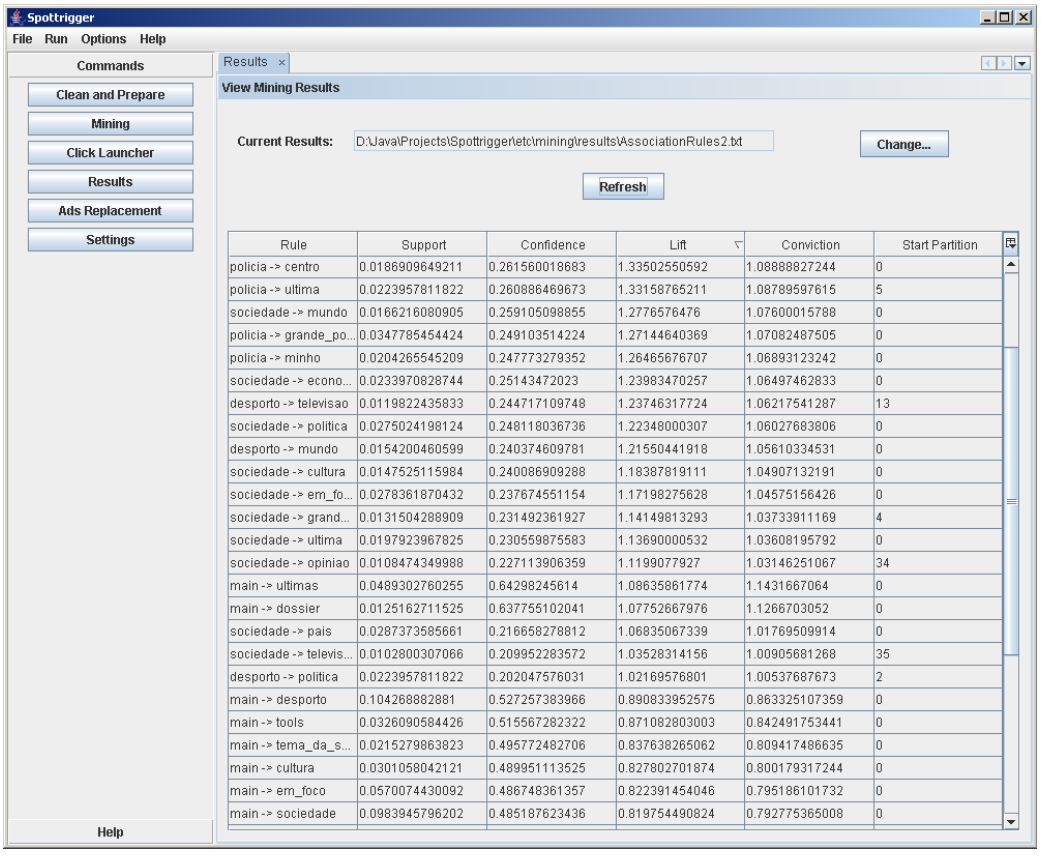


Figure 18: Incremental mining over three hours of log

In order to perform the incremental mining through the Spottrigger system, we shall access the same mining tab of the full mining step through the "Mining" button on the menu. Then, we must choose the incremental dataset file, mark the incremental mining option and finally click at the

"Run" button. Thus, for processing the additional 274 sessions in the incremental step and update the mining results, we reached a time of 28 seconds. Figure 18 above shows the execution and statistics of our incremental mining iteration over three hours of the Web site navigation log.

After each mining process, the patterns database is updated with the new results, which will be used in the forthcoming Spottrigger processes as well as in future mining iterations. Figure 19 below shows the association rules results from our incremental mining process. In order to perceive some differences between the two results, we also ordered this last one by the lift measure. We can see that with the addition of the incremental portion, the itemset "policia → ultima" that at the beginning appeared before "policia → centro", as demonstrated in figure 17, now appears after it, as shown in figure 19.



Rule	Support	Confidence	Lift	Conviction	Start Partition
policia -> centro	0.0186909649211	0.261560018683	1.33502550592	1.08888827244	0
policia -> ultima	0.0223957811822	0.260886469673	1.33158765211	1.08789597615	5
sociedade -> mundo	0.0166216080905	0.259105098855	1.2776576476	1.07600015788	0
policia -> grande_po...	0.0347785454424	0.249103514224	1.27144640369	1.07082487505	0
policia -> minho	0.0204265545209	0.247773279352	1.26465676707	1.06893123242	0
sociedade -> econo...	0.0233970828744	0.25143472023	1.23983470257	1.06497462833	0
desporto -> televisao	0.0119822435833	0.244717109748	1.23746317724	1.06217541287	13
sociedade -> politica	0.0275024198124	0.248118036736	1.22348000307	1.06027683806	0
desporto -> mundo	0.0154200460599	0.240374609781	1.21550441918	1.05610334531	0
sociedade -> cultura	0.0147525115984	0.240086909288	1.18387819111	1.04907132191	0
sociedade -> em_fo...	0.0278361870432	0.237674551154	1.17198275628	1.04575156426	0
sociedade -> grand...	0.0131504288909	0.231492361927	1.14149813293	1.03733911169	4
sociedade -> ultima	0.0197923967825	0.230559875583	1.13690000532	1.03608195792	0
sociedade -> opiniao	0.0108474349988	0.227113906359	1.1199077927	1.03146251067	34
main -> ultimas	0.0489302760255	0.64298245614	1.08635861774	1.1431667064	0
main -> dossier	0.0125162711525	0.637755102041	1.07752667976	1.1266703052	0
sociedade -> pais	0.0287373585661	0.216658278812	1.06835067339	1.01769509914	0
sociedade -> televis...	0.0102800307066	0.209952283572	1.03528314156	1.00905681268	35
desporto -> politica	0.0223957811822	0.202047576031	1.02169576801	1.00537687673	2
main -> desporto	0.104268882881	0.527257383966	0.890833952575	0.863325107359	0
main -> tools	0.0326090584426	0.515567282322	0.871082803003	0.842491753441	0
main -> tema_da_s...	0.0215279863823	0.495772482706	0.837638265062	0.809417486635	0
main -> cultura	0.0301058042121	0.489951113525	0.827802701874	0.800179317244	0
main -> em_foco	0.0570074430092	0.486748361357	0.822391454046	0.795186101732	0
main -> sociedade	0.0983945796202	0.485187623436	0.819754490824	0.792775365008	0

Figure 19: Incremental mining results

This signifies that when dynamically changing advertisements according to the lift measure, before the incremental portion being processed, if a newspaper reader visited the "policia" section, the "ultima" section would have higher level advertisements than the "centro" section. Nevertheless, after the incremental mining process, once a visitor enters the "policia" section, "centro" section will now be restructured with higher level advertisements than "ultima" section. Moreover, we can also notice some other changes, for example, the previously frequent rule "main, politica → sociedade", did not meet the support and confidence requirements, being removed from the set of association rules after the incremental mining processing.

Analyzing the full and incremental processes, we can see differences mainly in terms of processing time. While the full mining took over five minutes to run, the incremental step took only 28 seconds, which is over 10 times faster. Although five minutes is not a long time, this is the time for 15 days of log. Remind that the bigger is the data repository to be mined, the bigger the processing time will be. Besides, if we establish an even lower support value, the processing time will also increase. Therefore, re-running the full mining over the whole dataset would take each time longer, degrading system performance and jeopardizing the system's main goal of restructuring advertisements in a useful time. On the other hand, the incremental processing time will not have a great increase, since the average incremental dataset size should be always low. It is important to emphasize that, as we mentioned in previous chapters, if the incremental dataset size gets too big, the incremental mining may no longer be advantageous.

4.6 Click Launcher

Our initial tests were made by adding manually the incremental portion to perform the mining process. However, today's systems should be able to automatically perform the updates. This way, the company would not need to have a responsible person for this task and could ensure the updates are performed in a regular interval of times, making the system use the newest results for restructuring the Web site.

Aiming on testing our system in a real situation, we developed what we called the Click Launcher. The main idea is to simulate the addition of new data on clickstream as if the visitors' clicks were

really occurring at the present time. Thus, what we do is take an old log of one day, get the time in which the requests occurred in that day and change the date as if they were being requested at the current day. This way, for a simulation of what occurs in a real situation, we automatically launch the clicks when the current time is equal to the time of the request at the modified log.

We summarize the main idea of our click launcher in figure 20. In the example shown, the original log was dated December 30th of 2004. Supposing the current date is June 03rd of 2005, we transformed the records from the original log to have the current date. Then, the transformed dataset shown in the figure is nothing more than the old clickstream with the current date. When the current time is equals to the time of a request in the modified log, the request is launched to a new log, which will be automatically mined every three hours.

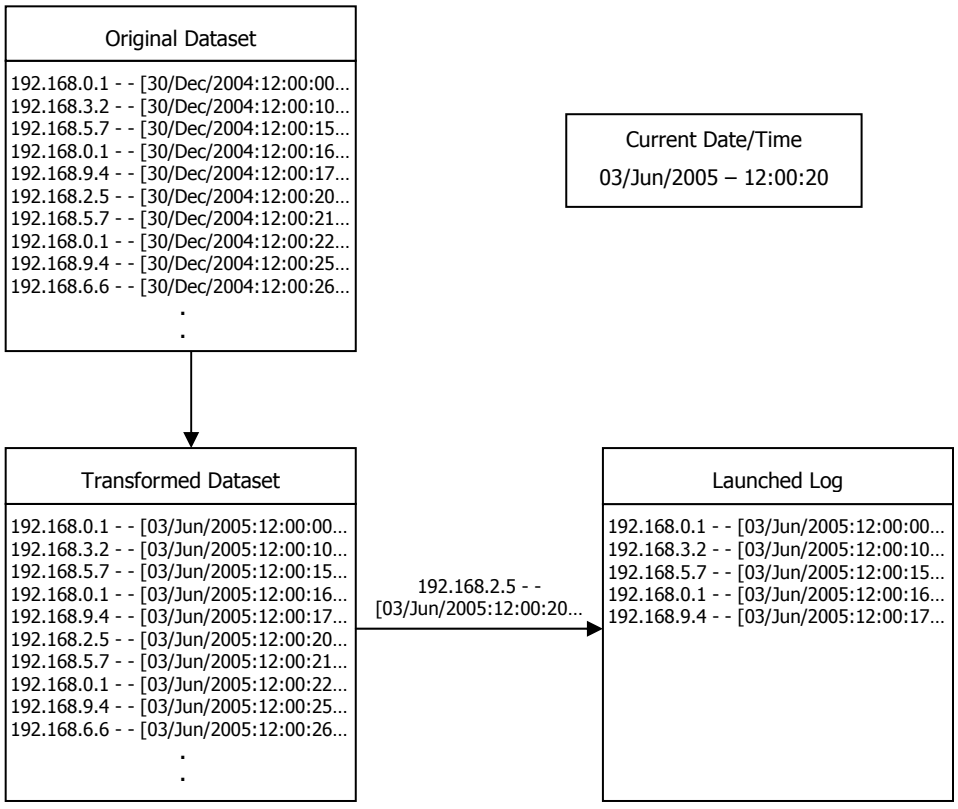


Figure 20: Click Launcher

Thus, the system is capable to identify, every three hours, what data have been already mined and what are the new data that need to be mined. This way, the system gets the new data and

performs the incremental mining. It is important to say that our click launcher can only be started if a full mining has already been performed, because we use an incremental approach over the new data launched.

For performing this real-time simulation through our system, we shall access the click launcher tab through the "Click Launcher" button on the menu. Then, there will be shown fields for choosing the source and destination file, where the destination file is already filled by a system default one. Thus, all we have to do is click on the "Start Launcher" button and the clicks will start being launched. Once started, the "Start Launcher" button will change to "Stop Launcher" for stopping the click launcher. When we start the launcher, the clicks that occurred before the current time are all launched at the moment. The ones that did not occurred yet will be launched at the right time. Then, data preprocessing and incremental mining operations will be performed every three hours until either the user stops the launcher or all the clicks from the source file have been already launched.

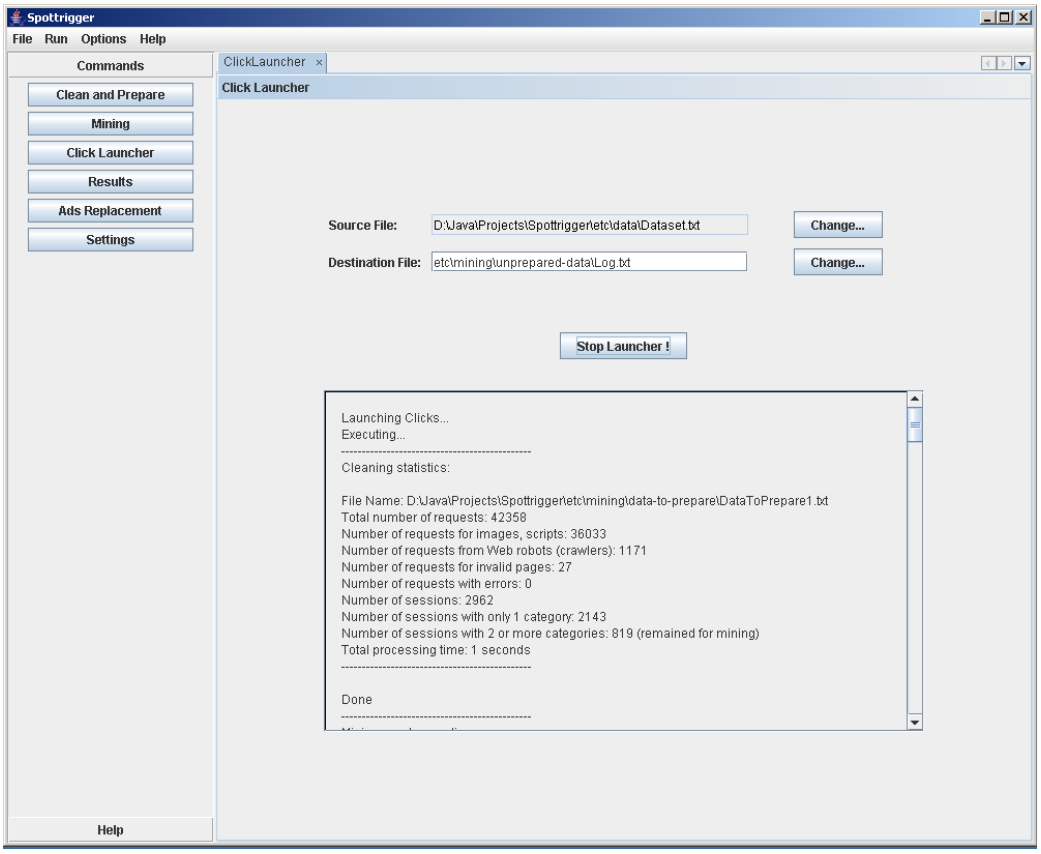


Figure 21: Click launcher execution

Figure 21 shows the click launcher after the execution of an incremental mining operation. After each mining iteration, the statistical logs from the preprocessing and incremental mining phases are shown at the log window. The results from the incremental mining iterations can be viewed through the results panel as we explained before. These results will be used to restructure the Web site with advertisements according to their level classification.

4.7 Advertisement Replacement on Page Spots

As formerly explained, the Spottrigger system has an advertisements database that holds the advertisements classified in levels by their price and importance. The classification in levels is important because according to the season of the year or with the occurrence of an important event, the range applied for the classification of the advertisements in level may vary. This occurs due to the fact that in these occasions, the search for advertising in the newspaper pages may increase or decrease, possibly affecting in the advertising price.

The constantly updated patterns database along with the advertisements database are the key sources of information for the decision process that will resolve what advertisement shall be shot in the spots of a page that a given visitor will probably access and thereby perform the page spots update. For accomplishing this task, we developed an advertisement replacement algorithm, which is the other key point of our Spottrigger system. Once having the frequent patterns and ensuring that an incremental mining process regularly updates them, we can guarantee that most people, regardless of their navigation behaviour on the site, will see a given advertisement.

Our advertisement replacement algorithm extracts the classified data from the advertisements database and connects them with the results stored in the patterns database. Furthermore, when a visitor is navigating through the Web page, his accessed pages are also sent to this algorithm. Matching the results obtained by processing the data from patterns and advertisements databases with the current reader navigation records, the algorithm analyzes what will be the next page sections probably accessed by the reader and chooses what advertisements shall be put in these page spots, restructuring the target Web pages. For clarifying the idea of our algorithm, we will give a practical example (figure 22).

Suppose we have two advertisements, ADV1 and ADV2. ADV1 is more expensive and therefore is classified as a level 1 advertisement. ADV2, on the other hand, belongs to level 2. Now, let us assume that "S -> B, T" is an association rule pertaining to the patterns database, meaning that a reader that accesses the sports section has a high probability of also accessing the business and technology sections. In this example, when the reader accesses the sports section, the advertisement spots in the business and technology sections will show ADV1, while the ADV2 will be shown if the reader accesses the other sections. We can have a high confidence in this operation because according to the association rule found, the latter will occur fewer times than the former, what corresponds to the expected results since we want the higher level advertisement to be seen more times than the lower one.

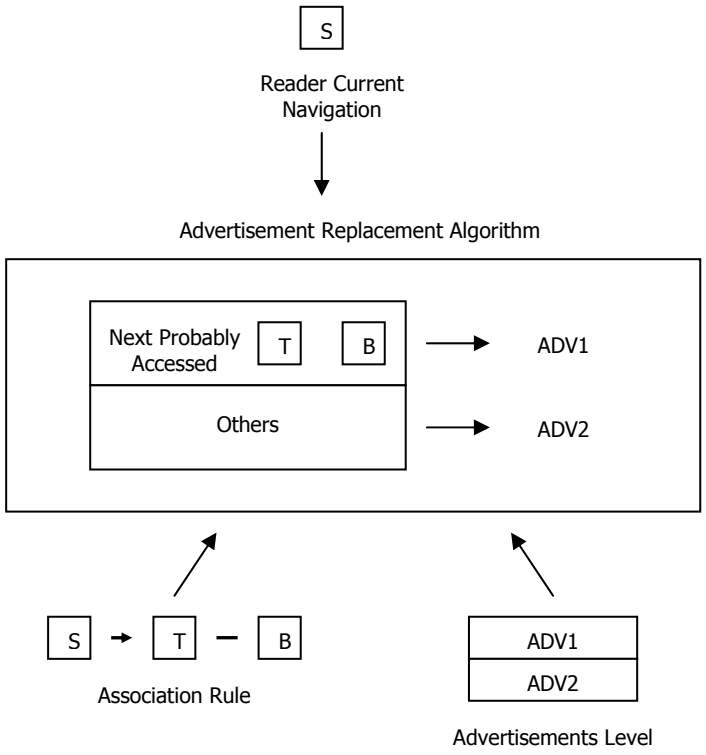


Figure 22: Advertisement Replacement Example

Once having the pages with the desired advertisements in spots, they can be finally updated in the Web server for the view of the specific visitor who had his navigation behaviour analyzed by the Spottrigger system. In this case, the Web server has to keep track of each reader's navigation records to send them to the advertisement replacement algorithm. One way the server can keep

track of the visitors' navigation steps is by storing separately the accessed pages by their IP-address, along with a timeout for the sessions. This way, the correct restructured Web pages can be sent to the reader.

Through the Spottrigger's interface, we implemented a way to simulate the navigation of a given visitor, and by choosing the main pattern interestingness measure from what the advertisement replacement will be based, we show the page concept rank for restructuring the pages of these concepts with the advertisements according to their level. This module is accessible through the "Ads Replacement" button on the menu. In order to exemplify, we are going to demonstrate a practical example through our system, based on the result obtained after our incremental execution previously explained and shown in figure 19.

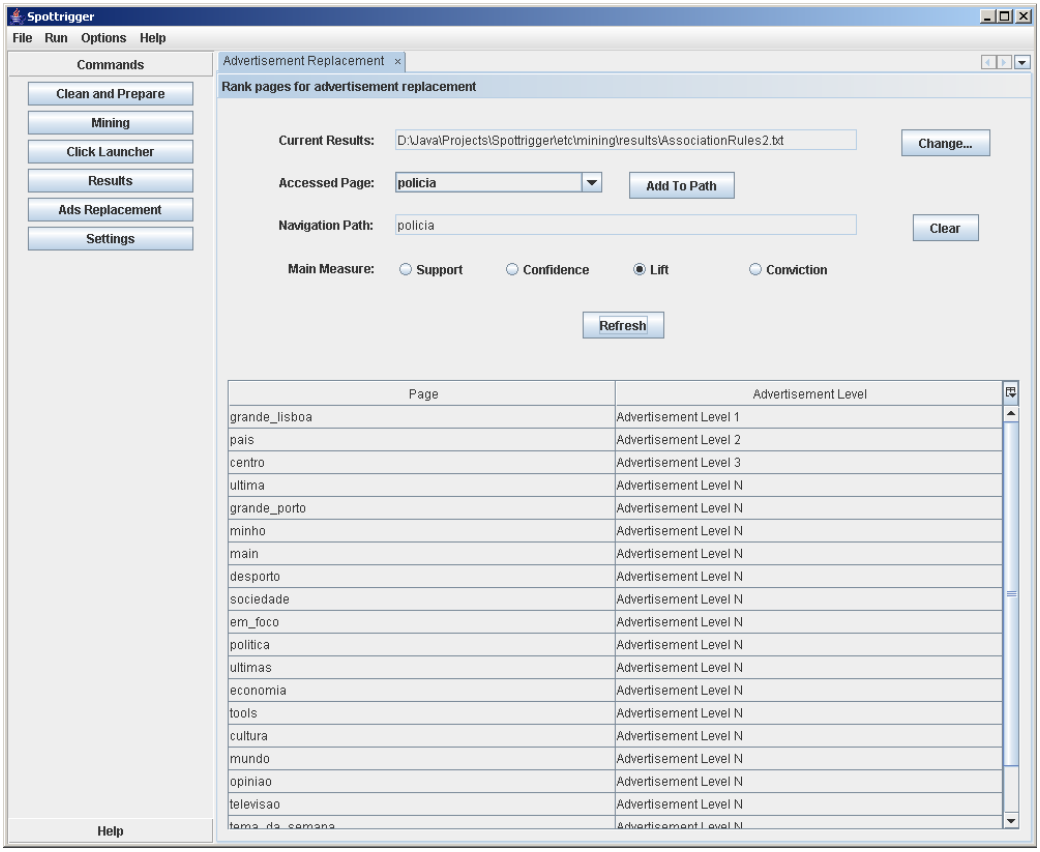


Figure 23: First advertisement replacement

Figure 23 shows how we simulate the advertisement replacement through our system. In order to simulate a real user navigation, we firstly choose a page concept in the available accessed page

concept combo. When we click on the "Add to Path" button, the page concept selected is inserted in the concept path field. Clicking now at the "Refresh" button, the advertisement replacement algorithm will make the relationship between the next probably accessed pages and the advertisements by analyzing the user navigation path and the association rules found.

Suppose we want to restructure the pages according to the lift measure. If a given visitor accesses the "policia" section, we can see through figure 23 that the section having a greater lift, which will receive the higher level advertisements, when this occurs, is the "grande_lisboa" section. Furthermore, the level 2 advertisements will be shown in "pais", level 3 in "centro" and so forth.

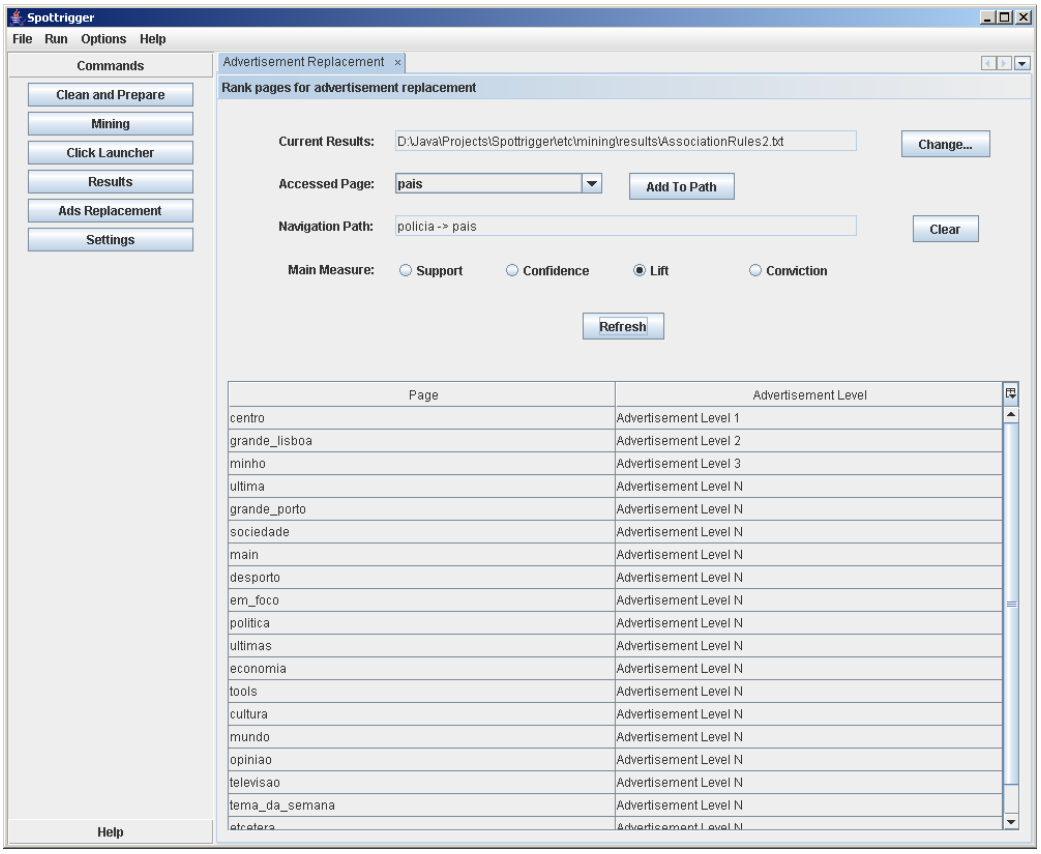


Figure 24: Second advertisement replacement

Now, let us assume that this user accessed the "pais" section, signifying that he will see the level 2 advertisements. At this point, the advertisement replacement algorithm will run again to see if the pages need to be once more restructured.

Figure 24 shows the results after the second advertisement replacement. We can see that the "grande_lisboa" section, which used to have level 1 advertisements, now has level 2 advertisements. According to the patterns matching this new navigation path, the "centro" section that previously had level 3 advertisements, now has level 1.

```
Initiate system operation;
establish FTP connection with Web Server;
terminate = false;

while not (terminate) do
       user authentication(userid, password);
       extract clickstream file not processed from Web Server;
       store clickstream file on system data staging area;
       clean new clickstream data;
       catalog cleaning errors;
       store cleaning data on system Webhousing;
       apply incremental data mining process;
       analyze mining results;
       identify target Web pages based on mining results and spots offering;
       restructure target Web pages;
       update Web site with restructured target Web pages;
       make garbage collection;
       continue or not (terminate)
end-do;

close FTP connection;
terminate system operation.
```

Figure 25: Spottrigger's Life Cycle Pseudo-Code

Through this short example, we could see how it is possible to restructure the advertisements on the pages. Guaranteeing that the patterns are regularly updated, these results will represent the actual navigation behaviour of the users, or have a good approximation of them. Thus, it is possible to have some confidence on this page restructuring method, which may be a valuable quality difference when merchandising the advertisement spots. After the page restructure, the entire process restarts.

When the entire data cycle is finished, with the visitor leaving the newspaper site, it is important to make a garbage collection because variables, temporary tables and other objects are used throughout the processes and do not need to be kept in memory anymore. Then, the FTP connection with the Web server is finally closed before the system terminates its operation. We summarize the whole Spottrigger Life Cycle in the pseudo-code shown in figure 25.

4.8 Implementation Resources

Our experiments were all performed on a notebook computer having an Intel Pentium M Centrino 1.6 MHz processor, with 1GB of RAM. They were executed on top of Microsoft Windows 2000 Professional.

We implemented the cleaning processes and the SWF incremental algorithm using Python [WWW01], version 2.3. Python has the advantage of being a fast scripting language and works well in an object-oriented manner. Besides, it is very concise and productive, being easy to learn, understand and, further, maintain.

For implementing the interface, we used the Java [WWW05] programming language and the Netbeans [WWW06] tool. The Java language has a broad *Application Program Interface* (API) for designing systems interfaces. Moreover, Netbeans provides a friendly environment for working with Java APIs.

The Spottrigger's Java made interface links to the python algorithms, passing the mining parameters established by the user. After processing, the results produced by the python algorithms which are stored in either a file or a database can be visualized through the interface. In addition, the processes log produced by the algorithms can also be visualized.

The performance results presented were derived from a clickstream of an existing newspaper company. For performing our experiments, we were given the clickstream data derived from one month of visitors' navigation in the newspaper Web site. We chose to use 15 days of log as being our initial dataset and so we incrementally added the remaining data in portions comprising 3

hours of users' navigation. Thus, we performed our tests in different ways, such as simulating real time accesses through the click launcher and also submitting incremental portions in a manual way. For performing the incremental mining in a manual way, we separated files "by hand" in order to make each one of them contain 3 hours of log, and thus designated the incremental file to the algorithm. On the other hand, when performing mining operations through the click launcher, we implemented an automatic way to break the file every 3 hours and submit this separated incremental portion to the incremental mining algorithm.

Chapter 5

Conclusions and Future Work

5.1 Final Remarks

The high-speed e-commerce race that we observe today among companies has made them realize the importance of exploring and analyzing data that they have stored in their operational systems. When properly studied through mining techniques, this data may bring up new knowledge that can be extremely valuable to determine the future behaviour of companies' customers.

When discovering clients' navigation behaviour in a Web site, we are actually looking towards mining clickstreams, which is comprised in the Web usage mining area. We saw that this area has attracted during the last few years much attention from research and e-business professionals, offering many benefits to an e-commerce Web site, namely: target customers based on usage behaviour or profiling (personalization); dynamically restructuring Web sites based on user page access patterns (site modification); answer stakeholders business questions which may help in the decision making processes for enhancing the service quality and delivery to the end users – cross-selling and up-selling (business intelligence); analyze the navigational strategy of users browsing the company's Web site for predicting their future behaviour while they interact with the Web site (usage characterization); or improve system performance, mainly based on the Web traffic analysis (system improvement).

Initially, the proposed mining algorithms were found to be inefficient when dealing with dynamic databases, because they needed to be re-executed over the entire database, each time appeared new update occurrences, in order to update results. In e-commerce sites, databases are constantly changing as time passes. Every time a product is sold, a client updates its shopping cart, or even a visitor accesses a page through a simple click, databases that support these tasks are, obviously, updated. Thus, the incremental mining techniques emerged for solving some of the needs of mining data on dynamic databases. On this field, some incremental algorithms, mainly for mining association rules [Chen et al. 2001] [Veloso et al. 2002a] and sequential patterns [Parthasarathy et al. 1999] [Masseglia et al. 2000] were proposed, having good performance when compared to previously developed ones. Taking into account the increasing size of databases due to the high and increasing dataflow, which is nowadays verified inside the most different Web sites, mining data in a suitable (or useful) time, so the decisions can be taken fast, is certainly an interesting challenge. This difficult and stimulating task is the main goal of the new incremental mining approach.

In this work, we explained and analyzed some of the most recent incremental mining algorithms. We compared these techniques and reached a conclusion that the choice of one technique to be applied depends on some characteristics like the database size, the available hardware, and eventual necessities of changes in the mining requirements. For each specific case, there might be one technique that fits better than the others. Thus, what we have to do is balance characteristics such as flexibility and fastness when choosing the algorithm for a specific application case.

For our study, we were given real data derived from an online newspaper site usage. Our main challenge was regarded to how an online newspaper company may commercialize its Web site advertisements spots. Organizations willing to publish their advertisements generally look for reaching the highest number possible of readers. However, readers' navigation behaviour may change very fast, so that pages that used to be frequently accessed might not be accessed anymore, and non-accessed pages might turn out to be frequently visited. In this way, an organization might have paid a lot of money for publishing on frequently accessed pages, thinking its advertisements would be highly seen, but suddenly realize that its investments went on the wrong way because readers do not access those pages anymore.

We demonstrated the possibility of dynamically changing the advertisements on newspaper's online pages, so one can assure that the most important advertisements will be the most exhibited to the readers regardless of their navigation behaviour changes. For that, as the visitor navigates, we incrementally construct a page ranking by the most visited pages according to his profile, acquired through the mining results, and then place the desired advertisements on pages he will probably visit. Thus, it is possible to merchandise the publicity "spots" considering how frequently the announcement will be seen.

For accomplishing this task, we built Spottrigger, a system capable of extracting the data stored in the Web server, cleaning, preparing and mining these data, and restructuring the newspaper pages according to the mining results along with a set of advertisements classified in levels of importance. Thus, the higher is the level of the advertisement, the more it will be exhibited. Therefore, through Spottrigger, we devised a life-cycle model designated for performing all the operations since the acquisition of the raw clickstream data from Web servers until the choice of advertisements level to be on the pages. According to a given reader's current navigation, it makes possible to restructure the Web site taking into account the newspaper visitors past navigation.

Spottrigger uses *Sliding Window Filtering* (SWF), which is an incremental association rules mining algorithm, for restructuring the pages of the online newspaper according to the reader's profile. The SWF was chosen because of its fastness, simplicity and adaptability to short term incremental mining operations. In other words, we wanted to mine in a specific window of time, by adding new transactions and removing old ones whenever each new incremental dataset arrived. For example, one can choose to have the navigation patterns of the last month with increments of one day. Thus, on every incremental process, one day of data would be removed from the considered dataset (the earliest one) and a new one-day increment would be added, with the final results being updated.

Our initial goal was to effectively analyze the possibility of restructuring the Web site in a useful time, that is, as fast as possible, to adapt the site to the user's profile as he navigates through the pages. In our practical case, we added new increments containing 3 hours of the newspaper Web site usage. This increment was added both in a manual way and simulating a real situation through our click launcher – a process that launches clicks from an old log, simulating the navigation of the newspapers readers as if they were occurring at the present time. We came up with this 3 hours

interval between the mining processes by analyzing the available data and realizing that if we established a shorter interval of time, there would not be a considerable amount of data for changing much the results, so the high overhead for often getting these data would not compensate. By refreshing the results at every 3 hours, we are guaranteeing that the current results reflect a good approximation of the actual readers' navigation behaviour, balancing with the memory and processing costs.

For testing, we firstly considered 15 days of log as our initial dataset. We achieved a time of over 5 minutes for the full mining process, considering a value of 0.01 for minimum support, and 0.2 for minimum confidence. Having lower support and confidence values or working with bigger datasets than the one we had available would obviously increase the processing time. Using a non-incremental approach, the processing time always increase over the time and soon would not allow the mining processes to be performed within a short interval of time. This would lead to a situation where by the time the results are finally obtained, they do not represent the current visitors' navigation behaviour anymore, being already outdated. On the other hand, we added 3 hours of log through an incremental approach, reaching a time of 28 seconds for this processing. The time for incremental mining processing will not grow much if the increments size keeps approximately the same. This way, we can assure that the results are fast updated, allowing the replacement of advertisements according to the recent readers' navigation behaviour.

For performing the advertisement replacement, we also developed, in the Spottrigger system, an algorithm that distributes the advertisements classified in levels in the newspaper pages according to the association rules acquired through the incremental mining algorithm. This distribution may be performed taking into consideration any of the most used pattern interestingness measures, like support, confidence, lift, or conviction.

One of the disadvantages of the algorithm we used is some lack of flexibility when changing mining parameters. If we change the support value for a lower value, the results comprehended within the older value and the new lower value will only be considered in the incremental data. With the addition of increments the results will gradually adapt to the new support, because each time we add a new increment we also remove an old one. However, it is not possible to instantly have the results from a lower support without performing a full mining operation in the database.

Spottrigger was designed to support any incremental mining algorithm though, meaning that we can develop other algorithms capable of acquiring the new increments, gathering with the already existing patterns, and updating the results. However, some algorithms might need to store some information that are not being stored and a full mining would be necessary in order to have this information. We resume Spottrigger main characteristics in table 8.

Through our experiments, we could realize the importance and application of the incremental mining algorithms as well as the subsisting problems. We could see the scalability of the incremental approach through the high savings in terms of processing time. Thus, the Spottrigger system could allow an effective Web site restructuring, in a useful time and in agreement with readers' profile, that is, reflecting the latest readers' navigation behaviour. This way, a newspaper company may merchandise its advertisements spots guaranteeing that most of its readers will visualize the published advertisement.

Table 8: Spottrigger Characteristics

Characteristic	Description
Original Dataset Format	• ECLF
Preprocessing treatments	• Filter requests for images, scripts and other unwanted ones • Remove logs derived from crawlers navigation • Eliminate logs with errors • Extract page concept from remaining requests, associating it with an ID • Identify the sessions
Mining Type Chosen	• Association Rules
Incremental Algorithm Used	• Sliding-Window Filtering
Results Refreshment Interval	• 3 hours

5.2 Future Work

In this work, we could successfully indicate advertisements for the newspaper pages. Nevertheless, we only implemented one of the many existing incremental mining algorithms. As we previously explained, there is no best algorithm, but simply one that may fit better than the others according to each specific case. Besides that, our system may also be improved in several ways. Based on the experience acquired during this work, we are able to say that the system must be improved and refined. Thus, some additional work must be done in a near future, namely to:

- Add other alternative incremental association rules mining algorithms on the system, allowing the comparison of results, improving performance and flexibility among the several mining possibilities.

- Develop an incremental algorithm for sequential patterns mining, which may be used for the same purpose as the association rules one, giving particular results when considering the order in which the pages are visited and, hence, its costs and benefits when compared to the association rules ones could be also analyzed.

- Improve and incorporate new treatments on the preprocessing phase. Since it was not our main focus, a lot of work can be done aiming on refining the data to achieve a high level on data quality, in order to ensure that results can be highly trustable.

- Allow page restructuring by other measures and with different calculations. We only indicated advertisements given one of the main interestingness measures (besides the required minimum support and confidence that each itemset has to attend to be considered as an association rule). Some other calculations, for instance, based on giving a weight to each measure, could be also applied in order to come up with the page rank for the advertisement replacement.

These are just some of the possible proposals. Depending on where the system will be applied, many other modifications can be necessary such as acquiring data from XML texts or writing results in specific formats to be used by other existing decision support systems. With the high research progress in this area, soon there will be even more improvements that can be possibly

incorporated. However, it is important to analyze each case for the organization's needs and draw the best possible strategy to attend these requirements. It might not be necessary to have the most complete and modern system, but one that attend to the basic knowledge discovering requirements of the organization, providing the desired results.

Bibliography

[Abraham 2003] A. Abraham: "Business Intelligence from Web Usage Mining". Journal of Information and Knowledge Management (JIKM), World Scientific Publishing Co., Singapore, Vol. 2, No. 4, pages 375-390, 2003.

[Agrawal et al. 1993] R. Agrawal, T. Imielinski, and A. Swami: "Mining association rules between sets of items in large databases". In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 207-216, Washington D.C., May 1993.

[Agrawal & Srikant 1994] R. Agrawal and R. Srikant: "Fast algorithms for mining association rules". In Proceedings of the 20th Very Large Data Base Conference (VLDB'94), Santiago de Chile, Chile, pages 487-499, 1994.

[Agrawal & Srikant 1995] R. Agrawal and R. Srikant: "Mining Sequential Patterns", In Proceedings of the 11th International Conference on Data Engineering (ICDE'95), Taipei, Taiwan, March 1995.

[Alves et al. 2004] R. Alves, O. Belo, F. Cavalcanti and P. Ferreira: "Clickstreams – The Basis to Establish User Navigation Patterns on Web Sites". In Proceedings of the Fifth International Conference On Data Mining, Text Mining and their Business Applications, Malaga, Spain, September 15-17, 2004.

[Baglioni et al. 2003] M. Baglioni, U. Ferrara, A. Romei, S. Ruggieri and F. Turini: "Preprocessing and Mining Web Log Data for Web Personalization". In Proceedings of the 8th Italian Conference on Artificial Intelligence: 237-249. Vol. 2829 of LNCS, September 2003.

[Banerjee & Ghosh 2001] A. Banerjee and J. Ghosh: "Clickstream Clustering using Weighted Longest Common Subsequences". In Proceedings of the Workshop on Web Mining in First International SIAM Conference on Data Mining, Chicago, 33-40, April 2001.

[Baraglia & Palmerini 2002] R. Baraglia and P. Palmerini: "Suggest: A web usage mining system". In Proceedings of the IEEE International Conference on Information Technology: Coding and Computing, 2002.

[Bosworth & Schiffman 2001] J. Bosworth and S. Schiffman: "The Clickstream Data Wharehouse". In Proceedings of the Oracle Development Tools User Group (ODTUG) Conference, San Diego, June 23 – 28, 2001.

[Brin et al. 1997] S. Brin, R. Motwani, J. D. Ullman and S. Tsur: "Dynamic itemset counting and implication rules for market basket data". In SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, pages 255-264, Tucson, Arizona, USA, May 1997.

[Buchner et al. 1999] A. G. Buchner, M. Baumgarten, S. S. Anand, M. D. Mulvenna, and J. G. Hughes: "Navigation pattern discovery from internet data". In WEBKDD, San Diego, CA, 1999.

[Catledge & Pitkow 1995] L. Catledge and J. Pitkow: "Characterizing browsing behaviors on the World Wide Web". Computer Networks and ISDN Systems 27(6), pages 1065-1073, 1995.

[Cavalcanti & Belo 2005] F. Cavalcanti and O. Belo: "Improving effectiveness of Web sites using incremental data mining over clickstreams". In Proceedings of the Sixth International Conference On Data Mining, Text Mining and their Business Applications, Skiathos, Greece, May 25-27, 2005.

[Chang & Yang 2003] C.-H. Chang and S.-H. Yang: "Enhancing SWF for Incremental Association Mining by Itemset Maintenance". In Proceedings of the seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining, Korea, 2003.

[Chen et al. 2001] M.-S. Chen, C.-H. Lee and C.-R. Lin: "Sliding-window Filtering: An efficient algorithm for incremental mining". In International Conference on Information and Knowledge Management (CIKM01), pages 263-270, November 2001.

[Cheung et al. 1996] D. Cheung, J. Han, V. Ng, and C. Y. Wong: "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique". In Proceedings of the 12th International Conference On Data Engineering, February 1996.

[Cheung et al. 1997] D. Cheung, S. Lee, and B. Kao: "A general incremental technique for maintaining discovered association rules". In Proceedings of the Fifth International Conference On Database Systems For Advanced Applications, pp. 185-194, Melbourne, Australia, March 1997.

[Cooley et al. 1997] R. Cooley, B. Mobasher and J. Srivastava: "Web Mining: Information and Pattern Discovery on the World Wide Web". In Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97), 1997.

[Cooley et al. 1999] R. Cooley, B. Mobasher, and J. Srivastava: "Data preparation for mining World Wide Web browsing patterns". Journal of Knowledge and Information Systems, 1(1), February 1999.

[Eirinaki & Vazirgiannis 2003] M. Eirinaki and M. Vazirgiannis: "Web Mining for Web Personalization". ACM Transactions on the Internet Technology (TOIT), volume 3, issue 1, pages 1-27, February 2003.

[Fayyad et al. 1996] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth: "From Data Mining to Knowledge Discovery: An Overview". In Advances in Knowledge Discovery and Data Mining, edited U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI Press/MIT Press, pages 1-34, 1996.

[Goethals & Bussche 2000] B. Goethals and J. Bussche: "Interactive Constrained Association Rule Mining". In Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery, London-Greenwich, United Kingdom, September 4-6, 2000.

[Gouda & Zaki 2001] K. Gouda and M. Zaki: "Efficiently Mining Maximal Frequent Itemsets". In proceedings of the IEEE International Conference on Data Mining, San Jose, USA, pages 163-170, November 2001.

[Hallam-Baker & Behlendorf 1996] P. M. Hallam-Baker and B. Behlendorf: "Extended Log File Format". World Wide Web Consortium [Working Draft Document], http://www.w3.org/pub/WWW/TR/WD-logfile.html, March 1996.

[Han et al. 2000] J. Han, J. Pei and Y. Yin: "Mining frequent patterns without candidate generation". In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, Proceeding of the 2000 ACM SIGMOD International Conference on Management of Data, volume 29:2 of SIGMOD Record, pages 1-12, ACM Press, 2000.

[Han & Kamber 2001] J. Han and M. Kamber: "Data Mining: Concepts and Techniques". Morgan Kaufmann Publishers, 2001.

[Hochheiser & Schneiderman 1999] H. Hochheiser and B. Schneiderman: "Understanding patterns of user visits to web sites: Interactive starfield visualizations of www log data". Technical Report CS-TR-3989, University of Maryland, 1999.

[Kimball & Merz 2000] R. Kimball and R. Merz: "The Data Webhouse Toolkit: Building the Web-Enabled Data Warehouse". John Wiley & Sons, Chichester, 2000.

[Kimball & Ross 2002] R. Kimball and M. Ross: "The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling". 2nd Edition, John Wiley & Sons, 2002.

[Lee & Cheung 1997] S. Lee and D. Cheung: "Maintenance of discovered association rules: When to update?". In Research Issues on Data Mining and Knowledge Discovery, 1997.

[Lee et al. 2001] C.-H. Lee, C.-R. Lin and M.-S. Chen: "On Mining Temporal Association Rules in a Publication Database". In Proceedings of the 1st IEEE International Conference on Data Mining (ICDM), 2001.

[Liu & Ma 2001] B. Liu and Y. Ma. "Analyzing the Interestingness of Association Rules from the Temporal Dimension". In 1st IEEE International Conference on Data Mining, November, 2001.

[Luotonen 1995] A. Luotonen, "The Common Logfile Format", World Wide Web Consortium, http://www.w3.org/pub/WWW/Daemon/User/Config/Logging.html, 1995.

[Masseglia et al. 2000] F. Masseglia, P. Poncelet and M. Teisseire: "Incremental Mining of Sequential Patterns in Large Databases". Actes des 16ièmes Journées Bases de Données Avancées (BDA'00), Blois, France, October 2000.

[Mobasher et al. 1996] B. Mobasher, N. Jain, E. Han and J. Srivastava: "Web mining: Pattern discovery from world wide web transactions". Technical Report, TR 96-050, University of Minnesota, Department of Computer Science, Minneapolis, 1996.

[Mobasher et al. 2000] B. Mobasher, R. Cooley and J. Srivastava: "Automatic personalization based on web usage mining". Commun. ACM, 43, pages 142-151, August 2000.

[Nakov 2000] P. Nakov, "Web personalization using Extended Boolean Operations with Latent Semantic Indexing", In Proceedings of the 9th International Conference on Artificial Intelligence: Methodology, Systems, and Applications, Springer-Verlag, London, UK, pages 189-198, 2000.

[Park et al. 1997] J.-S. Park, M.-S. Chen and P. S. Yu, "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules", IEEE Transactions on Knowledge and Data Engineering, October 1997, 813 - 825.

[Parthasarathy et al. 1999] S. Parthasarathy, M.J. Zaki, M. Ogihara and S. Dwarkadas. "Incremental and interactive sequence mining". ACM International Conference on Information and Knowedge Management (CIKM'99), Nov 1999.

[Parthasarathy et al. 2003] S. Parthasarathy, A. Veloso, M. Otey, and W. Meira. "Parallel and Distributed Frequent Itemset Mining on Dynamic Datasets". In Proceedings of the IEEE International Conference on High Performance Computing, 2003.

[Pirolli et al. 1996] P. Pirolli, J. Pitkow and R. Rao: "Silk from a sow's ear: Extracting useful structures from the Web". In Proceeding of 1996 Conference on Human Factors in Computing Systems (CHI-96), Vancouver, British Columbia, Canada, pages 118-125, 1996.

[Pitkow & Bharat 1994] J. Pitkow and K. Bharat: "Webviz: A tool for world-wide web access log analysis". In Proceedings of the First Intenational WWW Conference, 1994.

[Schafer et al. 2001] J. Schafer, J. Konstan and J. Riedl: "E-commerce recommendation applications". Data mining and knowledge discovery, 5, pages 115-153, 2001.

[Srikant & Agrawal 1996] R. Srikant and R. Agrawal. "Mining Sequential Patterns: Generalizations and Performance Improvements". In Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96), pages 3-17, Avignon, France, September 1996.

[Srivastava et al. 2000] J. Srivastava, R. Cooley, M. Deshpande and P.-N. Tan: "Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data". SIGKDD Explorations, 1(2), 12-23, January 2000.

[Tan & Kumar 2002] P-N. Tan and V. Kumar: "Discovery of Web Robot Sessions Based on their Navigational Patterns". Data Mining and Knowledge Discovery, 6, 9-35, 2002.

[Thomas et al. 1997] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka "An efficient algorithm for the incremental updation of association rules", in Proceedings of the 3rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 1997.

[Veloso et al. 2001] A. Veloso, B. Possas, W. Meira Jr. and M. B. de Carvalho, "Knowledge management on association rule mining". In Proceedings of the International Workshop on Integrating Data Mining and Knowledge Management, San José, USA, November 2001.

[Veloso et al. 2002a] A. Veloso, W. Meira Jr., M. Bunte, S. Parthasarathy, and M. Zaki: "Mining frequent itemsets in evolving databases". In Proceedings of the 2nd SIAM International Conference on Data Mining, USA, 2002.

[Veloso et al. 2002b] A. Veloso, B. Gusmão, W. Meira Jr., M. Carvalho, S. Parthasarathy, and M. Zaki: "Efficiently Mining Approximate Models of Associations in Evolving Databases". In Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2002), Helsinki, Finland, pages 435-448, August 2002.

[Veloso et al. 2003] A. Veloso, W. Meira, M. Carvalho, S. Parthasarathy and M. Zaki. "Parallel, Incremental and Interactive Mining for Frequent Itemsets in Evolving Databases". In Proceedings of the Sixth SIAM Workshop on High Performance Data Mining, May 2003.

[Ypma & Heskes 2002] A. Ypma and T. Heskes, "Categorization of web pages and user clustering with mixtures of hidden Markov models", In Proceedings of 14th Belgian-Dutch Conference on AI - BNAIC'02, Leuven, Belgium, pages 505-506, October 2002.

[Zaiane 2001] O. Zaiane: "Web Usage Mining for a Better Web-Based Learning Environment". In Proceedings of Conference on Advanced Technology for Education, pages 60-64, Banff, Alberta, June 27-28, 2001.

[Zaki 1998] M. Zaki: "Efficient enumeration of frequent sequences". In Proceedings of the 7th International Conference on Information and Knowledge Management, November 1998.

[Zheng et al. 2001] Z. Zheng, R. Kohavi and L. Mason: "Real World Performance of Association Rule Algorithms". In Proceedings of KDD-2001, 2001.

[Zheng et al. 2002] Q. Zheng, K. Xu, S. Ma and W. Lv: "The algorithms of Updating Sequential Patterns". The 2nd SIAM (Society for Industrial and Applied Mathematics) Data Mining'2002: workshop HPDM (High Performance Data Mining), Washington, USA, April 2002.

[Zheng et al. 2003] Q. Zheng, K. Xu and S. Ma: "When to Update Sequential Patterns of Stream Data?". In Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Korea, LNAI 2637, pages 545-550, 2003.

[Zhou & Ezeife 2001] Z. Zhou and C. I. Ezeife: "A low-scan incremental association rule maintenance method". In Proceedings of the 14th Canadian Conference on Artificial Intelligence, June 2001.

WWW References

[WWW01] http://www.python.org/

This site provides a large amount of information about the scripting and object-oriented language Python. It has current and old versions of the language, documentation and current researches.

[WWW02] http://wwwai.wu-wien.ac.at/~hahsler/research/association_rules/measures.html

This site provides detailed information about most of the existing objective measures of pattern interestingness.

[WWW03] http://www.w3.org

World Wide Web Consortium (W3C) - an international consortium composed of member organizations, a full-time staff, and the public that work together to develop Web standards. It develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding.

[WWW04] www.personalization.org

Personalization Consortium - an international advocacy group formed to promote the development and use of personalization technology on the Web.

[WWW05] http://www.java.sun.com

This site provides a large amount of information about the object-oriented language Java from Sun Microsystems. It is possible to find a wide variety of resources including current

and past versions of the language, a rich documentation of each version, discussion groups, researches, and much more.

[WWW06] http://www.netbeans.org

This site provides a large amount of information about the powerful Netbeans software development product from Sun Microsystems. It is an open source tool specially oriented to address the needs of software developers using Java.