ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

RECOMMENDATION MODELS FOR WEB USERS: USER INTEREST MODEL AND CLICK-STREAM TREE

Ph.D. THESIS
Şule GÜNDÜZ, M.Sc.
(504992401)

Department	**:**	**Computer Engineering**
Programme	**:**	**Computer Engineering**
Supervisor	**:**	**Prof. Dr. Eşref ADALI**

OCTOBER 2003

Acknowledgements

First and foremost, I would like to thank my husband Emre Öğüdücü for his support in all phases of my life. I would also like to thank to my advisor Prof. Dr. Eşref Adalı for encouraging me to start PhD, to work on data mining, and for his support, advice and guidance throughout my study. There is no doubt that my stay at the University of Waterloo, between October 2001 to January 2003 had a big impact on this thesis, as well as on my whole career as a researcher and as an academician. I thank to Prof. Dr. M. Tamer Özsu, for accepting me to his lab, for his time, patience and all the effort he has put on this study, and for his advices that allowed me to exploit his expertise in the scientific world. I am honoured to be one of his students.

The research in this thesis has been supported by three research programmes: by Istanbul Technical University Grant for Long Term Research Activities from October 2001 to February 2002, by a grant from Tinçel Vakfı and by a grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

I thank to Mirko Seifert, Andreas Doms (University of Dresden), Lei Chen (University of Waterloo) for valuable discussions that improved this research.

Finally, I thank my parents for their continuous support that brought me to this point.

October 2003 Şule GÜNDÜZ ÖĞÜDÜCÜ

Contents

ABBREVIATIONS

AI	: Artificial Intelligence
CGI	: Common Gateway Interface
CF	: Collaborative Filtering
CST	: Click-Stream Tree
CSTM	: Click-Stream Tree Model
C.Net	: ClarkNet
DB	: Database
EM	: Expectation-Maximization
E-step	: Expectation step
GGM	: Generalizable Gaussian Mixture
HTML	: Hypertext Markup Language
HTTP	: Hypertext Transfer Protocol
IF	: Information Filtering
IR	: Information Retrieval
ISP	: Internet Service Provider
LCS	: Longest Common Subsequences
LHS	: Left-hand-side
LRS	: Longest Repeated Subsequence
MAP	: Maximum Aposteriori
ML	: Maximum Likelihood
M-step	: Maximization step
NASA	: NASA Kennedy Space Center
OLAP	: On-line Analytical Processing
RDMS	: Relational Database Management Systems
RHS	: Right-hand-side
RMM	: Relational Markov model
UIM	: User Interest Model
URL	: Uniform Resource Locators
UOS	: University of Saskatchewan
W3C	: World Wide Web Consortium
WUM	: Web Utilization Miner
WWW	: World Wide Web

List of Tables

List of Figures

RECOMMENDATION MODELS FOR WEB USER: USER INTEREST MODEL AND CLICK-STREAM TREE

SUMMARY

One of the applications areas of data mining is *World Wide Web* (WWW), which serves as a huge, widely distributed, global information service center for every kind of information such as news, advertisements, consumer information, financial management, education, government, e-commerce, health services, and many other information services. With the rapid growth of the WWW, it becomes more important to find the useful information from these huge amount of data. The Web also contains a rich and dynamic collection of hyperlink information and Web page access and usage information, providing sources for data mining. The Web poses great challenges for effective knowledge discovery and data mining applications. Web mining is defined as the use of data mining techniques to automatically discover and extract information from Web documents and services. In general, Web mining is a common term for three knowledge discovery domains that are concerned with mining different parts of the Web: Web Structure Mining, Web Content Mining and Web Usage Mining.

While Web structure and content mining utilize real or primary data on the Web, Web usage mining works on the secondary data such as Web server access logs, proxy server logs, browser logs, user profiles, registration data, user sessions or transactions, cookies, user queries, and bookmark data. The continuous growth of the World Wide Web and available data in that domain imposes new methods of design and development of powerful yet computationally efficient Web usage mining tools. Web usage mining refers to the application of data mining techniques to discover usage patterns in order to understand and better serve the needs of Web-based applications. It has been used to improve the Web performance through caching, to recommend related pages, improve search engines and personalize browsing in a Web site.

Due to the steady growth of available information combined with the almost unstructured Web data, it becomes more difficult to find relevant and useful information for Web users. Thus, one of the goals of Web usage mining is to guide the Web users to discover useful knowledge and to support them for decision-making. In that context, predicting the needs of a Web user as she visits Web sites has gained importance. The requirement for predicting user needs in order to guide the user in a Web site and improve the usability of the Web site can be addressed by recommending pages to the user that are related to the interest of the user at that time.

This thesis develops and tests two models for discovering and modelling of the user's interest in a single session. These approaches rely on the premise that the visiting time of a page is an indicator of the user's interest in that page.

The first model, User Interest Model, uses only the visiting time and visiting frequencies of pages without considering the access order of page requests in user sessions. The resulting model has lower run-time computation and memory requirements, while providing predictions that are at least as precise as previous proposals. Our objective in this model is to assess the effectiveness of non-sequentially ordered pages in predicting navigation patterns. The main idea behind this work is that user sessions can be clustered according to the similar amount of time that is spent on common pages among sessions.

The second model, Click-Stream Tree, considers both the order information of pages in a session and the time spent on them. User sessions are clustered according to their pair-wise similarity and the resulting clusters are then represented by a click-stream tree. A new method is proposed for calculating the similarity between all pairs of user sessions considering both the order of pages and the time spent on them.

The results of the experiments on different Web sites show that the models are robust across Web sites and they could be used for caching as well. The results show that proper normalization of time yields a good prediction accuracy. Furthermore, the models are quite effective in representing a Web user’s access pattern and have an advantage over previous proposals in terms of speed and memory usage.

WEB KULLANICILARI İÇİN ÖNERİ MODELLERİ: KULLANICI İLGİSİ MODELİ VE TIKLAMA İZİ AĞACI

ÖZET

World Wide Web (WWW) haber, reklam, tüketici bilgisi, mali yönetim, eğitim, hükümet, e-ticaret, sağlık hizmeti ve birçok başka bilgi hizmeti için çok büyük, oldukça yayılmış, genel bilgi hizmet merkezi olduğundan veri madenciliğinin bir uygulama alanıdır. WWW'nin hızla büyümesi ile bu çok büyük miktardaki veri içinden yararlı bilgiyi bulmak daha önemli olmuştur. Web ayrıca veri madenciliği için kaynak oluşturan bol ve dinamik bağlantı bilgisi ve Web sayfası erişim ve kullanım bilgisi içerir. Web etkili bilgi keşifi ve veri madenciliği uygulamaları için büyük bir çalışma ortaya koymaktadır. Web madenciliği, Web doküman ve hizmetlerinden otomatik olarak bilginin keşfedilmesi ve elde edilmesi için veri madenciliği tekniklerinin kullanılması olarak tanımlanmıştır. Genel olarak, Web madenciliği Web'in değişik alanlarıyla ilgilenmesine göre ayrılan üç bilgi keşfi alanı için ortak bir terimdir: Web yapısı madenciliği, Web içerik madenciliği, Web kullanım madenciliği.

Web yapısı madenciliği ve Web içerik madenciliği Web'teki asıl ya da birincil veriyi kullanırken, Web kullanım madenciliği Web sunucu günlüğü, vekil sunucu günlüğü, tarayıcı günlüğü, kullanıcı profili, kayıt verisi, kullanıcı oturumu veya işlemi, çerez, kullanıcı sorgulaması ve yer imi verisi gibi ikincil veri üzerinde çalışır. World Wide Web'in ve bu alanda elde edilen verinin sürekli büyümesi, güçlü ve verimli Web kullanım araçlarının tasarımı ve geliştirilmesi için yeni yöntemleri zorunlu kılar. Web kullanım madenciliği, Web tabanlı ihtiyaçların anlaşılması ve daha iyi hizmet verilmesi amacıyla kullanıcı örüntülerini ortaya çıkarmak için veri madenciliği tekniklerinin uygulanması ile ilgilenir. Ön belleğe alma yöntemi ile Web başarımını artırma, ilgili sayfaları önerme, arama motorlarını geliştirme ve bir Web sitesine göz atarken kişiselleştirme için kullanılır.

Hemen hemen yapısız Web verisiyle birleşmiş sürekli büyüyen mevcut bilgiden konu ile ilgili ve yararlı bilgiyi bulmak daha güç olmuştur. Böylece Web kullanım madenciliğinin amaçlarından biri de Web kullanıcılarına, yararlı bilgiyi ortaya çıkarmak ve karar vermede desteklemek için yol göstermektir. Bu bağlamda Web sitesini ziyaret ederken bir Web kullanıcısının ihtiyaçlarını öngörmek önem kazanmıştır. Kullanıcıya Web sitesi içinde yol göstermek ve Web sitesinin kullanılabilirliğini artırmak için kullanıcı ihtiyaçlarını öngörme gereksinimi kullanıcının o andaki ilgisiyle bağlantılı sayfalar önererek çözümlenebilir.

Bu tez bir oturumdaki kullanıcı ilgisini ortaya çıkarmak ve modellemek için iki model geliştirmiş ve test etmiştir. Bu yaklaşımlar bir sayfayı ziyaret süresinin kullanıcının o sayfaya olan ilgisine iyi bir gösterge olduğuna dayanır.

İlk model, kullanıcı ilgisi modeli, kullanıcı oturumlarında sayfa erişim düzenini göz önünde bulundurmadan sadece sayfanın ziyaret süresini ve ziyaret sıklığını kullanır. Ortaya çıkan model en az önceki öneriler kadar doğru öngörü sağlarken daha az çalıştırma hesaplaması ve bellek gereksinimine sahiptir. Bu modeldeki hedefimiz yön bulma örüntülerini öngörürken sırasız düzenlenmiş sayfaların etkisini değerlendirmektir. Bu çalışma ardındaki ana düşünce kullanıcı oturumlarının, oturumlar arasındaki ortak sayfalarda benzer süre geçirmelerine göre demetlenebilir olmasıdır.

İkinci model, tıklama izi ağacı modeli, hem sayfaların bir oturum içindeki düzenini hem de bu sayfalarda geçirilmiş süreleri göz önünde bulundurur. Kullanıcı oturumları çiftli benzerliklerine göre demetlenir ve ortaya çıkan demetler bir tıklama izi ağacı ile temsil edilir. Hem sayfaların düzeni hem de o sayfalarda geçirilen süre göz önünde bulundurularak bütün kullanıcı oturumu çiftleri arasındaki benzerliği hesaplamak için bir yöntem önerilmiştir.

Değişik Web siteleri üzerinde yapılan deneyler modellerin Web siteleri arasında sağlam olduğunu ve cep belleğe alma için de kullanılabileceğini göstermektedir. Sonuçlar sürenin uygun normalizasyonunun iyi bir öngörü doğruluğu sağladığını göstermiştir. Ayrıca modeller Web kullanıcılarının erişim örüntülerini temsil etmede oldukça etkilidir ve hız ve bellek kullanımı açısından daha önceki önerilere göre bir avantaja sahiptir.

1. INTRODUCTION

Data mining has emerged as one of the most exciting and dynamic fields in computer science and software engineering. The terms "data mining" and "knowledge discovery in databases" are often used synonymously. Knowledge discovery in databases is the process of identifying valid, novel, potentially useful, and ultimately understandable patterns/models in data. Data mining is a step in the knowledge discovery process consisting of particular data mining algorithms that, under some acceptable computational efficiency limitations, finds patterns or models in data [1]. Simply stated, data mining refers to the process of extracting previously unknown, valid, and potentially useful knowledge from data. Another definition is that data mining is a variety of techniques used to identify nuggets of information or decision-making knowledge in bodies of data, and extracting these in such a way that they can be put to use in areas such as decision support, prediction, forecasting, and estimation. The data is often voluminous but, as it stands, of low value as no direct use can be made of it; it is the hidden information in the data that is useful. For this reason data mining is often referred to as "secondary" data analysis.

Data mining roots are traced back along three family lines. The longest of these three lines is classical statistics. Without statistics, there would be no data mining, as statistics is the foundation of most technologies on which data mining is built. Classical statistics embrace concepts such as regression analysis, standard distribution, standard deviation, standard variance, discriminant analysis, cluster analysis, and confidence intervals, all of which are used to study data and data relationships. These are the building blocks with which more advanced statistical analysis are underpinned. Certainly, classical statistical analysis plays a central role in current data mining tolls and techniques.

Data mining's second longest family line is Artificial Intelligence (AI). This discipline, which is built upon heuristics as opposed to statistics, attempts to apply human-thought-like processing to statistical problems. Because this approach requires vast computer processing power, it was not practical until the early 1980s, when computers began to offer useful power at reasonable prices. AI found a few applications at the very high end scientific/government markets, but the required supercomputers of the era priced AI out of the reach of virtually everyone else. The notable exceptions were certain AI concepts which were adopted by some high-end commercial products, such as query optimization modules for Relational Database Management Systems (RDBMS).

The third family line of data mining is machine learning, which is more accurately described as the union of statistics and AI. While AI was not a commercial success, its techniques were largely co-opted by machine learning. Machine learning, able to take advantage of the ever-improving price/performance ratios offered by computers of the 80s and 90s, found more applications because the entry price was lower than AI. Machine learning could be considered an evolution of AI, because it blends AI heuristics with advanced statistical analysis. Machine learning attempts to let computer programs learn about the data they study, such that programs make different decisions based on the qualities of the studied data, using statistics for fundamental concepts, and adding more advanced AI heuristics and algorithms to achieve its goals.

Data mining, in many ways, is fundamentally the adaptation of machine learning techniques to business applications. It is best described as the union of historical and recent developments in statistics, AI, and machine learning. These techniques are then used together to study data and find previously-hidden trends or patterns within. Some applications of data mining are the following:

- ***Common Business Applications:*** Such as market analysis and management, market basket analysis, cross selling, target marketing, customer profiling, customer behavior understanding, risk analysis and management;
- ***Fraud Detection and Management:*** Detecting telephone fraud, detecting automotive and health insurance fraud, detecting credit card fraud, detecting suspicious money transactions (money laundering);
- ***Text Mining:*** Message filtering (e-mails, newsgroup etc.);
- ***Medicine:*** Association of pathology and symptoms, analysis tolls for DNA arrays, medical imaging;
- ***Sports:*** Game statistics;

- ***Web Surfing and Mining:*** Recommending related pages, improving search engines or personalizing browsing in a Web site or caching.

1.1 Web Mining

One of the applications areas of data mining is *World Wide Web* (WWW), which serves as a huge, widely distributed, global information service center for every kind of information such as news, advertisements, consumer information, financial management, education, government, e-commerce, health services, and many other information services. With the rapid growth of the WWW, it becomes more important to find the useful information from these huge amounts of data. The Web also contains a rich and dynamic collection of hyperlink information and Web page access and usage information, providing sources for data mining. The Web poses great challenges for effective knowledge discovery and data mining applications. Web mining is defined as the use of data mining techniques to automatically discover and extract information from Web documents and services [2].

In general, Web mining is a common term for three knowledge discovery domains that are concerned with mining different parts of the Web: ***Web Structure Mining***, ***Web Content Mining*** and ***Web Usage Mining*** [3, 4].

1.1.1 Web Structure Mining

Web structure mining generates a structural summary of Web sites and Web pages. Given a collection of interconnected Web documents, interesting information can be discovered using the link information. The following structural information from the Web documents can be extracted:

- Measuring the frequency of the local links that connect different Web documents at the same Web site. This gives information about inter-related pages at the same Web server and the the ability of a Web document to cross-reference other related Web pages within the same Web server;
- Measuring the frequency of documents that have links to the documents at different Web sites. This measures the visibility of Web documents and ability to relate similar and related documents across different Web sites;

- Measuring the frequency of identical Web documents. This measures the replication of Web documents across different Web sites and may help to identify the mirrored sites.

1.1.2 Web Content Mining

As of early 2003, there were just over three billion Web pages listed in the Google search engine[1] index, widely taken to be the most comprehensive Web index. Netcraft Web server survey [5] lists 39.174.349 Web sites as of March, 2003. No one knows how many more Web pages there are on the Internet, or the total number of documents available over the public network, but there is no question that the number is enormous and growing quickly. Every one of these Web pages has come into existence within the past ten years. Today, Web users access the Web through two dominant interfaces: Clicking on hyperlinks and searching via keyword queries. Today's search engines have to cope with rapidly changing, heterogenous data collections that are orders of magnitude larger than ever before. They also have to remain simple enough for average and novice users to use. Better support is needed for expressing one's information need and and dealing with a search result. Because of the sheer number of documents available, we can find interesting and relevant results for any search query. The problem is that those results are likely to be hidden in a mass of semi-relevant and irrelevant information, with no easy way to distinguish the good from the bad. Web content mining has significant roles to play in addressing this problem. Web content mining describes the discovery of useful information from the Web contents and documents.

The unstructured nature of Web content mining forces a different approach towards Web content mining. Basically, the Web content consist of several types of data such as textual, image, audio, video, metadata, as well as hyperlinks. The research done in Web content mining could be differentiated from two different points of view: Information Retrieval (IR) and Database (DB) views [6]. The goal of Web content mining from the IR view is mainly to assist or to improve the information finding or filtering the information to the users usually based on either inferred or solicited user profiles, while the goal of Web content mining from the DB view mainly tries to model data on the Web for solving the problems of managing and querying the information on the Web.

[1]http://www.google.com/

1.1.3 Web Usage Mining

While Web content and structure mining utilize real or primary data on the Web, Web usage mining works on the secondary data such as Web server access logs, proxy server logs, browser logs, user profiles, registration data, user sessions or transactions, cookies, user queries, and bookmark data. Web usage mining refers to the application of data mining techniques to discover usage patterns from these secondary data, in order to understand and better serve the needs of Web-based applications. The usage data collected at different sources will represent the navigation patterns of different segments of the overall Web traffic, ranging from single-user, single-site browsing behavior to multi-user, multi-site access patterns. The information provided by the data sources can all be used to construct/identify several data abstractions, such as users, server sessions, episodes, click stream, and page views [7].

With the rapid growth of the WWW, the study of modelling and predicting a user's access on a Web site has become more important. It has been used to improve the Web performance through caching [8, 9] and prefetching [10, 11], to recommend related pages [12], improve search engines [13] and personalize browsing in a Web site [10].

1.2 Goal of the Thesis

The goal of this work is to build a recommendation model that has an advantage over previous proposals in terms of speed and memory usage by preserving the prediction accuracy. The model can be used to guide a user during her visit to a Web site as well as for prefetching of Web pages. The model does not capture personalization which can be defined as any action that makes the Web experience of a user personalized to the user's taste. On the contrary, we regard that every user may have different desires at different times. Thus, we intend to build a recommendation model that can recommend different Web pages to the same user every time she visits the Web site. The main goal of this work is to model each user session in terms of Web users' behavior on a Web site instead of modelling the behavior of each Web user.

1.3 Related Works

It is often necessary to make choices without sufficient personal experience of the alternatives. In every day life, we rely on recommendations from other people either by word of mouth, recommendation letters, movie and book reviews, or general surveys. Recommender systems assist and augment this natural social process. Since WWW serves as a huge, widely distributed, global information service center for every kind of information such as news, advertisements, consumer information, financial management, education, government, e-commerce, health services, and many other information services, it becomes more important to find the useful information from these huge amounts of data. Recommender systems on Internet help people make decisions in this complex information space where the volume of information is available to them is huge. Given a user's (who may, for example, be a customer in an e-commerce site) current actions, the goal is to determine which Web pages (items) will be accessed (bought) in the near future. Recommender systems on Internet can be divided into two parts: Recommender systems based on collaborative filtering and automated recommender systems. This section describes some approaches used in these systems.

1.3.1 Collaborative Filtering

One of the most successful and widely used technologies for building recommendation systems is *Collaborative Filtering* (CF). Goldberg et al. introduced the phrase "collaborative filtering", while describing Tapestry [14], which later became known as the first recommender system. The system relied on the explicit opinions of people from a small community, such as an office workgroup. However, recommender systems for large communities can not depend on each person knowing the others. Later on several ratings-based automated recommender systems were developed. CF systems collect visitor opinions on a set of objects, using ratings provided by the users or implicitly computed, to form peer groups and that establishes the basis of a learning system to predict a particular user's interest in an item. It is often based on matching, in real-time, the current user's profile against similar records (nearest neighbors) obtained by the system over time from other users. The ratings collected by the system may be both implicit and explicit.

Explicit voting refers to a user consciously expressing her preference for a title, usually on a discrete numerical scale. Some examples of systems that use this approach include SIFT [15], Tapestry [14] and the system described in [16]. The GroupLens project [17] is a purely CF approach that automates prediction by collecting explicit user ratings and employing statistical techniques.

The lack of explicit user ratings as well as the sparseness and the large volume of data pose limitations to standard CF. As a result, it becomes hard to scale CF techniques to a large number of items, while maintaining reasonable prediction performance and accuracy. A number of optimization strategies have been proposed and employed to remedy this shortcoming. These strategies include similarity indexing and dimensionality reduction to reduce real-time search costs. Pattern discovery techniques have been proposed to address some of the limitations of collaborative methods. A hybrid approach to recommendations combines aspects of both pattern discovery methods and CF. O'Conner and Herlocker use existing data partitioning and clustering algorithms to partition the set of items based on user rating data citech99 . Predictions of items are then computed independently within each partition.

Another example to hybrid approaches is Content-based systems, which work by comparing text descriptions or other representations associated with an item. Balabonović and Shoham [18] describe a system that helps users to discover new and interesting sites that are of interest to them. The system uses AI techniques to present users with a number of documents that it thinks the user would find interesting. Users evaluate the documents and provide feedback to the system. From the feedback, the system knows more about the users' areas of interest in order to better serve them in subsequent searches. The hybrid approach used in this model retains the advantages of content-based and collaborative approaches while overcoming their disadvantages. The recommendation system, termed Yoda, uses a hybrid approach that combines CF and content-based querying [19]. Yoda identifies similar groups of users by clustering user sessions from a training set, and learns typical patterns of user interests in each cluster by taking a vote among items browsed by users belonging to the cluster. The pattern of each cluster is then used to predict a list of recommendations.

The system described in [20] aims at offering innovative on-line services to support the trade fair business processes among a great number of exhibitors organized in a Web-based virtual fair. In order to build user profiles and provide recommendations, a method has been implemented which is based on the integration of data collected explicitly and implicitly about users. The system then provides appropriate recommendations to the user in any circumstances during the visit. In [21] a CF framework is used to combine personal Information Filtering (IF) agents and the opinions of a community of users to produce better recommendations than either agents or users can produce alone.

Implicit rating used for CF can be divided into three categories: 1^o rating based on examination, when a user examines an item; 2^o rating based on retention, when a user saves an item; and 3^o rating based on reference, when a user links all or part of an item into an other item. PHOAKS [22] represents people by their mention of Uniform Resource Locators (URL) in Usenet messages. Usenet is a world-wide distributed discussion system that consists of a set of "newsgroups" with names that are classified hierarchically by subject. "Articles" or "messages are posted to these newsgroups by people on computers with the appropriate software. An URL in a Usenet messages is considered an implicit rating. The process of recommendation in PHOAKS entails mining URLs, filtering irrelevant links via a number of heuristics and computing a weight for each. A link's weight is the number of times the link appears in the Usenet messages. Finally the output is a set of relevant URL's and their associated weights. Siteseer [23] is another collaborative system that uses implicit ratings. Its recommendation function is based on bookmark folder representation of people. Bookmarks are an implicit declaration of interest of Web users that is less noisy in comparison to other mechanisms such as mouse click or a URL embedded in a newsgroup message. Furthermore, binary nature of bookmark eliminates the possibility of partial preference. The recommendation function of Siteseer computes set intersection between input bookmark folders. The output of the recommendation function is a set of bookmarks.

CF techniques can be an important part of the recommender systems. One key advantage of CF based on explicit voting is that it does not consider the content of the items being recommended, rather than map user to items through user ratings. While CF systems based on explicit voting have proven to be accurate enough for entertainment domains [17, 24], they have yet to be successful in content domains where higher risk is associated with accepting a filtering recommendation. In addition to the limitations mentioned above, another difficult, though common problem of CF systems is the cold-start problem, where recommendations are required for items that no one in the data set has yet rated.

1.3.2 Automated Recommender Systems

Most of the techniques for automated recommender systems are based on data mining methods, which attempt to discover patterns or trends from a variety of sources. Web usage mining is an obvious and popular one of these techniques. Recently, a number of approaches have been developed dealing with specific aspects of Web usage mining like automatically discovering user profiles, recommender systems, Web prefetching, design of adaptive Web sites, etc. In all these applications the goal is the development of an effective prediction algorithm. The core issue in prediction is the development of an effective algorithm that deduces the future user requests. The most successful approach towards this goal has been the exploitation of the user's access history to derive prediction. This section describes pattern discovery methods that have been applied to Web domain. It is very difficult to classify the studies according to the methods they use for Web usage mining. In most of the works mentioned below, methods are combined together in discovering usage patterns in Web domain.

Statistical Analysis

Statistical techniques are the most common methods to extract knowledge about visitors to a Web site. By different kinds of statistical analysis (frequency, median, mean, etc.) of the session file, one can extract statistical information such as the most frequently accessed pages, average view time of a page or average length of a path through a site. This type of knowledge can be potentially useful for improving the system performance, enhancing the security of the system, and providing support for marketing decisions.

An example for the application of statistical methods to Web mining is PageGather [25]. This algorithm processes the access logs by using a statistical approach to find pages that are often visited together by the site's user. It then creates a graph in which each node represents a page at the Web site and finds maximal cliques in the graph in order to discover user profiles. While the generated profiles were not integrated as part of a recommender system, they were used to automatically synthesize alternative static index pages for a site.

Larsen et al. used a hierarchical probabilistic clustering method based on the Generalizable Gaussian Mixture (GGM) model in [26] for analysis and interpretation of WWW data. The unsupervised GGM model is applied for segmentation of user's behavior when shopping on a Web site. However, it has not been extended to recommend new items during shopping.

Borges and Levene modelled the user navigation sessions as hypertext probabilistic language generated by a hypertext probabilistic grammar, which has a one-to-one mapping between the set of non-terminal symbols and the set of terminal symbols [3]. Each non-terminal symbol corresponds to a Web page. The higher probability generated navigation paths of the hypertext probabilistic grammar correspond to the user preferences when navigating through the Web. Moreover, the use of entropy as an estimator of the statistical properties of grammar is proposed in this study.

Association Rules

Association rules capture the relationships among items based on their patterns of co-occurrence across transactions. The problem of discovering association rules was introduced in [27]. Given a set of transactions, where each transaction is a set of items, an association rule is an expression of the form $X \Rightarrow Y$, where X (defined as the left-hand-side (LHS) of the association rule) and Y (defined as the right-hand-side (RHS) of the association rule) are sets of items such that no item appears more than once in $X \cup Y$. The intuitive meaning of such a rule is that transactions in the database which contain the items in X tend to also contain the items in Y. Two common numeric measures assigned to each association rule are "support" and "confidence". Support quantifies how often the items in X and Y occur together in the same transaction as a fraction of the total number of transactions, or $\frac{|X \cup Y|}{|D|}$ where $|D|$ denotes the total number of transactions. Confidence quantifies how often X and Y occur together as a fraction of the number of transactions in which X occurs, or $\frac{|X \cup Y|}{|X|}$. In the context of Web usage mining, association rules refer to sets of pages that are accessed together with a support value exceeding some specified threshold. These pages may not be directly connected to one another via hyperlinks. For example, using association rule discovery techniques, we can find correlations such as following:

- 40% of users visit the Web page with URL /home/page1, and the Web Page with URL /home/page2 in the same user session.

- 30% of users who accessed the Web page with URL /home/products, also accessed /home/pruducts/computers.

Yang et al. used association rules extracted from Web server logs for Web caching and prefetching [28]. For any given observed sequence of URL's the algorithm chooses a rule whose LHS matches the sequence and has the longest length among all applicable rules. However, the sequential data is not considered in that work.

Chen, Park and Yu converted the log data into a set of maximal forward references, a form which is amenable to being processed by existing association rule techniques [29]. Two algorithms are given to mine the rules, which in this context consist of large itemsets with the additional restriction that the references must be consecutive in a transaction.

Clustering

Clustering is a technique to group together a set of items having similar characteristics. In the Web usage domain, there are three kinds of interesting clusters to be discovered: 1^o Session clusters; 2^o User clusters; and 3^o Page clusters. Session clustering implementation allows clustering of user sessions in which users have similar access patterns. Clustering of users tends to establish groups of users exhibiting similar browsing patterns. Page clustering can be partitioned into two methods. The first is to cluster pages according to their contents. For this method an analysis of the content of Web site is needed. The second method computes clusters of page references based on how often they occur together.

In [30] a method is proposed to classify Web site visitors according to their access patterns. Each user session is stored in a vector that contains the number of visits to each page and an algorithm is given to find clusters of similar vectors. The clusters obtained with this method do not take into account the order in which the pages were accessed. This system consists of an offline module that will perform cluster analysis and an online module which is responsible for dynamic link generation of Web pages.

Shahabi et al. described a prototype system that uses viewing time as the primary feature to describe a user session [31]. Then, using a similarity measure roughly based on inner products, they cluster the sessions using k-means clustering [32]. The system is evaluated on a fictional 34-page site with simulated path data.

Fu et al. suggested a model that uses the URLs to construct a page hierarchy which is used to categorize the pages [33]. For example, all pages under /computers/ index.html would be classified as “computers” pages. The page accesses in each user session are then described using these page categorization. This is called “Generalization-based Clustering”, and is similar to using URL tokens (tokenize URLs on “/” and other delimiters). Unfortunately, this approach only works if the URLs contain useful tokens, or if page categorization can be determined ahead of time manually.

Banerjee et al. utilized the combination of time spent on a page and Longest Common Subsequences (LCS) to cluster the user sessions [34]. The LCS algorithm is first applied on all pairs of user sessions. Then each LCS path is reduced using page hierarchy in a generalization based approach called ”Concept based Clustering“. This is basically a simpler form of generalization-based clustering, because they only use the top-most level of the page hierarchy to categorize Web pages. Then similarities between LCS paths are computed as the function of viewing time spent at each stage of the paths.

Several attempts have been made to learn the click behavior of a Web user by probabilistic clustering of individuals with a mixture of Markov models. Markov models are a popular method for modeling stochastic sequences with an underlying finite-state structure. For this reason they are well suited for modeling and predicting a user’s browsing behavior on a Web site. In general, the input for these problems is the sequence of Web pages that were accessed by a user and the goal is to build Markov models that can be used to model the user behavior during her visit to the Web site and predict the Web page that the user will most likely access next. Sarukkai used Markov models for predicting the next page accessed by the user and notes the need to reduce the size of the model by clustering the URLs [11]. Experimental results are reported which show that a Markov model can be useful both in the prediction of http requests and in the prediction of the next link to be requested. However, individual user behavior is not considered. Cadez, Gaffney and Smyth proposed a methodology for clustering individuals given the collections of their user navigation session [35]. Cadez et al. proposed a methodology for visualizing the navigational patterns characterizing each cluster presented [36]. The navigation behavior of the users is represented by a Markov model in which the pages are classified into predefined categories. This model does not analyze the log data in its finest level of detail.

Anderson, Domingos and Weld introduced a Relational Markov model (RMM) [37] which is a generalization of Markov models where states can be of different types with relational predicates. Thus, RMM groups pages of the same type in a Web site into relations, with each relation described by its own set of variables. These variables themselves are grouped together, forming a hierarchy of values and a shrinkage is carried out over the cross product of these hierarchies. In that model, if the states between relations are not reflected in the distribution of the data, RMM can preform worse then traditional Markov models.

Classification

Classification is the task of mapping a data item into one of several predefined classes. In the context of Web usage mining, one is interested in developing a usage profile belonging to a particular class or category. This requires extraction and selection of features that best describe the properties of a given class or category. In Web usage mining, classification techniques allow one to develop a profile for users who access particular server files based on their demographic information available on those users, or based on their access patterns. Classification techniques enable us to find usage patterns such as followings:

- 50% of users who placed an online order in home/products/product1 were in the 20-25 age group and lived in Istanbul.
- If an user put more than 2 items in the shopping cart, she will place an order during that visit to the site.

The “Web log miner” uses On-line Analytical Processing (OLAP) technology for prediction, classification and time series analysis of Web log data [38]. The Web log miner project has the following steps. In the first step, the data are filtered to remove irrelevant information and it is transformed into a relational database in order to facilitate the following operation. In the second step, a multi-dimensional array structure, called a data cube is built, each dimension representing a field with all possible values described by attributes. OLAP is used in the third step in order to provide further insight of any target data set from different perspectives and at different conceptual level. In the last step, data mining techniques such as data characterization, class comparison, prediction, association, classification or time-series analysis can be used on the Web log data cube and Web log database. Interesting results are obtained on Web traffic analysis and on the evolution of user behavior (e.g. preferred pages) over time.

Sequential Patterns

Sequential pattern mining, which discovers frequent subsequences as patterns in a sequence database, is an important data mining problem with broad applications, including the analysis of customer purchase behavior, Web access patterns, scientific experiments, disease treatments, natural disasters, DNA sequences, and so on. The sequential pattern mining problem was first introduced by Agrawal and Srikant in [39]. Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user specified min_support threshold, sequential pattern mining finds all of the frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min_support.

In Web server logs, a visit of a user is recorded over a period of time. A time stamp can be attached either to the user session or to the individual page requests of user sessions. By analyzing this information with sequential pattern discovery methods, the Web mining system can determine temporal relationships among data items such as the following:

- 30% of users who visited /home/products/dvd/movies, had visited /home/products/games within the past week.
- 40% of users request the page with URL /home/products/monitors after visiting the page /home/products/computers.

For example, IndexFinder uses the data about how often pages occur together in user visits [40]. IndexFinder then applies a statistical cluster mining technique [41] to the data and produces candidate clusters as output. In statistical cluster mining, objects (Web pages) are grouped together based solely on a similarity measure (how often they co-occur in user visits). The clusters are then used for optimizing the structure of Web sites.

The Web Utilization Miner (WUM) system discovers navigation patterns with interesting statistical properties like existence of cycles, repeated accesses, or rarely followed paths etc. [42]. The query processor is incorporated to the miner in order to specify characteristics of discovered paths that are interesting to the analyst.

Incorporating the mining language early in the mining process allows the construction only of patterns that have the desired characteristic while irrelevant pattern are removed. However, no performance studies were reported and the use of query language to find patterns with predefined characteristics may prevent the user finding unexpected patterns.

Schechter et al. propose a method for predicting the next move of a Web user [43]. A tree which contains the user paths is generated from the log data. The prediction of the next request of a user is based on matching the user's current session against the paths in the tree. The ranking of matches is determined by a heuristic. The maximal prefixes of each path (the first $N-1$ elements of N-length path) are compared element-wise against the same length suffixes of the user path, and the paths in the tree with the highest number of element-wise matches are returned. Partial matches are disallowed. Schecter et al. found that storing longer paths in the tree offered some improvements in prediction but they did not study this case systematically. Although this method is very accurate in the way it represents the user session, it is not very compact since for every path stored all its suffixes are duplicated.

Nanopoulos et al. modified appropriately the candidate generation procedure and the apriori-pruning criterion [44] to determine the access sequences of a Web user [45]. The work has been proposed for prefetching which refers to the process of deducing user's future request for Web objects and getting that objects into the cache before an explicit request is made for them.

Pitkow and Pirolli proposed the Longest Repeated Subsequence (LRS) algorithm [10]. A LRS is a sequence of items where 1^o Subsequence means a set of consecutive items; 2^o Repeated means the item occurs more than a threshold T, where T typically equals one; and 3^o Longest means that although a subsequence may be part of another repeated subsequence, there is at least one occurrence of this subsequence where this is the longest repeating. However, this approach exacerbates the problem of state space complexity.

Agent Based Approaches

An agent could be defined as a system that can be viewed a perceiving its environment through sensors and acting upon that environment through effectors [46]. Several intelligent Web agents have been developed for information browsing. WebWatcher is an assistant agent that helps the user by using visual representations of links that guide the user reach a particular target page or goal [47]. It learns by creating and maintaining a log file for each user and from the user feedback it improves its guidance.

Another agent based recommendation system, Letizia, learns the areas that are of interest to a user by recording the users' browsing behavior [48]. It performs some tasks at idle times (when a user is not reading a document and is not browsing). These tasks include looking for more documents that are related to the user's interest or might be relevant to future requests.

The studies in [49, 50] are examples of a knowledge intensive approach, where the agent is pre-programmed with an extensive model of what resources are available on the network and how to access them. The knowledge-based approach is complementary to the relatively pure behavior-based approach here, and they could be used together.

1.3.3 Methodology for Automated Recommender Systems

As shown in Figure 1.1, the overall process of automated recommendation can be divided into four components, namely: 1^o Data collection; 2^o Data preparation and cleaning; 3^o Pattern extraction; and 4^o Prediction.

The first step is data collection from different kind of resources. Web usage mining can potentially use data from the following resources [6]:

- ***Server Level Collection:*** A Web server log is an important source for performing Web usage mining because it explicitly records the browsing behavior of Web users. Server log files provide details about file requests to a Web server and the server response to those requests. In the access log, which is the main log file, each line describes the source of a request, the file requested, the date and time of the request, the content type and length of the transferred file, and other data such as errors and the identity of referring pages.
- ***Client Level Collection:*** Client-side data collection can be implemented by using a remote agent (such as Java scripts or applets) or by modifying the source code of an existing browser (such as Mosaic or Mozilla) to enhance its data collection capabilities. The implementation of client-side data collection

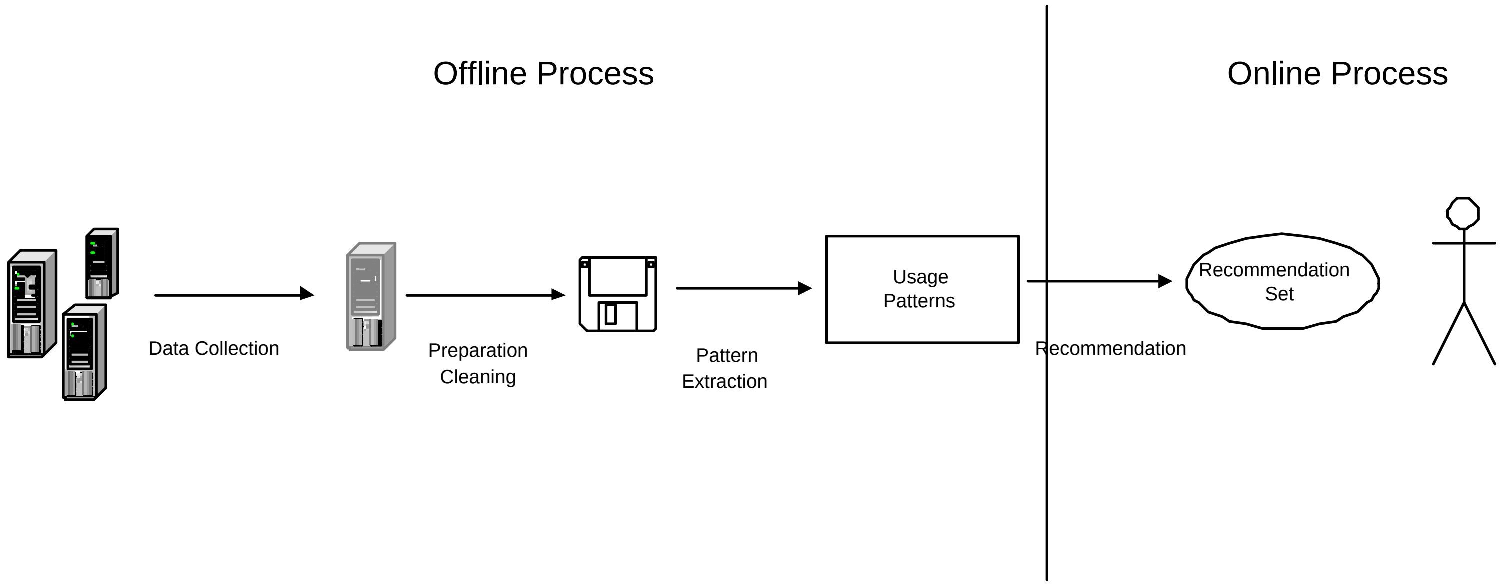


Figure 1.1: Methodology of recommender systems

methods requires user cooperation. Client-side data collection ameliorates problems caused by caching and session identification. A modified browser enables data collection about a single user over multiple Web sites. However, the most difficult part of using this method is convincing the users to use the browser for daily browsing activities.

- ***Proxy Level Collection:*** A Web proxy acts as an intermediate level of caching between client browsers and Web servers. Proxy server may serve as a data source for characterizing the browsing behavior of a group of anonymous users sharing a common proxy server.

The data resource should be selected according to the application. The second step is to clean the data and prepare for mining the usage patterns. Fundamental methods of data cleaning and preparation have been well studied in [51, 52, 53] and the details of this step will be given in the Chapter 2. The third step is to extract usage patterns. The main techniques traditionally used for modelling usage patterns in a Web site are mentioned above. The fourth step is to build a predictive model based on the extracted usage patterns. The prediction step is the real-time processing of the model, which considers the active user session and makes recommendations. Once the mining tasks are accomplished, the discovered patterns are used by the online component of the model to provide dynamic recommendations to users based on their current navigational activity. The Web server keeps track of the active user session as the user browser makes HTTP requests. This can be accomplished by a variety of methods such as URL rewriting, or by temporarily caching the Web server access logs. The produced recommendation set is then added to the last requested page as a set of links before the page is sent to the client browser.

1.4 Contributions of the Thesis

The most commonly used techniques to predict the user's next request are sequential patterns, association rules and Markov models. These techniques work well for Web sites that do not have a complex structure, but experiments on complex, highly interconnected sites show that the storage space and runtime requirements of these techniques increase due to the large number of patterns for sequential pattern and association rules, and the large number of states for Markov models.

The discovery of usage patterns discussed above is not sufficient to accurately describe the user's navigational behavior in a *server session*. An important feature of the user's navigation path is the time that a user spends on different pages [31]. Even the same person may have different desires at different times. If we knew the desire of a user every time she visits the Web site, we could use this information for recommending pages. Unfortunately, experience shows that users are rarely willing to give explicit feedback. Thus, the time spent on a page is a good measure of the user's interest in that page, providing an implicit rating for that page. If a user is interested in the content of a page, she will likely spend more time there compared to the other pages in her session.

In this work, we present two new models that use the time spent on visiting pages and study the impact of time, that a user spent on each page during her single visit to a Web site. The first model (User Interest Model) uses only the visiting time and visiting frequencies of pages without considering the access order of page requests in user sessions. The resulting model has lower run-time computation and memory requirements, while providing predictions that are at least as precise as previous proposals. The second model (Click-stream tree model) uses both the sequences of visiting pages and the time spent on that pages. As far as we know, existing tools for mining two different information types like the order of visited Web pages and the time spent on those pages, are hard to find. Therefore, we concentrate in this study on a model that well reflects the structural information of a user session and handles two-dimensional information. This approach to recommendation is novel and unique. The experimental results of both of the models show that using time improves the accuracy of the prediction of the next request. Equally important, these results are robust across sites with different structures. To confirm our findings, the results are compared to the results of three other well known recommendation techniques.

1.5 Organization of the Thesis

This thesis is organized as five chapters. We propose in this thesis two new models for recommendation of Web pages in a single site. As mentioned in Section 1.3.3 the process of recommendation has four components. The first two components , data collection and data preparation are common for two models. Therefore these two components are explained in detail in Chapter 2. Our original work is where usage patterns are extracted and a recommendation set is generated based on these extracted patterns. For this objective, two new models are proposed and explained in detail in Chapter 3. Chapter 4 describes our experimental work. It provides details of our data sets, evaluation metrics and the results of experiments using proposed models. Finally, Chapter 5 provides a conclusion.

2. DATA PREPARATION AND CLEANING

In this research, we use three sets of server logs. The first one is from the NASA Kennedy Space Center, the second log is from ClarkNet Web server and the last one is from the Web server at the University of Saskatchewan. The details of these data sets are given in Chapter 4. For each log data set we apply the same pre-processing steps[1].

A Web server log is an important source for performing Web usage mining because it explicitly records the browsing behavior of site visitors. The server records the time and date of the transaction. It records the name of the file that was sent and how big that file was. It records the Internet address to which the file was sent. If the user goes to a page by clicking a link on some other page, the server records the address of the page with that link. It also records some details about how the file was sent and any errors that may have occurred as well as information about the browser that the user is using. The data recorded in the server logs reflects the (possibly concurrent) access of a Web site by multiple users. These log files can be stored in various formats such as Common log or Extended log formats. Here is a basic definition of log files (also called extended log files), given by the World Wide Web Consortium (or W3C), the Internet standards group [54]:

An extended log file contains a sequence of lines containing ASCII characters terminated by either the sequence LF or CRLF. Log file generators should follow the line termination convention for the platform on which they are executed. Analyzers should accept either form. Each line may contain either a directive or an entry.

Entries consist of a sequence of fields relating to a single HTTP transaction. Fields are separated by whitespace, the use of tab characters for this purpose is encouraged. If a field is unused in a particular entry dash " - " marks the omitted field.

Basically an entry in Common Log Format consists of 1^o The user's IP address; 2^o The access date and time; 3^o The request method (GET, POST ...); 4^o The URL of the page

[1]Except further cleaning techniques for the "NASA" data set the details of which are given in the next section.

accessed, 5^o the protocol (HTTP 1.0, HTTP 1.1,...); 6^o The return code; and 7^o The number of bytes transmitted. A few lines of a typical access log in the Common Log file format for a sample Web site are presented in Table 2.1.

An extended common log format file is a variant of the common log format file that simply adds two additional fields to the end of the line, the referrer and the user agent fields.

The information provided by the Web server can all be used to construct a data model consisting of several abstractions, notably, users, pages, click-streams, server sessions. In order to provide some consistency in the way these terms are defined, the W3C has published a draft of Web term definitions relevant to analyzing Web usage. A Web user is a single individual who is accessing files from one or more Web servers through a Browser. A page file is the file that is served through a Hypertext Transfer Protocol (HTTP) to a user. The set of page files that contribute to a single display in a Web browser constitutes a Web page. A Browser is a client site software application that interprets Hypertext Markup Language (HTML), the programming language of the Internet, into the words and graphics that the user sees when viewing a Web page. The click-stream is the sequence of pages followed by a user. A server session consists of a set of pages that a user requests from a single Web server during her single visit to that Web site.

In order to understand the user behavior the following information should be extracted from server logs:

- *Who is visiting the Web site?* One of the major steps in Web usage mining is to identify unique users in order to obtain the path that each follows;
- *The path users take through the Web pages.* With knowledge of each page that a user viewed and the order, one can identify how users navigate through the Web pages;
- *How much time users spend on each page?* A pattern of lengthy viewing time on a page might lead one to deduce that the page is interesting;
- *Where visitors are leaving the Web site?* The last page a user viewed before leaving the Web site might be a logical place to end a server session.

Table 2.1: Sample server logs in Common Log Format

Source of Request	**User ID**	**Date and Time of Request**	**Method,URL, (HTTP Protocol)**	**Status Code**	**Num. of Bytes**
216.35.116.28	-	[11/Jan/2002:00:58:25 -0500]	"GET / HTTP/1.0"	200	6557
216.35.116.28	-	[11/Jan/2002:00:58:53 -0500]	"GET a.gif HTTP/1.0"	200	5478
216.35.116.28	-	[11/Jan/2002:00:59:53 -0500]	"GET b.gif HTTP/1.0"	200	6057
216.35.116.28	-	[11/Jan/2002:00:59:54 -0500]	"GET B.html HTTP/1.1"	200	59825
216.35.116.28	-	[11/Jan/2002:00:59:54 -0500]	"GET B.gif HTTP/1.1"	200	2050
24.102.227.6	-	[11/Jan/2002:00:59:55 -0500]	"GET index.html HTTP/1.1"	200	6557
216.35.116.28	-	[11/Jan/2002:00:59:55 -0500]	"GET C.html HTTP/1.1"	200	2560
24.102.227.6	-	[11/Jan/2002:00:59:56 -0500]	"GET a.gif HTTP/1.1"	200	5478
24.102.227.6	-	[11/Jan/2002:00:59:56 -0500]	"GET b.gif HTTP/1.1"	200	6057
24.102.227.6	-	[11/Jan/2002:00:59:57 -0500]	"GET D.HTML HTTP/1.1"	200	12800
24.102.227.6	-	[11/Jan/2002:00:59:58 -0500]	"GET G.gif HTTP/1.1"	200	1500
24.102.227.6	-	[11/Jan/2002:00:59:58 -0500]	"GET e.gif HTTP/1.1"	200	1230
24.102.227.6	-	[11/Jan/2002:00:59:59 -0500]	"GET e.jpg HTTP/1.1"	200	3345
216.35.116.28	-	[11/Jan/2002:00:59:59 -0500]	"GET c.jpg HTTP/1.1"	200	2247
216.35.116.28	-	[11/Jan/2002:01:00:00 -0500]	"GET E.jpg HTTP/1.1"	200	2247
216.35.116.28	-	[11/Jan/2002:01:00:00 -0500]	"GET D.html HTTP/1.1"	200	32768
216.35.116.28	-	[11/Jan/2002:01:00:01 -0500]	"GET D.gif HTTP/1.1"	200	7977
216.35.116.28	-	[11/Jan/2002:01:00:01 -0500]	"GET d.jpg HTTP/1.1"	200	6121
216.35.116.28	-	[11/Jan/2002:01:00:02 -0500]	"GET e.jpg HTTP/1.1"	200	3567
24.102.227.6	-	[11/Jan/2002:01:00:02 -0500]	"GET C.html HTTP/1.1"	200	32768

However, a log file does not contain all of the information required for Web usage mining. Even if it contains other data make it difficult to interpret. Regardless of the application, data preparation and cleaning steps should be completed in order to create server sessions. Data preparation and cleaning tasks performed in this study consist of the following steps: 1^o User Identification; 2^o Session Identification; 3^o Page Time Calculation; and 4^o Data Cleaning. This chapter also presents the methods applied in these steps.

2.1 User Identification

In order to know who is visiting the Web site, the log file must contain a person ID such as login to the server or to the user's own computer. However, most Web sites do not require users to log in, and most Web servers do not make a request to learn the user's login identity on her own computer. Thus, the information available according to the HTTP standard is not adequate to distinguish among users from the same host or proxy. More often it is an IP address assigned by an Internet Service Provider (ISP) or corporate proxy server to a user's TCP/IP connection to the site, preventing unique identification. The most widespread remedy for this problem is the usage of cookies. A cookie is a small piece of code associated with a Web site; it installs itself in the user's host and associates a cookie identifier with user's browser. This identifier is sufficient to recognize the user that launches each URL request, as long as the same browser is being employed. Since cookies are not available in our data sets we use a heuristic method which identifies an unique IP as an user, bearing in mind that a single IP can be used by a group of users. In this step, we remove the information about the size of transmitted files, because we do not need this information in our further cleaning steps. Our system converts a set of server logs expressed as:

$$L = L_1, L_2, ..., L_{|L|}$$

$$L_i = (IP_i, TIME_i, URL_i, PROT_i, CODE_i, BYTES_i)$$

$$L_i \in L,\ i \in [1...|L|]$$

into a set of user transactions T:

$$T = T_1, T_2, ..., T_{|L|}$$

$$T_i = (UID_i, TIME_i, METHOD_i, URL_i, PROT_i, CODE_i)$$

$$T_i \in T,\ i \in [1...|L|]$$

where $|L|$ is the number of logs in L and UID_i is the User Identification Number. For every identical IP address, we assign a unique User Identification Number. Thus, some of the user requests in T have the same User Identification Number.

Table 2.2: Transactions after user and session identification steps

UID	Date-Time	Method, URL, HTTP Protocol	Status Code	SID
1	[11/Jan/2002:00:58:25 -0500]	"GET / HTTP/1.0"	200	1
1	[11/Jan/2002:00:58:53 -0500]	"GET a.gif HTTP/1.0"	200	1
1	[11/Jan/2002:00:58:53 -0500]	"GET b.gif HTTP/1.0"	200	1
1	[11/Jan/2002:00:59:54 -0500]	"GET B.html HTTP/1.1"	200	1
1	[11/Jan/2002:00:59:54 -0500]	"GET B.gif HTTP/1.1"	200	1
1	[11/Jan/2002:00:59:55 -0500]	"GET C.html HTTP/1.1"	200	1
1	[11/Jan/2002:00:59:59 -0500]	"GET c.jpg HTTP/1.1"	200	1
1	[11/Jan/2002:01:00:00 -0500]	"GET E.jpg HTTP/1.1"	200	1
1	[11/Jan/2002:01:00:00 -0500]	"GET D.html HTTP/1.1"	200	1
1	[11/Jan/2002:01:00:01 -0500]	"GET D.gif HTTP/1.1"	200	1
1	[11/Jan/2002:01:00:01 -0500]	"GET d.jpg HTTP/1.1"	200	1
1	[11/Jan/2002:01:00:02 -0500]	"GET e.jpg HTTP/1.1"	200	1
2	[11/Jan/2002:00:59:55 -0500]	"GET index.html HTTP/1.1"	200	1
2	[11/Jan/2002:00:59:56 -0500]	"GET a.gif HTTP/1.1"	200	1
2	[11/Jan/2002:00:59:56 -0500]	"GET b.gif HTTP/1.1"	200	1
2	[11/Jan/2002:00:59:57 -0500]	"GET D.HTML HTTP/1.1"	200	1
2	[11/Jan/2002:00:59:58 -0500]	"GET G.gif HTTP/1.1"	200	1
2	[11/Jan/2002:00:59:58 -0500]	"GET e.gif HTTP/1.1"	200	1
2	[11/Jan/2002:00:59:59 -0500]	"GET e.jpg HTTP/1.1"	200	1
2	[11/Jan/2002:01:00:02 -0500]	"GET C.html HTTP/1.1"	200	1

2.2 Session Identification

Once users have been identified, the click-stream for each user must be divided into sessions. A session can be defined as the time period of an activity from its beginning until its end. The activity may end for a variety of reasons: The user reaches her goal, the user finds the activity not interesting anymore, or there is a time constraint involved. A session has a clean meaning in an online system with user login and logout facilities. A session, in this case, starts from the time when a user performs login, and finishes upon logout.

However, it is different on the Web. According to W3C, a session is the group of activities performed by a user from the moment she enters the site to the moment she leaves it [54]. Due to the fact that there is no official login and logout to access and use most of the Web sites, it is not very clear when a session begins and ends. Since page request from other servers are not typically available and a user may visit a site more than once, the Web server log records multiple sessions for each user [51, 55]. The goal of session identification is to divide the page accesses of each user into individual sessions. The simplest method of achieving this is through a timeout, where if the time between page requests exceeds a certain limit, $\triangle t$, it is assumed that the user is starting a new session. A timeout is the time between two consecutive activities. Catledge and Pitkow found a 25.5 minute timeout on their user experiments in 1994 [55]. In many commercial products this timeout has been rounded up to 30 minutes. In this study we set $\triangle t$ to 30 minutes. A new session is created when a new IP-address is encountered or if the visiting page time exceeds 30 minutes for the same IP-address.

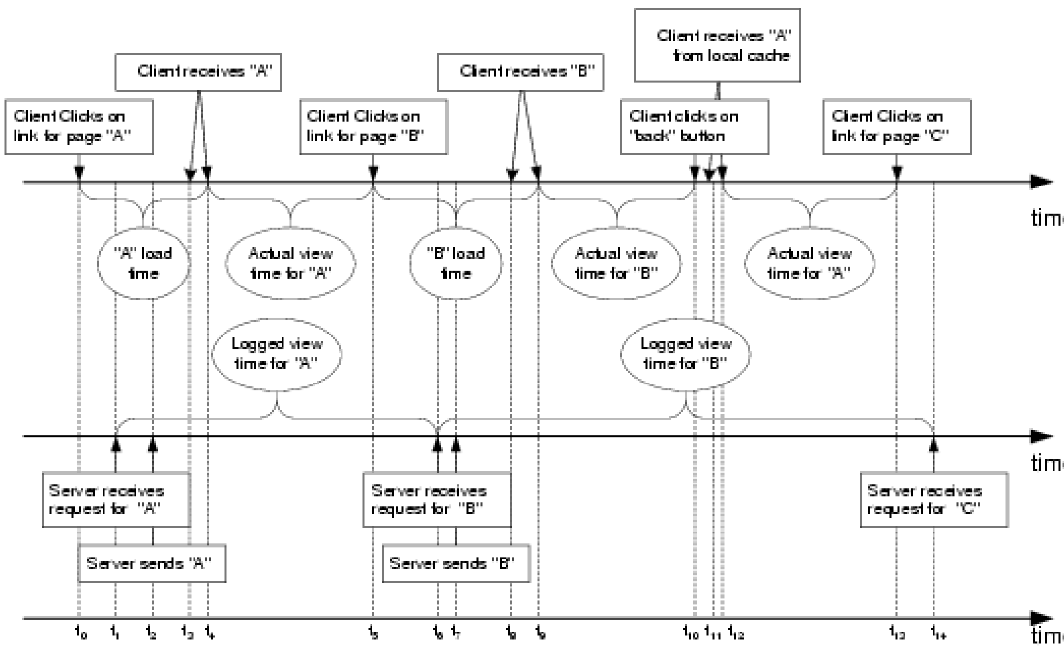
A new field is added to the user transactions created in the previous step. Thus, every user transaction in T takes the form:

$$T_i = (UID_i, TIME_i, METHOD_i, URL_i, PROT_i, CODE_i, SID_i)$$

where SID_i is a unique session identification number given every time a new session is created for the same user. Table 2.2 shows user transactions with UID and SID extracted from the server logs in Table 2.1.

2.3 Page Time Calculation

In this step, we calculate visiting page time for each page which we define as the time difference between consecutive page requests for a user in the same session. This measurement is subject to a fair amount of noise as the user's behavior can not always be accurately determined, e.g., the user could be taking a cup of coffee, talking on the phone, or accurately reading the page. However, we can assume that such interruptions during visiting a Web page are in minority. Since it is impossible to determine the accurate behavior of a Web user these interruptions could appear as outliers in the data set. With sufficient size of the data set the common behavior of Web users in terms of visiting page times could be determined in spite of these outliers. However, if the size of the data set is insufficient for determining the common behavior of Web users the performance of the model could be low.



Client Clicks on link for page "A"
Client receives "A"
Client Clicks on link for page "B"
Client receives "B"
Client receives "A" from local cache
Client clicks on "back" button
Client Clicks on link for page "C"
time
"A" load time
Actual view time for "A"
"B" load time
Actual view time for "B"
Actual view time for "A"
Logged view time for "A"
Logged view time for "B"
time
Server receives request for "A"
Server sends "A"
Server receives request for "B"
Server sends "B"
Server receives request for "C"
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14}
time

Figure 2.1: Timeline for page file request and response

Another problem is that the view time of a page as recorded in a server log is often longer than the actual view time on the client side. As shown in Figure 2.1 [51], the time between requests for pages A and B is $t_6 - t_1$, but the actual view time for page A is only $t_5 - t_4$. A page can consist of several files such as frames, graphics and scripts. The user requests a Web page, but she does not explicitly ask for frames or graphics to be loaded into her browser. Depending on the connection speed of the client and the size of the page file, the difference in recorded and actual view time may range from a second to several minutes. To reduce this side effect we apply a heuristic method during the calculation of the visiting page time. We calculate the visiting page time of a HTML page as the time difference between the last non-HTML page that has been sent after the HTML page request and the next HTML page that has been requested by the user. Figure 2.2 presents the algorithm for this calculation. For example, suppose that the user requested first a HTML page A at time t_0 as in Figure 2.1. A new session is opened for that user and the end time for that session is set to t_0 (line 15-16 in Figure 2.2). The time stamp for the last non-HTML page after A is t_4 (line 21). When the user requests a HTML page B at time t_6 (line 5), the visiting time for page A will be calculated as $t_6 - t_4$ (line 8). For the last page of the user session, we set the page time to be the mean of visiting page times for the page taken across all sessions in which the page is not the last page request. Table 2.3 presents visiting page times calculated for the HTML pages requested by the user in the sample log data in Table 2.1. Since there are only two user sessions, the visiting time for the last page of each session is set to the visiting time of that page in the other session.

Input : $T,\ \triangle t$
Output : $T\ with\ page\ visiting\ time$

```
1:  Sort $T$ by $UID$ and $SID$
2:  **for all** unique $UID_i$ and $SID_i$ pair **do**
3:    $OPEN\ SESSION = \{\emptyset\}$
4:    **for all** $T_j$ with $UID_i$ and $SID_i$ **do**
5:      **if** $URL_j$ is HTML page **then**
6:        **if** $\exists S_k \in OPEN\ SESSION$ **then**
7:          **if** $TIME_j - END\,TIME(S_k) \leq \triangle t$ **then**
8:            $visiting\,time(URL_j) = TIME_j - END\,TIME(S_k)$
9:          **else**
10:           close $S_k$
11:           Open new session $S_k$
12:           $END\,TIME(S_k) = TIME_j$
13:         **end if**
14:       **else**
15:         Open new session $S_k$
16:         $END\,TIME(S_k) = TIME_j$
17:       **end if**
18:     **else**
19:       **if** $\exists S_k \in OPEN\ SESSION$ **then**
20:         **if** $TIME_j - END\,TIME(S_k) \leq \triangle t$ **then**
21:           $END\,TIME(S_k) = TIME_j$
22:         **else**
23:           close $S_k$
24:           Open new session $S_k$
25:           $END\,TIME(S_k) = TIME_j$
26:         **end if**
27:       **else**
28:         Open new session $S_k$
29:         $END\,TIME(S_k) = TIME_j$
30:       **end if**
31:     **end if**
32:   **end for**
33: **end for**
```

Figure 2.2: Algorithm for calculating visiting page times

Table 2.3: Visiting and normalized page times for HTML pages

UID	Method, URL HTTP Protocol	visiting time (sec.)	normalized time	Status code	SID
1	"GET / HTTP/1.0"	29	10	200	1
1	"GET B.html HTTP/1.1"	1	1	200	1
1	"GET C.html HTTP/1.1"	0	1	200	1
1	"GET D.html HTTP/1.1"	3	2	200	1
2	"GET index.html HTTP/1.1"	1	4	200	1
2	"GET D.HTML HTTP/1.1"	3	10	200	1
2	"GET C.html HTTP/1.1"	0	1	200	1

However, the raw time durations may not be an appropriate measure for the interest of a user in that page. This is because a variety of factors, such as structure, length, and the type of Web page, the speed of network connection, as well as the user's interests in a particular item, may affect the amount of time spent on that page. Appropriate normalization of the time can play an essential role in correcting for these factors. Since we want to capture the relative importance of a page to a particular user relative to other pages visited by that user in the same session, we normalized the visiting times across the visiting times of pages in the same session S_k:

$$\begin{aligned} norm_{URL_i} &= [(TIME_i - min(T(S_k)))/(max(T(S_k)) - min_(T(S_k)))] \\ &* \; (max(norm) - min(norm)) \; + \; min(norm) \end{aligned} \tag{2.1}$$

where $max(T(S_k))$ and $min(T(S_k))$ are the maximum and minimum visiting page times respectively taken across the visiting page times spent by a user in the same session S_k. $max(norm)$ and $min(norm)$ are the maximum and minimum values for normalized page times respectively. For evaluating the effect of the normalization values, we try 4 different maximum values: 2, 3, 5 and 10. The minimum value of normalized time is set to 1 in order to differentiate the existence or non-existence of a page in a session. Table 2.3 shows the normalized time of pages in the sample log file. The maximum value for the normalized times in this case is 10. At the end of this step, the log entries in the data sets are converted to the form:

$$T_i = (UID_i, norm_{URL_i}, METHOD_i, URL_i, PROT_i, CODE_i, SID_i)$$

2.4 Data Cleaning

In this step several filtering methods are applied in order to remove irrelevant log entries. A user's request to view a particular page often results in several log entries since graphics and scripts are downloaded in addition to the HTML file [51]. Since the main intent of Web Usage Mining is to extract a pattern from the user's behavior, it does not make sense to include file requests that the user did not explicitly request. All log entries with filename suffixes such as `gif`, `GIF`, `jpeg`, `JPEG`, `jpg`, `JPG`, and maps are removed. The page requests made by the automated agents and spider programs traversing links can often cause to a skewed analysis. The simplest method for dealing with agent traffic is to check the agent field of the usage data. A simple string match during the data cleaning step can remove a significant amount of agent traffic. Robots that follow the conventions of [56] will check for the existence of a file named "`robot.txt`". An agent can be identified through the agent field or by the requesting "`robot.txt`" file. The log entries that are made by an agent are removed from the log data.

The status code is returned by the server as a response to the user request. The status codes of 400 series means a failure and 500 series means that there is a server error. For that reason, the log entries that have a status code of 400 and 500 series are removed. The next step is extracting HTML pages and removing any CGI (Common Gateway Interface) data. Web servers that implement the CGI standard parse the URL of the requested file to determine if it is an application program. The URL for CGI programs may contain additional parameters to be passed to the CGI application. If the request is made through a hidden "`POST`" method, the parameters are not available in the log data saved by common log format. Since our data are in common log format, hidden CGI parameters are not logged. Thus the data stored in these log files are cleaned such that only URL page requests of the form "`GET ...HTML`" are maintained.

The next step of the data cleaning task is to normalize the URLs in the log file. Most Web servers treat a requests for a directory as a request for a default file such as "`index.html`" or "`home.html`". The directory request may come in with or without a trailing slash. This means request for "`www.sample`", "`www.sample/`", "`www.sample/index.html`", "`www.sample/home.html`" are all for the same file. The data cleaning must choose a common form for each Web page. This can be done using a Web crawler. Web crawlers start by parsing a specified Web page, noting any hypertext links on that page that point to other Web pages. They then parse those pages for new links, and so on, recursively. The crawler simply sends HTTP requests for documents to Web server, just as a Web browser does when the user clicks on links. All the crawler really does is to automate the process of following links. A Web crawler is implemented for this study which retrieves the content of the Web pages as well. Only links that point to the HTML pages within the site are added to the list of pages to explore. Comparing the content of pages provides us with a way to determine different URLs belonging to the same Web page. The Web crawler produces a representation of the Web site where each Web page has the following information:

- *URL*;
- *children*, which are the Web pages that are pointed by the page;
- *inlinks*, which is the number of links that point to the page;
- *outlinks*, which is the number of links the page contains to other Web pages;
- *size*, in bytes;
- *time*, that the page was last modified;
- *Top-10 words*, which is a list consisting of ten words that appear most frequently in the Web page.

However ClarkNet Web site and the Web site of University of Saskatchewan can not be explored using a Web crawler, because the ClarkNet Web server does not exist anymore, and the Web server of the University of Saskatchewan is not up-to-date. For these data sets, we apply simple string matching for identifying unique Web pages. Even some URLs in the NASA log are not up-to-date such that the Web crawler is unable to find these pages. After finding some pages with the Web crawler, we apply string matching to the NASA log as well.

In the last step of the data cleaning some filtering methods are applied. The navigation pages that provide links to guide users to the content pages are determined. This is an easy task if the number of outlinks from each page could be counted. The pages that have outlinks more than a predefined threshold can be determined as navigation pages. For the Web sites on which the Web crawler can not be used, the number of requests for each page can be counted. If a page is requested more than the average number of the requests, it is more likely to be a navigation page or home page. For consistency of filtering methods for the data sets, we do not determine the navigation pages on NASA Web site using the Web crawler. The number of requests are counted in each data set. This process shows that the page requests are very scattered, i.e. even the most popular pages such as home pages are requested in about 10% of the sessions. Since our objective in this study is to recommend pages that contain a portion of the information content that the Web site provides, the Web pages that appear in more than 10% of the sessions are eliminated from the data sets. The intuition behind this is that the system should recommend pages that the user may find interesting. Furthermore, the recommendation should guide the user during her visit to the Web site. However, recommending a navigation page or a home page does not help the user in that context. If a user requests a navigation page, this can be interpreted as the recommendation made by the system is unsuccessful. It is not reasonable to recommend a page to a user when she first enters the home page of a Web site.

We apply FP-tree algorithm [57] for discovering pages that are frequently requested together. Some of the page views appear together in less than 1% of sessions in the entire data set if the Web site has a complex structure. A learning algorithm for predicting the next request of the user will learn not to recommend the pages that appear together with a low request frequency. Thus, reducing the dimensionality of the input data by removing less frequent page requests at the beginning of the learning algorithm makes it efficient. On the other hand, using frequent patterns as a filter for eliminating pages covers simple non-personalized recommendation such that "users who visit page A also visit page B". Pages that appear together in more than 1% of all sessions are used for recommendation in order to capture the relationship between page requests. These filtering steps produce a set of URL's

$$P = \{p_1, ..., p_n\}$$

where each URL has a unique code p_i. The pages that are not in the set of P are removed from the user sessions. Finally, a filtering method is applied in order to remove sessions whose length is less than 4 or longer than twice the average length of the sessions in order to eliminate the noise that random accesses or search engines would introduce to the model. Since the user identifications are not used in the subsequent examinations, they are removed from the log entries. The output of this step is a set of user sessions S:

$$S = \{S_1, S_2, ..., S_{|S|}\}$$

where $|S|$ is the number of sessions of S. Each session of the set of sessions S is defined by a tuple $PAGES,\ NORMS$:

$$S_i = (PAGES_i,\ NORMS_i)$$

where $PAGES_i$ is a subset of P that the user visits in her session S_i and $NORMS_i$ is the normalized visiting times of pages in P:

$$PAGES_i = \{p_{i1}, ..., p_{ik}\} \quad i \in [1, ..., |S|] \wedge p_{i,k} \subset P$$

$$NORMS_i = \{norm_{p_1}, ..., norm_{p_n}\}$$

where $norm_{p_j}$ is the normalized visiting time of page p_j if $p_j \in PAGES_i$ or 0 otherwise. After these cleaning steps, the user sessions for the sample log data are shown in Table 2.4. For simplicity, the intermediate steps of the cleaning process, such as frequent item set mining, are not presented in the example. In the sample log data there are only 4 pages, namely $P = \{p_1, p_2, p_3, p_4\}$. As can be seen from Table 2.4 the page requests in line 1 and line 5 in Table 2.3 are assigned to the same page number, p_1. The $PAGES$ field of the user sessions corresponds to the pages that the user visits in her session whereas the $NORMS$ field corresponds to the normalized time values for all pages in the Web site calculated for that user.

Table 2.4: User Sessions for sample log data

S	PAGES	NORMS
1	$\{p_1,\ p_2,\ p_3,\ p_4\}$	{10, 1, 1, 2}
2	$\{p_1,\ p_4,\ p_3\}$	{4, 0, 1, 10}

3. WEB PAGE NAVIGATION MODELS

The process of a recommendation system was shown in Figure 1.1 in Chapter 1. The first two components of this process is implemented using the methods which were detailed in Chapter 2. For the last two components of the recommendation process we propose two new models, namely:

- ***User Interest Model***
- ***Click-Stream Tree Model***

The User Interest Model (UIM) uses only the visiting time and visiting frequencies of pages without considering the access order of page requests in user session. On the other hand, Click-Stream Tree Model (CSTM) uses three kinds of information: 1^o Access order of Web pages; 2^o Visiting time of Web pages; and 3^o The distance between visited Web pages in a session. The main difference between these two models is the time needed for online recommendation and the accuracy tradeoff. The common characteristic of the two models is that the discovered patterns do not depend on any personal data about the site users. Each section in this chapter presents a model. First, a brief background information of the related model is given in each section. Next, the details of the proposed models are discussed.

3.1 User Interest Model

Making a recommendation requires predicting what is of interest to a user at a specific time. Even the same user may have different desires at different times. It is important to extract the aggregate interest of a user from her navigational path through the site in a session. This chapter concentrates on the discovery and modelling of the user's aggregate interest in a session.

3.1.1 Background

The UIM relies on the premise that the visiting time of a page is an indicator of the user's interest in that page. The proportion of times spent in a set of pages requested by the user within a single session forms the aggregate interest of that user in that session. We first partition user sessions into clusters such that only sessions which represent similar aggregate interest of users are placed in the same cluster. The key idea behind this work is that user sessions can be clustered according to the similar amount of time that is spent on common pages among user sessions. In particular, we model user sessions in log data as being generated in the following manner: 1^o When a user arrives to the Web site, her current session is assigned to one of the clusters; 2^o The behavior of that user in this session, in terms of visiting time, is then generated from a Poisson model of visiting times of that cluster. Since we do not have the actual cluster assignments, we use a standard learning algorithm, the *Expectation-Maximization* (EM) algorithm [58], to learn the cluster assignments of sessions as well as the parameters of each Poisson distribution. The resulting clusters consist of sessions in which users have similar interests and each cluster has its own parameters representing these interests.

The next page request of an active user is predicted using parameters of the cluster to which the active user is assigned. The model produces a set of recommendations based on this prediction. The detailed model is given in this chapter.

Model-Based Cluster Analysis

Model-based clustering methods optimize the fit between the given data and some mathematical model. Such methods are often based on the assumption that the data are generated by a mixture of underlying probability distributions [59]. Given a data set of K observations $\mathbf{D} = \{\mathbf{x}_1, ..., \mathbf{x}_K\}$, every observation $\mathbf{x}_i, (i \in [1, ..., K])$ is generated according to a probability distribution defined by a set of parameters, denoted $\mathbf{\Theta}$. The probability distribution consists of a mixture model of components $c_j \in C = \{c_1, c_2, ..., c_G\}$. The parameters of each component, $\mathbf{\Theta}_g$, is a disjoint subset of $\mathbf{\Theta}$, where $\mathbf{\Theta}_g$ $(g \in [1...G])$ is a vector specifying the probability distribution function (pdf) of the g^{th} component. An observation, $\mathbf{x}_i$, is created by first selecting a mixture component according to the mixture weights (or cluster prior probabilities), $p(c_g|\mathbf{\Theta}) = \tau_g$, where $\sum_{g=1}^{G} \tau_g = 1$, then having this selected mixture component generate an observation according to its own parameters, with distribution $p(\mathbf{x}_i|c_g;\boldsymbol{\Theta}_g)$. Thus, the likelihood of a data point, $\mathbf{x}_i$, can be characterized with a sum of total probabilities over all mixture components:

$$p(\mathbf{x}_i|\boldsymbol{\Theta}) = \sum_{g=1}^{G} p(c_g|\boldsymbol{\Theta})p(\mathbf{x}_i|c_g,\boldsymbol{\Theta}_g) = \sum_{g=1}^{G} \tau_g p(\mathbf{x}_i|c_g,\boldsymbol{\Theta}_g) \tag{3.1}$$

Statisticians refer to such a model as *mixture model with G components*. Thus, the model-based clustering problem consists of finding the model, i.e. the model structure and parameters for that structure that best fit the data. The parameters are chosen in two ways. The *maximum likelihood* (ML estimation) approach maximizes:

$$\ell_{ML}(\boldsymbol{\Theta}_1,...,\boldsymbol{\Theta}_G;\tau_1,...,\tau_G|D) = \prod_{i=1}^{K}\sum_{g=1}^{G} \tau_g p(\mathbf{x}_i|c_g,\boldsymbol{\Theta}_g) \tag{3.2}$$

The second approach, Maximum Aposteriori (MAP estimation), maximizes the *posterior probability* of $\boldsymbol{\Theta}$ given the data:

$$\ell_{MAP}(\boldsymbol{\Theta}_1,...,\boldsymbol{\Theta}_G;\tau_1,...,\tau_G|D) = \prod_{i=1}^{K}\sum_{g=1}^{G} \frac{\tau_g p(\mathbf{x}_i|c_g,\boldsymbol{\Theta}_g)p(\boldsymbol{\Theta})}{p(D)} \tag{3.3}$$

The term $p(D)$ can be ignored in Equation 3.3, since it is not a function of $\boldsymbol{\Theta}$.

In practice, the log of these expressions is often used. Thus, the log likelihood of Equation 3.2 and 3.3 are respectively:

$$L(\boldsymbol{\Theta}_1,...,\boldsymbol{\Theta}_G;\tau_1,...,\tau_G|D) = \sum_{i=1}^{K}\ln\left(\sum_{g=1}^{G} \tau_g p(\mathbf{x}_i|c_g,\boldsymbol{\Theta}_g)\right) \tag{3.4}$$

$$L(\boldsymbol{\Theta}_1,...,\boldsymbol{\Theta}_G;\tau_1,...,\tau_G|D) = \sum_{i=1}^{K}\ln\left(\sum_{g=1}^{G} \tau_g p(\mathbf{x}_i|c_g,\boldsymbol{\Theta}_g)\right) + \ln p(\boldsymbol{\Theta}) \tag{3.5}$$

The set of parameters of the model ($\boldsymbol{\Theta}$) include mixture weights representing cluster prior probabilities (τ_g), which indicate the probability of selecting different mixture components and the set of the parameters of the probability distribution assumed for the data:

$$\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_1,...,\boldsymbol{\Theta}_G,\tau_1,...,\tau_G\},\ \sum_{g=1}^{G}\tau_g = 1 \tag{3.6}$$

EM Algorithm for Clustering

The model parameters can be trained using the *Expectation Maximization* (EM) algorithm. The EM algorithm is a very general iterative algorithm for parameter estimation by maximum likelihood when some of the random variables involved are not observed (i.e., considered missing or incomplete). In the Expectation step (E-step), the values of the unobserved variables are essentially “filled in”, where the filling-in is achieved by calculating the probability of the missing variables, given the observed variables and the current values of parameters. In the Maximization step (M-step), the parameters are adjusted based on the filled-in variables [59].

Let $D = \{\mathbf{x}_1, ..., \mathbf{x}_K\}$ be a set of K observed variables, and $H = \{\mathbf{z}_1, ..., \mathbf{z}_K\}$ represent a set of K values of hidden variables Z, such that each $\mathbf{z}_i$ is in the form of $\mathbf{z}_i = \{z_{1i}, ..., z_{Gi}\}$ and corresponds to a data point $\mathbf{x}_i$. It can be assumed that Z is discrete and represents the class (or cluster) labels for the data with the following possible values:

$$z_{ji} = \begin{cases} 1 & \text{if} \quad \mathbf{x}_i \text{ belongs to cluster } j; \\ 0 & \quad\quad \text{otherwise}. \end{cases}$$

If Z could be observed, then the ML estimation problem would be based on the maximization of the quantity:

$$L_c(\mathbf{\Theta}; D, H) \triangleq \ln p(D, H|\mathbf{\Theta}) \tag{3.7}$$

In the presence of missing data, we calculate conditional expectation of the complete data likelihood given the observed data and the current parameter estimate as follows:

$$Q(\mathbf{\Theta}, \mathbf{\Theta}') = E\left[L_c(D, H|\mathbf{\Theta})|D, \mathbf{\Theta}'\right] \tag{3.8}$$

where the term $L_c(D, H|\mathbf{\Theta})$ is:

$$L_c(D, H|\mathbf{\Theta}) = \sum_{i=1}^{K} \ln p(\mathbf{x}_i, \mathbf{z}_i|\mathbf{\Theta}) \tag{3.9}$$

Equation 3.8 involves $\mathbf{\Theta}$, which is the parameter of the complete likelihood and $\mathbf{\Theta}'$, which is the parameter of the conditional distribution of complete data.

The Q-function in Equation 3.8 can be expanded as follows:

$$\begin{aligned}
E\left[L_c(D,H|\mathbf{\Theta})|D,\mathbf{\Theta}'\right] &= E\left[\sum_{i=1}^{K}\ln p(\mathbf{x}_i,\mathbf{z}_i|\mathbf{\Theta})|D,\mathbf{\Theta}'\right] \\
&= \sum_{l=1}^{G}\sum_{i=1}^{K}\ln p(\mathbf{x}_i,\mathbf{z}_i|\mathbf{\Theta})\prod_{j=1}^{K}p(z_{lj}|\mathbf{x}_j,\mathbf{\Theta}') \\
&= \sum_{i=1}^{K}\sum_{l=1}^{G}\left(\ln p(\mathbf{x}_i,\mathbf{z}_i|\mathbf{\Theta})p(z_{li}|\mathbf{x}_i,\mathbf{\Theta}')\right)\prod_{j\neq i}\sum_{l=1}^{G}p(z_{lj}|\mathbf{x}_j,\mathbf{\Theta}') \\
&= \sum_{i=1}^{K}\sum_{l=1}^{G}\ln p(\mathbf{x}_i,\mathbf{z}_i|\mathbf{\Theta})p(z_{li}|\mathbf{x}_i,\mathbf{\Theta}') \\
&= \sum_{i=1}^{K}\sum_{\mathbf{z}_i}\ln p(\mathbf{x}_i,\mathbf{z}_i|\mathbf{\Theta})p(\mathbf{z}_i|\mathbf{x}_i,\mathbf{\Theta}') \\
&= \sum_{i=1}^{K}\sum_{\mathbf{z}_i}p(\mathbf{z}_i|\mathbf{x}_i,\mathbf{\Theta}')\ln\left[p(\mathbf{x}_i|\mathbf{z}_i,\mathbf{\Theta})p(\mathbf{z}_i|\mathbf{\Theta})\right] \\
&= \sum_{i=1}^{K}\sum_{\mathbf{z}_i}p(\mathbf{z}_i|\mathbf{x}_i,\mathbf{\Theta}')\left[\ln p(\mathbf{x}_i|\mathbf{z}_i,\mathbf{\Theta})+\ln p(\mathbf{z}_i|\mathbf{\Theta})\right] \quad (3.10)
\end{aligned}$$

At each EM iteration, the Q-function is maximized with respect to the parameters $\mathbf{\Theta}$ using the current parameters $\mathbf{\Theta}'$. At the end of each iteration, a set of new optimal parameters $\mathbf{\Theta}$ becomes the current parameters $\mathbf{\Theta}'$ for the next iteration. Given these steps, the EM algorithm can be implemented as follows:

1. Choose an initial estimate for parameter set $\mathbf{\Theta}'(0)$, and set $n = 0$.
2. **(E)xpectation Step:** For n, compute $Q(\mathbf{\Theta},\mathbf{\Theta}'(n))$ using Equation 3.8.
3. **(M)aximization Step:** Replace the current estimate $\mathbf{\Theta}'(n)$ with the new estimate $\mathbf{\Theta}'(n+1)$ where,

$$\mathbf{\Theta}'(n+1) = argmax_{\mathbf{\Theta}}Q(\mathbf{\Theta},\mathbf{\Theta}'(n))$$

4. Set $n = n+1$ and iterate steps 2 and 3 until convergence.

By iteratively applying the E-step and M-step, the parameters $\mathbf{\Theta}$ will converge to at least a local maximum of the log likelihood function.

Table 3.1: A set of user sessions as running example

S	$PAGES$	$NORMS$
1	$\{p_1, p_3, p_6, p_5, p_{10}\}$	{1, 0, 8, 0, 3, 10, 0, 0, 0, 1}
2	$\{p_4, p_9, p_6, p_{10}, p_7\}$	{0, 0, 0, 2, 0, 1, 10, 0, 8, 10}
3	$\{p_7, p_6, p_5, p_4, p_{10}, p_9\}$	{0, 0, 0, 2, 1, 1, 9, 0, 8, 10}
4	$\{p_1, p_3, p_6, p_5, p_{10}, p_9, p_2\}$	{10, 10, 3, 0, 1, 6, 0, 0, 1, 4}
5	$\{p_1, p_{10}, p_8, p_2, p_5, p_3, p_6\}$	{1, 1, 7, 0, 3, 10, 0, 1, 0, 1}
6	$\{p_9, p_{10}, p_6, p_3, p_2, p_1\}$	{10, 9, 2, 0, 0, 6, 0, 0, 1, 4}
7	$\{p_1, p_5, p_6, p_3\}$	{1, 0, 8, 0, 4, 10, 0, 0, 0, 0}

3.1.2 Model Implementation

Once the data cleaning and preprocessing tasks described in the Chapter 2 are performed, Web server logs will have been converted into a set of user sessions. User sessions can be clustered according to the similar amount of time spent on common pages.

Example 3.1. A sample set of user sessions, for a Web site with ten pages, $P = \{p_1, p_2, ..., p_{10}\}$ is shown in Table 3.1. $PAGES$ corresponds to a subset of pages in P and $NORMS$ corresponds to the normalized visiting times of pages in P.

Clustering User Sessions in Web Log Data

In this section, we first describe the specific mixture model that we use for clustering the user sessions in Web log data. Next, the update parameters for training the mixture model of Poisson distributions with the Expectation Maximization algorithm are given. We use a model-based technique to group the user sessions according to the interests of users in each session. We assume the data to be generated in the following fashion:

1. When a user arrives at a Web site, her session is assigned to one of G clusters with some probability.
2. Given that a user's session is in a cluster, her next request in that session is generated according to a probability distribution specific to that cluster.

Since it is assumed that the data are produced by a mixture model, every user session is generated according to the probability distribution defined by a subset of model parameters, denoted $\mathbf{\Theta}_g$. Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_K\}$ be a set of K user sessions and $\mathbf{C}$ be a discrete valued variable taking values $c_1, ..., c_G$, which corresponds to an unknown cluster assignment of a user session. Then the mixture model for a user session is:

$$\begin{aligned} p(\mathbf{X} = \mathbf{x}_i|\mathbf{\Theta}) &= \sum_{g=1}^{G} p(\mathbf{C} = c_g|\mathbf{\Theta}) p(\mathbf{X} = \mathbf{x}_i|\mathbf{C} = c_g, \mathbf{\Theta}_g) \\ &= \sum_{g=1}^{G} \tau_g p(\mathbf{X} = \mathbf{x}_i|c_g, \mathbf{\Theta}_g) \end{aligned} \tag{3.11}$$

where τ_g is the probability of selecting cluster c_g. A user session, $\mathbf{x}_i$, is considered to be an $n-$dimensional vector of visiting page times, $(x_{i1}, x_{i2}, ..., x_{in})$, where x_{ij} corresponds to $norm_{p_i}$ in $NORMS$ field of a user session; each p_j is a page in the set of pages (in a given site) $P = \{p_1, p_2, ..., p_n\}$. Each page in the set of pages P corresponds to a dimension in the model. The n-dimensional vector represents the aggregate interest of the user.

In our case, the mixture model can be regarded as a distribution in which the class labels are missing. Although we reduce the dimensions of the input data using frequent pattern mining, there is still a problem of how to estimate the probabilities. One of the key ideas to handle this problem is to impose a structure on the underlying distribution, for example by assuming the independence of dimensions:

$$p(\mathbf{x}_i) = \prod_{j=1}^{n} p_j(x_{ij}) \tag{3.12}$$

Since a user session is an n-dimensional vector of normalized visiting times, we can easily adapt this assumption to our model. Even the order of visiting pages may be different in two user sessions, each session can be represented by the equal vectors if the normalized page times corresponding to the same page in each session are equal.

Example 3.2. To illustrate the independence assumption for our model, consider the user sessions 1, 4 and 7 in Table 3.1. The order of page requests in sessions 1 and 7 are different. However, the aggregate interests of these sessions are very similar, because the normalized page times of each page are similar. Although the first 5 pages in sessions 1 and 4 are requested in the same order, the aggregate interests of these sessions are not similar. According to our clustering criteria, sessions 1 and 7 would be in the same cluster, whereas session 4 would be in a different cluster. Thus, the value of the m^{th} dimension of a session, where $m \in [1...n]$, is independent of the values in the preceding dimensions.

The independence assumption enables us to use n separate probability distributions to model each dimension of a user session. To model this data, we assume that the data at each dimension have been generated by a mixture of Poisson distributions. A random variable X has a *Poisson distribution* with parameter m if for some $m > 0$ [60]:

$$p(X = k) = \frac{m^k\, e^{-m}}{k!} \qquad k = 0, 1, ... \tag{3.13}$$

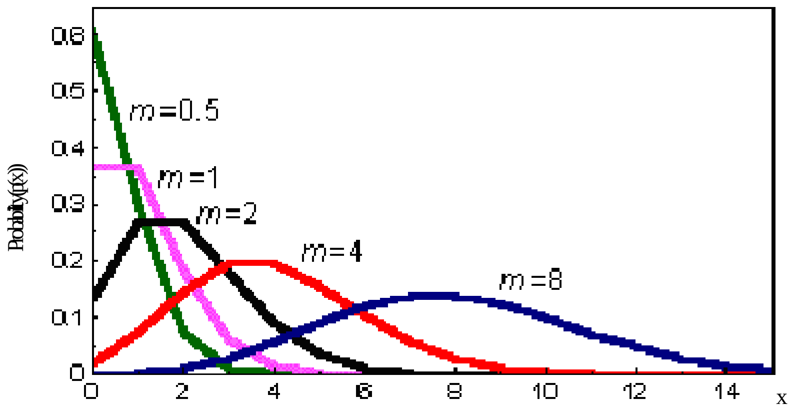


Figure 3.1: Shape of the Poisson distribution for different parameters

Figure 3.1 presents the shape of the Poisson distribution for different parameters, m. As m increases, the shape of the Poisson distribution begins to resemble a bell shaped distribution. The Poisson model can be used to model the rate at which individual events occur, for example the rate at which a user session has the value 1 for a particular page. To confirm our assumption, that the data in each dimension have been generated by a Poisson distribution, the histogram of the occurrence of each of the ten possible values at each dimension has been plotted. Most of the histograms verify our assumption. Figure 3.2 presents one of these histograms. As can be seen, the histogram has the shape of the Poisson distribution with a low parameter m.

According to the independence assumption, a user session $\mathbf{x}_i$ is generated in a cluster g by a Poisson model as follows:

$$p(\mathbf{x}_i | c_g, \mathbf{\Theta}_g) = \prod_{j=1}^{n} \frac{(\theta_{gj})^{x_{ij}}\, e^{-\theta_{gj}}}{x_{ij}!} \tag{3.14}$$

where θ_{gj} is the parameter of the Poisson distribution for a dimension j in cluster g.

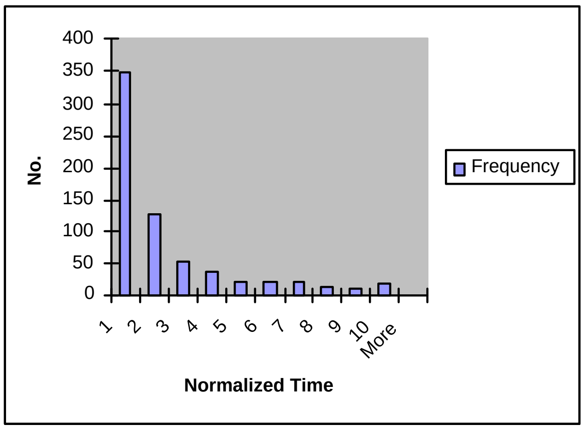


Figure 3.2: Histogram of a page from NASA Web server

By combining Equation 3.11 and Equation 3.14 we obtain:

$$p(\mathbf{x}_i|\mathbf{\Theta}) = \sum_{g=1}^{G} \tau_g \left(\prod_{j=1}^{n} \frac{(\theta_{gj})^{x_{ij}}\, e^{-\theta_{gj}}}{x_{ij}!} \right) \tag{3.15}$$

where θ_{gj} $(g \in [1...G], j \in [1...n])$ is the Poisson parameter of cluster c_g at dimension j.

Example 3.3. For the sample set of user sessions in Table 3.1, there are 10 Poisson parameters for each cluster, where the number of unique pages is 10 in that data set.

The model parameters to be learned are then:

$$\mathbf{\Theta} = \{\mathbf{\Theta}_1, ..., \mathbf{\Theta}_G, \tau_1, ..., \tau_G\},\ \mathbf{\Theta}_g = (\theta_{g1}, ..., \theta_{gn})\,,\ \sum_{g=1}^{G} \tau_g = 1 \tag{3.16}$$

Learning the Model Parameters

We can train the model parameters of the mixture model, developed in the previous subsection, using EM algorithm where the conditional independence assumption is enforced during Maximization step. The learning algorithm is carried out for each component of the model. There are several reasons for using the EM algorithm:

- We want to represent the behavior of the user in one session using Poisson distribution;
- Its performance is linear to the number of sessions;

- It is robust to noisy data;
- It accepts as input the desired number of clusters;
- It provides a cluster membership probability per session;
- It can handle high dimensionality;
- It converges fast given a good initialization.

In order to implement the EM algorithm we should pick the number of clusters (G), an initial starting point ($\mathbf{\Theta}'(0)$), a convergence criteria and prior probabilities for $\mathbf{\Theta}$ in case of MAP estimate of model parameters. To determine the number of clusters, we run the algorithm with several numbers of clusters. We initialize the parameters of our components, $\mathbf{\Theta}_g,\ (g \in [1...G])$ by estimating the Poisson parameters for a single component model and then randomly perturbing the parameter values by a small amount to obtain G sets of parameters. We determine the convergence criteria such that the algorithm converges when the log likelihoods of two consecutive iterations on the training data differ less than 0.001%. There is a trade-off between the estimation accuracy of parameters and the number of iterations. With a smaller value the number of iterations required for convergence will increase so that the algorithm converges in a longer period of time. If it is greater than the selected value, then the estimation for the parameters would be less precise. Finally, to assign prior probabilities to $\mathbf{\Theta}$ for MAP estimate we use a prior distribution for the Poisson distribution.

Example 3.4. Let us determine the initial parameters of the EM algorithm for the data set in Table 3.1. Assume that the number of clusters is 3 and the cluster prior probabilities are set to $\tau_g = 1/3$ where $g \in\ [1, 2, 3]$. We determine 10 initial values for Poisson parameters for each cluster, giving a total of 30 Poisson parameters for the model, as mentioned earlier. Table 3.2 presents these parameters. Then, for the first iteration of the EM algorithm the cluster prior of the first cluster would be $\tau_1 = 1/3$, and the Poisson parameter of the second dimension in that cluster is $\theta_{11} = 0.2207$.

ML Estimate of Model Parameters One approach to learning parameters from data is to find those parameter values that maximize the likelihood of data:

$$\mathbf{\Theta}^{ML} = argmax_{\mathbf{\Theta}}\{p(D|\mathbf{\Theta}_1, ..., \mathbf{\Theta}_G, \tau_1, ..., \tau_G)\} \tag{3.17}$$

Table 3.2: Poisson parameters for three clusters

page number	cluster 1	cluster 2	cluster 3
10	0.1917	0.2030	0.2049
9	0.2126	0.2380	0.2095
8	1.0020	1.0067	0.9942
7	0.0954	0.0923	0.1111
6	0.1395	0.1548	0.1623
5	0.4321	0.4011	0.4178
4	0.4864	0.4824	0.4972
3	0.1711	0.1685	0.1728
2	0.1464	0.1377	0.1481
1	0.2204	0.2207	0.1996

These parameters are often referred to as a maximum likelihood or ML estimate.

The E-step of ML estimate of parameters involves an update of the conditional probability of missing class labels given the current parameter set $\mathbf{\Theta}'$. We define this probability as *cluster-posterior* probability, $P_{ig}(\mathbf{\Theta}')$, that the transaction $\mathbf{x}_i$ arose from the g^{th} cluster.

$$P_{ig}(\mathbf{\Theta}') = \frac{\tau_g p(\mathbf{x}_i|c_g, \mathbf{\Theta}'_g)}{\sum_{j=1}^{G} \tau_j p(\mathbf{x}_i|c_j, \mathbf{\Theta}'_j)} \tag{3.18}$$

In the M-step, the Q function in Equation 3.10 is maximized and this step consists of the update of cluster priors and Poisson parameters:

$$\widehat{\tau}_g = \frac{1}{K} \sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') \tag{3.19}$$

$$\widehat{\theta}_{gm} = \frac{\sum_{i=1}^{K} (P_{ig}(\mathbf{\Theta}') x_{im})}{\sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}')} \tag{3.20}$$

At the end of the EM algorithm each cluster has its own set of parameters such that:

$$pc_g = \{\tau_g, (\theta_{g1}, ..., \theta_{gn})\}$$

The details of EM algorithm for ML estimate problem is given in Appendix B.

MAP Estimate of Model Parameters One difficulty associated with using the maximum likelihood approach relates to zero probabilities. For example, if there is no request for a page p_i in the data set, then our estimate of the Poisson parameter for that page will be zero. That is, according to our model, the probability of requesting page p_i is zero. To address this difficulty, we can assign prior probabilities to $\mathbf{\Theta}$ and use the maximum of the posterior distribution over $\mathbf{\Theta}$ as our estimate for the parameters. Thus, the MAP parameters that correspond to the maximum of posterior distribution of $\mathbf{\Theta}$ can be found by maximizing the posterior probability of $\mathbf{\Theta}$ given the data:

$$\mathbf{\Theta}^{MAP} = argmax_{\mathbf{\Theta}} = \{p(D|\mathbf{\Theta}_1, ..., \mathbf{\Theta}_G, \tau_1, ..., \tau_G)p(\mathbf{\Theta})\} \tag{3.21}$$

where the second identity follows by Bayes' rule and is the prior distribution of the model parameters.

To perform MAP estimate of parameters we first need to choose a functional form for the prior $p(\mathbf{\Theta})$. The parameter set $\mathbf{\Theta}$ consists of a set of Poisson parameters and the class weights. An often used prior distribution for Poisson distribution is Gamma distribution with two parameters of α and β [61]. The distribution of selecting a class can be regarded as a multinomial distribution. The conjugate prior distribution for the multinomial distribution is Dirichlet distribution with the parameter γ [61]. The details of conjugate prior distributions are given in Appendix A.

The choice of the parameters of the conjugate prior distributions is to be determined by one's prior beliefs based on the knowledge of the problem. In general, however, such prior knowledge is difficult to obtain. In the absence of such knowledge one usually uses a "non-informative" prior, typically a uniform prior. In this work several combinations of the parameters are tested.

The E-step of the MAP parameter estimation consists of an update of the conditional probability of missing class labels given the current parameter set $\mathbf{\Theta}'$ as in Equation 3.17. The Q-function for the log-posterior (MAP) function is defined as:

$$Q(\mathbf{\Theta}, \mathbf{\Theta}') = \sum_{i=1}^{K}\sum_{g=1}^{G} P_{ig}(\mathbf{\Theta}')\left[\ln p(\mathbf{x}_i|c_g, \mathbf{\Theta}_g) + \ln \tau_g\right] + \ln p(\mathbf{\Theta}) \tag{3.22}$$

If we maximize the Q-function with respect to each subset of parameters $\mathbf{\Theta}$ one can show that the following update rules for mixture weights and Poisson parameters can be inferred for the M-step of the EM algorithm:

$$\widehat{\tau}_g = \frac{\sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') + \gamma_g}{\sum_{j=1}^{G}\left[\sum_{i=1}^{K} P_{ij}(\mathbf{\Theta}') + \gamma_j\right]} \tag{3.23}$$

$$\widehat{\theta}_{gm} = \frac{\sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}')x_{im} + \alpha_{gm}}{\sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') + \beta_{gm}} \tag{3.24}$$

where γ_i is the hyperparameter associated with $\tau_i,\ i \in [1, ..., G]$, α_{im} and β_{im} are the hyperparameters associated with $\theta_{im},\ i \in [1, ..., G],\ m \in [1, ..., n]$. The details of these step are given in Appendix B.

The output of EM algorithm with MAP estimates is a set of cluster parameters such that each cluster has its own parameters:

$$pc_g = \{\tau_g, (\theta_{g1}, ..., \theta_{gn})\}$$

Example 3.5. For the data set in Table 3.1 we compute in the E-step the cluster posterior probabilities using Equation 3.17. In the M-step we update the model parameters using Equation 3.19 and Equation 3.20. Thus, the parameters in Table 3.2 and the cluster priors are updated in each M-step. The E and M-steps are applied until the convergence criteria is obtained. The output of this algorithm is the set of cluster parameters. For example, $\{0.3; (0.02, 1.2, ...)\}$ tells us that a cluster has a prior probability of 0.3 and the Poisson parameter of the first page is 0.02, of the second page is 1.2 and so on. In case of using the MAP estimate, there will be a small difference in the parameters according to the hyperparameters of conjugate priors. The clusters in Table 3.3 are obtained by assigning each session to the cluster that has the highest cluster-posterior probability.

Table 3.3: Clusters built by using EM algorithm

Cluster No.	S	$PAGES$	$NORMS$
1	1	$\{p_1,\ p_3,\ p_6,\ p_5,\ p_{10}\}$	{1, 0, 8, 0, 3, 10, 0, 0, 0, 1}
	5	$\{p_1,\ p_{10},\ p_8,\ p_2,\ p_5,\ p_3,\ p_6\}$	{1, 1, 7, 0, 3, 10, 0, 1, 0, 1}
	7	$\{p_1,\ p_5,\ p_6,\ p_3\}$	{1, 0, 8, 0, 4, 10, 0, 0, 0, 0}
2	2	$\{p_4,\ p_9,\ p_6,\ p_{10},\ p_7\}$	{0, 0, 0, 2, 0, 1, 10, 0, 8, 10}
	3	$\{p_7,\ p_6,\ p_5,\ p_4,\ p_{10},\ p_9\}$	{0, 0, 0, 2, 1, 1, 9, 0, 8, 10}
3	4	$\{p_1,\ p_3,\ p_6,\ p_5,\ p_{10},\ p_9,\ p_2\}$	{10, 10, 3, 0, 1, 6, 0, 0, 1, 4}
	6	$\{p_9,\ p_{10},\ p_6,\ p_3,\ p_2,\ p_1\}$	{10, 9, 2, 0, 0, 6, 0, 0, 1, 4}

Cluster Profiles

In order to obtain a set of pages for recommending and rank them in this set *recommendation scores* are calculated for every page in each cluster using the Poisson parameters of that cluster. Thus, each cluster has a set of recommendation scores additional to its parameter set created in the previous subsection. We modify the cluster parameters such that each cluster has a recommendation score set, $RS_g = \{rs_{g1}, ..., rs_{gn}\}$, where $rs_{gi}, i \in [1, ..., n]$ is the recommendation score for page p_i in cluster c_g. The updated cluster parameters are then in the form $pc_g = \{\tau_g; (\theta_{g1}, ..., \theta_{gn}); (rs_{g1}, ..., rs_{gn})\}$: Those are the only parameters that the system needs in order to produce a set of pages for recommendation. We define the number of parameters stored in memory as the *model size*. It is clear that the smaller the model size the faster the online prediction.

We use five different methods for calculating recommendation scores for every page. The recommendation scores are then normalized such that the maximum score has a value of 1. These methods are as follows:

Method 1. For the first method, we only use the Poisson parameters of the active cluster as recommendation scores, namely:

$$rs_{gi} = \theta_{gi} \tag{3.25}$$

For the remaining calculations we assign each session in the training set to a cluster that has the highest posterior probability. Next we count the number of requests for every page in each cluster. We define this number as popularity, (f_{gi}), where $i \in [1, ..., n]$ and $g \in [1, ..., G]$. For example, if R_{p_i} is the total number of page requests for page p_i and R_p is the total request for all pages in a cluster c_g, then the popularity of page p_i in that cluster is $f_{gi} = R_{p_i}/R_p$

Method 2. In the second method we use only the popularity information for recommending pages. The intuition behind this is to recommend pages that are most likely visited in a cluster. The recommendation score for page p_i in active cluster c_g is then:

$$rs_{gi} = f_{gi} \tag{3.26}$$

Method 3. For the third method, we calculate recommendation scores by multiplying the popularity by the Poisson parameter:

$$rs_{gi} = f_{gi} \times \theta_{gi} \tag{3.27}$$

Method 4. We use for the fourth recommendation method the entropy values. According to our clustering criteria, the normalized visiting page times for a given page in a cluster should not vary greatly among sessions. Thus, we take advantage of a technique used in decision theory called the *entropy*. We calculate the entropy for each page using the relative frequency of each of the ten possible values of normalized times. A low entropy value means that the visiting time of that page mostly has one of the normalized values. High entropy value, on the other hand, indicates wide divergence in page visiting times among sessions. Our recommendation score is then:

$$rs_{gi} = f_{gi} \times \frac{1}{(entropy)_{gi}} \times \theta_{gi} \tag{3.28}$$

Method 5. For the last calculation, the log of the popularity is taken in order to decrease the effect of the popularity in recommendation score:

$$rs_{gi} = [\log f_{gi}] \times \frac{1}{(entropy)_{gi}} \times \theta_{gi} \tag{3.29}$$

Recommendation Engine

The real-time component of the model calculates cluster posterior probability $P(c_g|s_i)$ for every cluster $c_g \in C = \{c_1, ..., c_G\}$ where s_i is the portion of a session in test set that is used to find the most similar cluster. The active session is assigned to the cluster that has the highest probability. We define this cluster as the *active cluster*. A *recommendation set*, which is the set of predicted pages by the model, is then produced ranking the recommendation scores of the active cluster in descending order. The recommendation set consists of pages which have a recommendation score greater than a threshold ξ (or top N items with the highest recommendation scores where N

is a fixed number) in the active cluster and that the user has not yet visited. The choice of specific alternative depends on the evaluation metric discussed in the Chapter 4.

3.2 Click-Stream Tree Model

In this section, we present a new model that uses both the sequences of visiting pages and the time spent on that pages. Markov models and their variations, or models based on sequence mining have been found well suited for predicting the next request of a Web user. However, higher order Markov models are extremely complicated due to their large number of states whereas lower order Markov models do not capture the entire behavior of a user in a session. The models that are based on sequential pattern mining only consider the frequent sequences in the data set, making it difficult to predict the next request following a page that is not in the sequential pattern. Furthermore, it is hard to find models for mining two different kinds of information of a user session. We propose a new model that considers both the order information of pages in a session and the time spent on them. We cluster user sessions based on their pairwise similarity and represent the resulting clusters by a *Click-Stream Tree* (CST). The new user session is then assigned to a cluster based on a similarity measure. The CST of that cluster is used to generate the recommendation set. The model we propose has two characteristics: 1^o Preservation of the whole path of a user session; and 2^o Preservation of the time information of the visited pages. A path is a sequence of URL's that a user visits in her single session, ordered by time of access. The model can be used as part of a cache prefetching system as well as a recommendation model. The hit ratio is highly satisfactory in both cases.

3.2.1 Background

Our overall approach can be summarized as follows. Once the log data has been cleaned, the user sessions are clustered based on the pairwise similarities of the user sessions. Then, each cluster is represented by a CST such that each branch of the tree is an unique session of the cluster. When a request is received from an active user, a recommendation set consisting of three different pages that the user has not yet visited is produced using the best matching user session[1]. For the first two requests of an active user session all clusters are explored to find the one that best matches the active

[1]The user session that has the highest similarity to the active user session is defined as the best matching session.

user session. For the remaining requests, the best matching user session is found by exploring the top-N clusters that have the highest N similarity values computed using the first two requests of the active user session. The rest of the recommendations for the same active user session are made by using the top-N clusters.

The novelty of our approach lies in the method by which we compute the similarity of user sessions and how we cluster them. We propose a method for calculating the similarity between all pairs of user sessions considering both the order of pages, the distance between identical pages, and the time spent on these pages. The distance between identical pages is taken into consideration, because the similarity between two user sessions also reflects the distance between identical pages as measured by the number of user requests between these pages with the same order of occurrence in these sessions. Using these pairwise similarity values, a graph is constructed whose vertices are user sessions. An edge connecting two vertices in the graph has a weight equal to the similarity between these two user sessions. Using an efficient graph-based clustering algorithm the user sessions are clustered, and each cluster is then represented by a CST whose nodes are pages of user sessions of that cluster.

One Dimensional Sequence Alignment

In our study, we use sequence alignment techniques to analyze the sequence of user requests that appear in user sessions. For two strings of length m and n, optimal sequence alignment has zero or more gaps inserted into the sequence to maximize the number of positions in the aligned strings that match. For example, consider aligning the sequences "p_1 p_2 p_4 p_5" and "p_4 p_1 p_2 p_5 p_3 p_6". By inserting gaps $(-)$ in the appropriate place, the number of positions where two sequences match can be maximized:

$$\begin{array}{ccccccc} - & p_1 & p_2 & p_4 & p_5 & - & - \\ p_4 & p_1 & p_2 & - & p_5 & p_3 & p_6 \end{array}$$

Here the aligned sequences match in three positions. Algorithms for efficiently solving this type of problem are well known and are based on dynamic programming. In our study we use FastLSA algorithm [62] to find the best alignment for a pair of user sessions.

Graph Based Clustering

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and dissimilar to objects in other clusters. In this study, we focus on pairwise data clustering. Formally, the pairwise data clustering problem is defined as follows: Given $S = (D, N, W, C)$, where D is a set of data, $N \subseteq D \times D$ is a set of data pairs, W is a symmetric matrix of similarity values of each data pair in N, and C is the clustering criterion function, partition the data set D into clusters such that the criterion function in C is optimized.

Given the similarity metric between any pair of data items, the symmetric matrix W forms a weighted adjacency matrix (weight matrix) of an undirected graph. Thus, the pairwise data clustering problem becomes a graph partitioning problem. Given a weighted, undirected graph $G = G(V, E)$ with a set of vertices V that represent the data items in D and an edge set E that represent the data pairs in N whose weights are defined by W, we wish to partition it into several subgraphs such that criteria function C is optimized.

3.2.2 Model Implementation

Following the data cleaning and preprocessing steps, we use a similarity metric in the second step for calculating the similarities between all pairs of sessions. In the third step, the sessions are clustered based on this similarity metric using the graph partitioning algorithm and each cluster is represented by a CST. In order to produce the recommendation set, we first select the cluster and then select the path from the CST of the cluster that best matches the active user session.

Session Similarity Measure

In this section, we propose a session similarity measure. Since user sessions are ordered URL requests, we can refer to them as sequences of Web pages. The problem of finding the optimal sequence alignment is solved using a dynamic programming formulation. We develop an algorithm based on FastLSA [62] for this purpose. Briefly, the algorithm consists of three steps. The first step is initialization as in the FastLSA algorithm (Figure 3.3), where a scoring matrix is created with $K + 1$ columns and

$N + 1$ rows where K and N correspond to the size of the sequences to be aligned. One sequence is placed along the top of the matrix (sequence#1) and the other one along the left-hand-side of the matrix (sequence#2). A gap is added to the end of each sequence which indicates the starting point of calculation of similarity score. Since the most recent visited pages are more important for prediction of the next request than older ones, we start the calculation from the end of sequences. There are three scores in this matrix: $Score_{l,r} = s_m$ which means that the residue at position l of sequence #2 is the same as the residue at position r of sequence #1 (match score); otherwise $Score_{l,r} = s_d$ (mismatch score) or $Score_{l,r} = s_g$ (gap penalty). From this starting point, the last row is filled from right-to-left such that each cell in the last row is the sum of the previous cell and the gap penalty. The last column of the matrix is filled from bottom-to-top in the same manner.

Input : s_g
Output : Score matrix M

```
    $M(N+1, K+1) \leftarrow 0$
2:  for $l = N$ down to $l = 1$ do
        $M(l, K+1) \leftarrow M(l+1, K+1) + s_g$
4:  end for
    for $r = K$ down to $r = 1$ do
6:      $M(N+1, r) \leftarrow M(N+1, r+1) + s_g$
    end for
```

Figure 3.3: Algorithm of the initialization step

The second step of the algorithm is FindScore (Figure 3.4), in which we modify the FastLSA algorithm to calculate the two dimensional similarity of sessions. We implemented the algorithm with an additional module for that step that takes into account the time spent on matching pages. In our implementation the identical matching of Web pages and time spent on those pages is given a score $s_m = 2$ and each mismatch or gap inserted to the sequences is given a penalty score of -1, i.e. $s_d = s_g = -1$. Then the two dimensional matching score $Score_{l,r} = s(p_{il}, p_{jr})$ of the matrix is calculated for a pair of matching pages, $p_{il} = p_{jr} = p_m$ as follows:

$$Score_{l,r} = s(p_{il}, p_{jr}) = s_m \frac{min(norm_{p_{il}}, norm_{p_{jr}})}{max(norm_{pil}, norm_{p_{jr}})} \tag{3.30}$$

where $p_m \in P$ and $norm_{p_{il}}$ is the normalized value of time corresponding to the page p_{il} in session S_i, and $norm_{p_{jl}}$ is the normalized value of time corresponding to the page p_{jr} in session S_j. Thus the two dimensional matching score is the ratio of the normalized time values spent on matching pages multiplied by the identical matching score. In that step the scoring matrix is filled by starting in the lower right hand corner in the matrix and finding the maximal score $M_{i,j}$ for each position in the matrix. Going left corresponds to inserting a gap in sequence #2. Going up inserts a gap in sequence #1. Going diagonally up-left corresponds to matching. For each position, $M_{i,j}$ is defined to be the maximum of the three incoming values:

$$M_{i,j} = max[M_{i+1,j+1} + Score_{i,j}(match/mismatch\ in\ the\ diagonal),$$

$$M_{i,j+1} + s_g(gap\ in\ sequence\ \#2),$$

$$M_{i+1,j} + s_g(gap\ in\ sequence\ \#1)]$$

Input : $S_i,\ S_j,\ s_m,\ s_g$
Output : Score matrix M

```
for $r = K$ down to $r = 1$ do
    for $l = N$ down to $l = 1$ do
        if $p_{il} = p_{jr}$ then
            $Score_{l,r} = s_m \frac{min(norm_{p_{il}}, norm_{p_{jr}})}{max(norm_{pil}, norm_{p_{jr}})}$
        else
            $Score_{l,r} = s_d$
        end if
        $Vertical = M(l+1, r) + s_g$
        $Diagonal = M(l+1, r+1) + Score_{l,r}$
        $Horizontal = M(l, r+1) + s_g$
        $M(l, r) = Max[Vertical, Diagonal, Horizontal]$
    end for
end for
```

Figure 3.4: *FindScore* algorithm

The third step is FindPath which is the same as in the FastLSA algorithm and determines the actual alignment(s) that lead to the maximal score (Figure 3.5). FindPath traverses the matrix beginning from the destination point (upper left corner) to the start point (lower right corner) of the matrix. It takes the current cell and looks at the neighboring cells that could be direct predecessors. This means that it looks at the neighbor to the right (gap in sequence #2), the diagonal neighbor (match/mismatch), and the neighbor below it (gap in sequence #1). The algorithm for FindPath chooses as the next cell in the sequence one of the possible predecessors that leads to the maximal score.

Input : Score Matrix M
Output : Aligned sequences Sequence1 and Sequence2

```
$Sequence1 \leftarrow \{\emptyset\}$
$Sequence2 \leftarrow \{\emptyset\}$
$r \leftarrow 1\,, l \leftarrow 1$
while $r < K + 1$ do
    while $l < N + 1$ do
        $Vertical = M(l+1, r)$
        $Diagonal = M(l+1, r+1)$
        $Horizontal = M(l, r+1)$
        $max = Max\{Vertical, Diagonal, Horizontal\}$
        if $max = Vertical$ then
            $Sequence1 \leftarrow Sequence1 + \{-\}$
            $Sequence2 \leftarrow Sequence2 + \{p_{il}\}$
            $l \leftarrow l + 1$
        else
            if $max = Diagonal$ then
                $Sequence1 \leftarrow Sequence1 + \{p_{jr}\}$
                $Sequence2 \leftarrow Sequence2 + \{p_{il}\}$
                $l \leftarrow l + 1\,, r \leftarrow r + 1$
            else
                if $max = Horizontal$ then
                    $Sequence1 = Sequence1 + \{p_{jr}\}$
                    $Sequence2 = Sequence2 + \{-\}$
                    $r \leftarrow r + 1$
                end if
            end if
        end if
    end while
end while
```

Figure 3.5: *FindPath* algorithm

The similarity between sessions is then calculated such that only the identical matching of sequences has a similarity value of 1. The similarity measure has two components, which we define as *alignment score component* and *local similarity component*. The alignment score component computes how similar the two sessions are in the region of their overlap. If the highest value of the score matrix of two sessions, S_i and S_j, is σ and the number of matching pages is M in the aligned sequence, then the alignment score component is:

$$s_a(S_i, S_j) = \frac{\sigma}{s_m * M} \tag{3.31}$$

The intuition behind this is that score σ is higher if the sessions have more consecutive matching pages. This value is normalized by dividing it by the matching score and the number of matching pages. The local similarity component computes how important the overlap region is. If the length of the aligned sequences is L, the local similarity component is :

$$s_l(S_i, S_j) = \frac{M}{L} \tag{3.32}$$

Then the overall similarity between two sessions is given by

$$sim(S_i, S_j) = s_a(S_i, S_j) * s_l(S_i, S_j) \tag{3.33}$$

Table 3.4: Sample user sessions as running example

S	PAGES	NORMS
S_1	$\{p_1,\ p_5,\ p_7,\ p_3,\ p_4\}$	{1, 0, 1, 2, 1, 0, 2, 0, 0, 0}
S_2	$\{p_5,\ p_3,\ p_7,\ p_9\}$	{0, 0, 1, 0, 1, 0, 2, 0, 2, 0}
S_3	$\{p_2,\ p_8,\ p_4,\ p_3\}$	{0, 1, 2, 2, 0, 0, 0, 1, 0, 0}
S_4	$\{p_5,\ p_3,\ p_6,\ p_8\}$	{0, 0, 1, 0, 1, 1, 0, 2, 0, 0}
S_5	$\{p_2,\ p_8,\ p_6,\ p_5\}$	{0, 1, 0, 0, 1, 2, 0, 2, 0, 0}
S_6	$\{p_5,\ p_3,\ p_7,\ p_9\}$	{0, 0, 1, 0, 1, 0, 2, 0, 2, 0}
S_7	$\{p_2,\ p_8,\ p_6,\ p_4\}$	{0, 1, 0, 2, 0, 1, 0, 2, 0, 0}
S_8	$\{p_1,\ p_7,\ p_6,\ p_5\}$	{1, 0, 0, 0, 2, 2, 1, 0, 0, 0}
S_9	$\{p_2,\ p_8,\ p_6,\ p_5,\ p_3\}$	{0, 1, 1, 0, 1, 2, 0, 2, 0, 0}

Table 3.5: The scoring matrix for two dimensional sequences

	p_2	p_8	p_6	p_5	p_3	-
p_2	**2**	-1	-2	-2	-2	-4
p_8	-1	0	-1	-1	-1	-3
p_4	-3	-2	-1	0	0	-2
p_3	-3	-2	-1	0	1	-1
-	-5	-4	-3	-2	-1	0

Example 3.6. Let us illustrate the calculation of the similarity between two user sessions with an example in Table 3.5. Suppose that the set of pages P consists of 10 pages, $P = \{p_1, ..., p_{10}\}$ and the user sessions are as in Table 3.4. For two user session, S_3 and S_9, the score matrix is given in Table 3.5. Since p_2 is identical in both sessions the matching score of this page is $Score_{1,1} = 2 * (1/1) = 2$. Then, the maximum score of that cell in the matrix is $M_{1,1} = M_{2,2} + Score_{1,1} = 0 + 2 = 2$. However, the time spent on page p_3 is not equal in both sessions. The matching score of that page is $Score_{4,5} = 2 * (1/2) = 1$. Then, the maximum score of that cell in the matrix is $M_{4,5} = M_{5,6} + Score_{4,5} = 0 + 1 = 1$. Since the length of the aligned sequences is 5 and the number of matching pages in that sessions is 3, the overall similarity between these sessions is $sim(1, 2) = (2/(2 * 3)) * (3/5)$.

Pairwise Clustering

After calculating pair-wise similarities of all user sessions, a graph is constructed whose vertices are user sessions. There is an edge between two vertices (S_i, S_j) if the similarity value between S_i and S_j computed as described in the previous subsection is greater than 0 and this edge is weighted by this similarity value. The problem of clustering user sessions is formulated as partitioning the graph G into k disjoint subgraphs $G_m,\ (m \in [1, ..., k])$ by minimizing *MinMaxCut* function [63]. MinMaxCut function combines both the minimization of similarity between each subgraph and the maximization of similarity within each subgraph and is defined as:

$$minimize \sum_{m=1}^{k} \frac{cut(G_m,\ G - G_m)}{\sum_{v_i, vj \in Gm} sim(v_i, v_j)}$$

where $cut(G_m,\ G - G_m)$is the sum of edges connecting the vertices in G_m to the rest of the vertices in graph $G - G_m$ and $sim(v_i, v_j)$ is the similarity value between vertices v_i and v_j calculated using the similarity metric. In this study an efficient and fast graph partitioning algorithm called Cluto is used for graph partitioning [64].

Cluster Representation

The clusters created by the graph partitioning algorithm contain user sessions. Each cluster has the following properties:

1. The order of occurrence of Web pages is similar across the user sessions in the same cluster.
2. The distance between identical Web pages of two user sessions in a cluster is shorter than the distance between identical Web pages of two user sessions in different clusters.
3. The amount of time spent on identical Web pages of two user sessions in a cluster is similar.

Each user session in a cluster is a sequence of Web pages visited by a single user and the normalized time spent on those pages with a unique session number. It is important to represent each user session in a cluster while preserving the properties of a cluster. One way of doing this is to represent each unique user session in a cluster as a branch of a tree, which we define as the click-stream-tree. Thus, we generate a click-stream-tree for each cluster. Each CST has a *root* node, which is labelled as “null”. Each node except the *root* node of the click-stream-tree consists of three fields: *data*, *count* and *next_node*. *Data* field consists of page number and the normalized time information of that page. *Count* field registers the number of sessions represented by the portion of the path arriving to that node. *Next_node* links to the next node in the CST that has the same *data* field or null if there is any node with the same *data* field. Each CST has a *data_table*, which consists of two fields: *data* field and *first_node* that links to the first node in the CST that has the *data* field.

The tree for each cluster is constructed by applying the algorithm given in Figure 3.6. Let each session be in the form of $S_i = (PAGES_i,\ NORMS_i)$, where $PAGES_i = \{p_{i1}, ..., p_{ik}\}$ consists of visited pages in session S_i and $NORMS_i$ is the normalized visiting times of pages in P, as stated before in Chapter 2. $S_i.length$ corresponds to the number of pages in $PAGES_i$. We start the algorithm with an empty tree that contains only the *root* node (line 1 in Figure 3.6). For each session of a cluster (line 2) and for each request in the session (line 6) the algorithm does the following: Since the algorithm starts to insert a session in the tree from the *root* node, first the *root* node is stored as $Current\_Node$ (line 5). For each request of the session the requested page and corresponding normalized time value are merged to create the *data* field of a node (line 7) and labelled as $Active\_Data$. If the $Current\_Node$ has a child with the same *data* field as the $Active\_Data$ (line 8), then the *count* field of the child node

Input : Sessions in a cluster
Output : Click-Stream tree

```
Create a $root$ node of a CST, and label it as “null”
for all user sessions in a cluster do
    $S_i \leftarrow$ next user session in the cluster
    $m \leftarrow 0$
    $Current\_Node \leftarrow root$ node of the CST
    while $m \leq S_i.length$ do
        $Active\_Data \leftarrow \{p_{im}\}\_\{norm_{p_{im}}\}$
        if there is a $Child$ of $Current\_Node$ with the same data field then
            $Child.count++$
            $Current\_Node \leftarrow Child$
        else
            create a child node of the $Current\_Node$
            $Child.data = Active\_Data$
            $Child.count = 1$
            $Current\_Node \leftarrow Child$
        end if
        $m++$
    end while
end for
```

Figure 3.6: *Build Click-Stream Tree* algorithm

of the $Current\_Node$ is incremented by 1 (line 9) and the child node is labelled as $Current\_Node$ (line 10). If the $Current\_Node$ does not have a child with the same *data* field as $Active\_Data$ (line 11) a new node with that *data* field is created as the child node of the $Current\_Node$ (line 12), its *data* field is set to $Active\_Data$ (line 13), its count is set to 1 (line 14) and is labelled as $Current\_Node$ (line 15). Thus, the next request of a session is inserted always in a node that is a child of the node of the previous request. Such a tree has an inherent orientation, since each session is considered from beginning to end. This covers the first property of a cluster. If the distance between identical pages of two user sessions is short, then the number of nodes between these pages in the tree is less compared to the number of nodes of pages that have longer distances. This covers the second property of a cluster. By constructing the CST the page number and corresponding normalized time value are merged together which forms the *data_field* of a node in the tree. This covers the last property of a cluster.

Table 3.6: Sessions of two clusters.

Cl.	S	PAGES	NORMS
1	S_1	$\{p_1,\ p_5,\ p_7,\ p_3,\ p_4\}$	{1, 0, 1, 2, 1, 0, 2, 0, 0, 0}
1	S_2	$\{p_5,\ p_3,\ p_7,\ p_9\}$	{0, 0, 1, 0, 1, 0, 2, 0, 2, 0}
1	S_4	$\{p_5,\ p_3,\ p_6,\ p_8\}$	{0, 0, 1, 0, 1, 1, 0, 2, 0, 0}
1	S_6	$\{p_5,\ p_3,\ p_7,\ p_9\}$	{0, 0, 1, 0, 1, 0, 2, 0, 2, 0}
1	S_8	$\{p_1,\ p_7,\ p_6,\ p_5\}$	{1, 0, 0, 0, 2, 2, 1, 0, 0, 0}
2	S_3	$\{p_2,\ p_8,\ p_4,\ p_3\}$	{0, 1, 2, 2, 0, 0, 0, 1, 0, 0}
2	S_5	$\{p_2,\ p_8,\ p_6,\ p_5\}$	{0, 1, 0, 0, 1, 2, 0, 2, 0, 0}
2	S_7	$\{p_2,\ p_8,\ p_6,\ p_4\}$	{0, 1, 0, 2, 0, 1, 0, 2, 0, 0}
2	S_9	$\{p_2,\ p_8,\ p_6,\ p_5,\ p_3\}$	{0, 1, 1, 0, 1, 2, 0, 2, 0, 0}

The children of each node in the CST is ordered in the count-descending order such that a child node with bigger count is closer to its parent node. The code for constructing the CST is given in Appendix C. The programs are written in Java without any code optimization. The resulting CSTs are then used for recommendation.

Example 3.7. Let us illustrate the construction of the CST with an example. Let the sessions in a data set be clustered into 2 clusters and let each cluster consist of the sessions given in Table 3.6. First the *root* of the tree of the first cluster is created and labelled with null (Figure 3.7). The first page of the session S_1 is inserted to the tree as the child of the *root*, and its *count* field is set to 1. The following pages of this session are inserted as the children of the previous page and their *count* fields are set to one (First For Iteration in Figure 3.7). For session S_2, a new node is created with *data* field 5_1 as the child node of the *root* node since that node does not exist in the tree. The remaining pages are inserted as the child node of the previous page in the session (Second For Iteration). For session S_8, since the *root* node has a child with the *data* field 1_1, the *count* of that node is incremented by one. A new node is created with *data* field 7_1, and linked as the child of 1_1 (Fifth For Iteration). The remaining sessions of that cluster are inserted in the tree in the same manner. The CST of the second cluster is built as the first one. Figure 3.8 shows the constructed CSTs. Circles represent tree nodes. Each node in a tree has a *data* field (shown in Figure 3.8 as PageNumber_NormalizedTime in the first line in each node) and a *count* field (shown in Figure 3.8 as [count] in the second line in each node). For simplicity, PageNumber of the *data* field is represented by only the subscript of pages in $PAGES$ field of a user session. After inserting all the sessions in a cluster, the CST of one cluster with the associated *next_node* and *data_table* is shown in Figure 3.7.

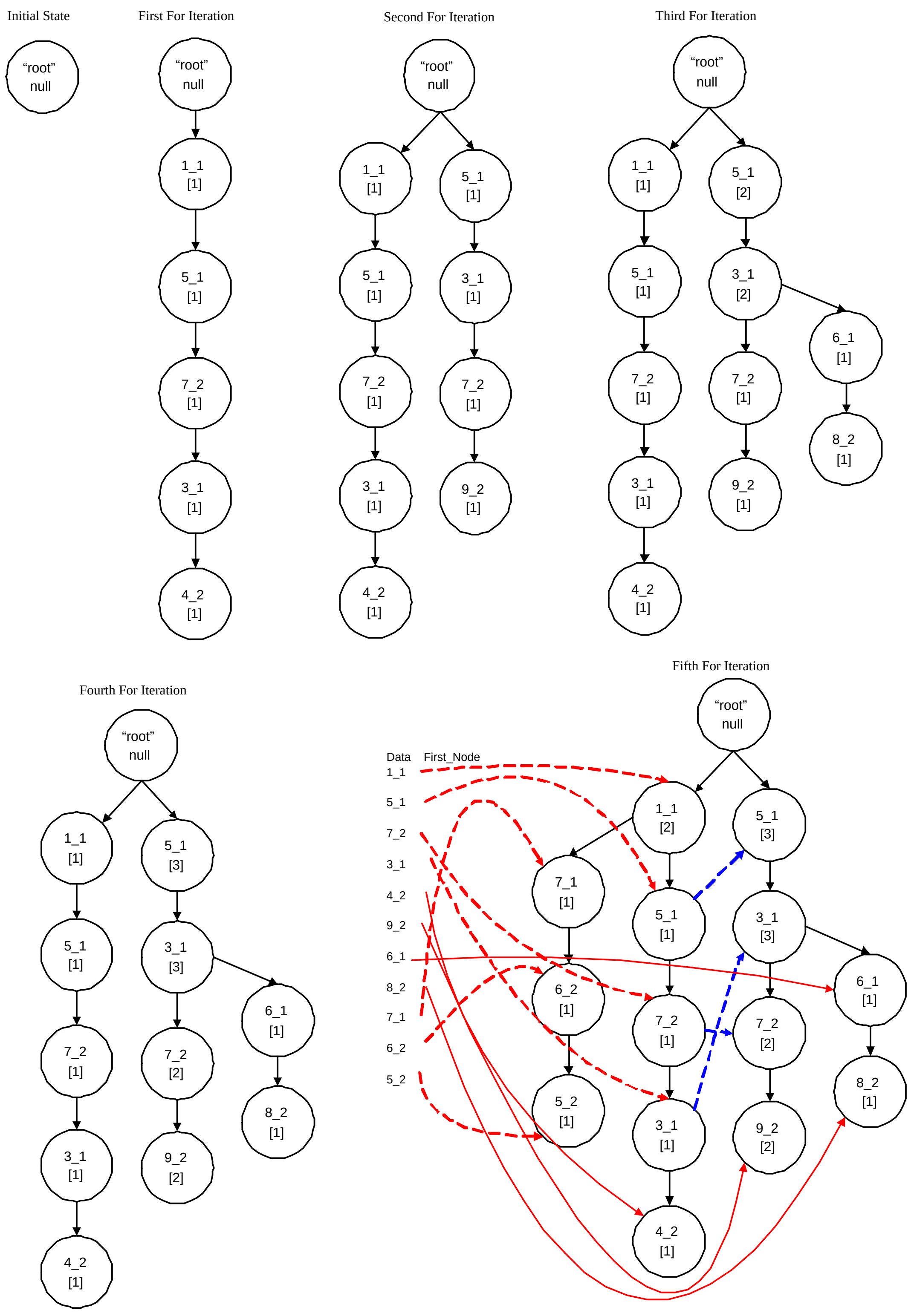


Figure 3.7: Construction of the click-stream Tree of the first cluster

Recommendation Engine

The recommendation engine is the real time component of the model that selects the best path for predicting the next request of the active user session. There is a

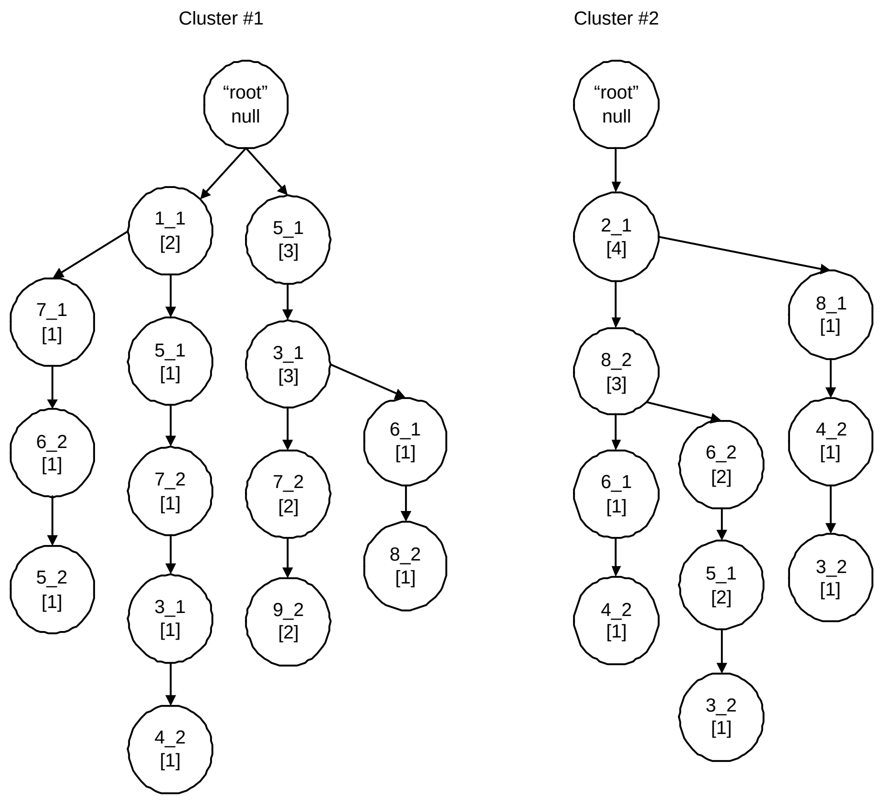


Figure 3.8: Click-stream trees of two clusters

trade-off between the prediction accuracy of the next request and the time spent for recommendation. The speed of the recommendation engine is of great importance in on-line recommendation systems. Thus, we propose the clustering of user sessions in order to reduce the search space and represent each cluster by a CST. Given the time of the last visited page of the active user session, the model recommends three pages. The most recent visited page of the active user session contains the most important information. Before describing the recommendation algorithm, it is useful to first understand a few special properties of the CST. If a *data* field d_i is found in the tree, all the possible paths that contain d_i can be obtained by following d_i's *next_node*, starting from d_i's *first_node* in the data_table of the tree. The CST enables us to insert the entire session of a user without any information loss. We not only store the frequent patterns in the tree but also the whole path that a user follows during his or her session. Besides this, the tree has a compact structure. If a user session with the same $PAGES$ and $NORMS$ fields occurs more than once, only the *count* of its nodes is incremented. Based on the construction of the CST, a user session $(p_{i1}, p_{i2}, ..., p_{ik}), (norm_{p_1}, norm_{p_2}, ..., norm_{p_n})$ occurs in the tree $d_k.count$ times, where d_k is the *data* field formed by merging the page request p_{im} and corresponding normalized time value $norm_{p_{im}}$ of the user session.

Input : Active user session, Click-Stream trees
Output : Best matching session

```
1: $S_a \leftarrow Active\,User\,Session$
2: **if** $S_a.length \leq 2$ **then**
3:    $Clusters = All\,Clusters$
4: **else**
5:    $Clusters = Top - N\ Clusters$
6: **end if**
7: **for** $i = 0$ to $NumberOfClusters$ **do**
8:    $cl = Clusters[i]$
9:    $Sim[cl] = 0$
10:   $m \leftarrow S_a.length$
11:   $d_a \leftarrow \{p_{am}\}\_\{norm_{p_{am}}\}$
12:   $Node \leftarrow data\_table[cl](d_a).first\_node$
13:   $path = null$
14:   **while** $Node \neq null$ **do**
15:      $path = \{path\} + \{Node.data\}$
16:      $Parent\_Node \leftarrow Node.Parent$
17:      **while** $Parent\_Node \neq null$ **do**
18:         $path = \{path\} + \{Parent\_Node.Data\}$
19:         $Parent\_Node \leftarrow Parent\_Node.Parent$
20:      **end while**
21:      $Sim(path) = sim(S_a, path) * Node.count/S[cl]$
22:      **if** $Sim(path) > Sim[cl]$ **then**
23:         $Sim[cl] \leftarrow Sim(path)$
24:         $BestPath[cl] \leftarrow path$
25:      **end if**
26:      $path = null$
27:      $Node \leftarrow Node.next\_node$
28:   **end while**
29: **end for**
30: **if** $S_a.length = 2$ **then**
31:    $Top - N\ Clusters \leftarrow N$ Clusters with highest $Sim[cl]$ values
32: **end if**
```

Figure 3.9: *Find Best Path* algorithm

Figure 3.9 presents the algorithm for finding the path that best matches the active user sessions. For the first two pages of the active user session all clusters are searched to select the best path (line 3 in Figure 3.9). After the second request of the active user top-N clusters that have higher recommendation scores among other clusters are selected (line 30-32) for producing further recommendation sets (line 5). To select the best path we use a backward stepping algorithm. The last visited page and normalized time of that page of the active user session are merged together to build the *data* field (line 11). We find from the *data_table* of the CST of a cluster the *first_node* that has the same *data* field (line 12). We start with that node and go back until the *root* node (or until the active user session has no more pages to compare) to calculate the similarity of that path to the active user session (line 17-20). We calculate the similarity of the optimal alignment. To obtain the recommendation score of a path, the similarity is multiplied by the relative frequency of that path, which we define as the count of the path divided by the total number of paths ($S[cl]$) in the tree (line 21). Starting from the *first_node* of the data field and following the *next_node*, the recommendation score is calculated for the paths that contain the *data* field in the cluster (line 27). The path that has the highest recommendation score is selected as the best path for generating the recommendation set for that cluster (line 22-25). The first three children nodes of the last node of the best path is used for producing the recommendation set. The pages of these child nodes are recommended to the active user.

Example 3.8. Let us illustrate how to find the best path with a simple example for top 1 Cluster (Figure 3.10). Let the active user session be $\langle S_a, [2, 5, 3, 7], [0, 1, 1, 0, 1, 0, 2, 0, 0, 0]\rangle$ and the CSTs be as in Figure 3.8. The first request of the active user is page 2 with a normalized time value 1. The *data* field is formed by merging 2 and 1. Since only cluster 2 has a node with a *data* field 2_1 only the path $Node(2\_1)$ of cluster 2 is examined and found as the best path with a similarity value of 1. $Node(2\_1)$ in cluster 2 has only children with the page number 8 and page 8 is recommended to the active user. The second request of the active user is page 5 with the normalized time value 1. Thus, the active user path becomes $2\_1 \rightarrow 5\_1$. Starting from the *first_node* of $Node(5\_1)$ on the *data_table* of cluster 1, all the paths that contain $Node(5\_1)$ in cluster 1 are examined by following *next_node* of $Node(5\_1)$. Cluster 2 has only one path with a data field 5_1, $Node(2\_1) \rightarrow Node(8\_2) \rightarrow Node(6\_2) \rightarrow Node(5\_1)$, however the similarity value of this path is $1/8$. Thus the path with the highest similarity value in the first cluster, $Node(5\_1)$, is selected as the best path to produce the recommendation set and page 3 is recommended to the active user. The third request of the active user is page 3 with the normalized time value 1. The active user path becomes $2\_1 \rightarrow 5\_1 \rightarrow 3\_1$. After the first two request of the active user, only top 1 cluster is examined for further recommendation. Since the last best path is from cluster 1, the best path is searched only in cluster 1. Starting from the *first_node* of 3_1 on the *data_table* of cluster 1, two paths are examined by following the next_node of $Node(3\_1)$. The path $Node(5\_1) \rightarrow Node(3\_1)$ is found as the best path with a similarity value of $2/5$. Page 7 and page 6 are recommended to the active user.

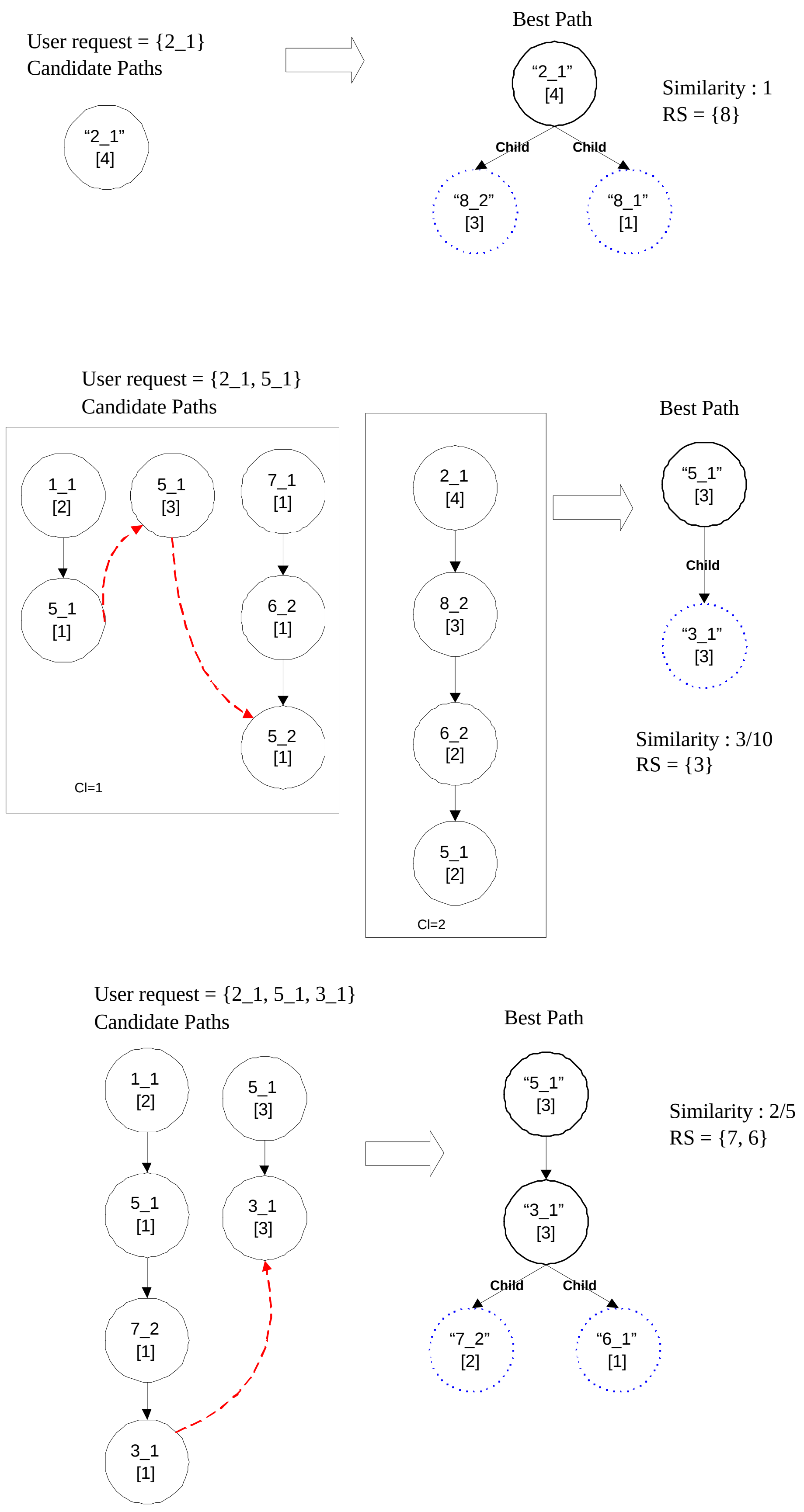


Figure 3.10: A sample recommendation set generation

4. EXPERIMENTAL RESULTS

In order to test the recommendation models discussed in Chapter 3, the raw data collected from server logs are cleaned and converted into user sessions. This chapter discusses the results from several experiments run to test the performance and effectiveness of the models.

4.1 Data Sets

The first data set is from the NASA Kennedy Space Center (NASA) server over the months of July and August 1995 [65]. The second log is from ClarkNet (C.Net) Web server which is a full Internet access provider for the Metro Baltimore-Washington DC area [66]. This server log was collected over the months of August and September, 1995. The last server log is from the Web server at the University of Saskatchewan (UOS) from June to December, 1995 [67].

All the server logs are in Common log format. Since the data sets have different characteristics, the cleaning step results in different numbers of sessions and pages. Even before filtering the data at the last step of the cleaning and preprocessing procedure, 80% of sessions in C.Net Web log have lengths less than three pages and 35% of sessions in UOS Web log have lengths of one page request. Since our evaluation metric is not appropriate for user sessions that have a length less than four pages, we eliminate user sessions that are shorter than this. After cleaning the data sets, the number of sessions are decreased significantly in these logs. Table 4.1 shows the number of remaining URL's and the number of sessions for each data set.

Table 4.1: Characteristics of cleaned log data sets

	NASA	C.Net	UOS
Number of URL's	92	67	171
Number Of Sessions	15369	6846	7452

Approximately 30% of these cleaned sessions are randomly selected as the test set, and the remaining part as the training set. We run the experiments using the same training and test sets for both of the models.

4.2 Evaluation Metrics

The main focus in evaluating the performance of the models is to determine the extent to which the recommended pages match the actual user session. We define the following metrics to evaluate our models:

Hit-Ratio: Given the visiting time of a page in the current session, the model recommends three pages that have the highest recommendation score in the active cluster. A hit is declared if any one of the three recommended pages is the next request of the user. The hit-ratio is the number of hits divided by the total number of recommendations made by the system.

Click-Soon-Ratio: A Click-Soon is declared if any one of the recommended pages is requested by the user during the active user session. The Click-Soon-Ratio is the number of click-soons divided by the total number of recommendations made by the system.

Precision: For each session S_i in the test set we select the first *w* requests in S_i. These *w* requests are used to calculate the active cluster and produce the recommendation set. The recommendation set contains all the pages that have a recommendation score greater than threshold ξ and that are not in the first *w* requests. We denote this set as $PS(w,\xi)$ and the number of pages in this set that match with the remaining part of active session as *m*. Then the precision for a session is defined as:

$$precision(S) = \frac{m}{|PS(w,\xi)|} \tag{4.1}$$

These metrics have been widely used in other studies [68, 69]. Related to our model we use some of these evaluation metrics. For example, precision metric is not appropriate for CSTM. Since the order of user requests are considered in that model, it is more reasonable to use the Hit-Ratio metric.

4.3 Results of the User Interest Model

To evaluate the UIM proposed in Chapter 3, we run several experiments with different sets of initial parameters for EM algorithm and different parameters for prior distributions in case of MAP estimation.

For MAP estimation problem, we conduct the experiments with different parameters ranging from 1 to 5 for Gamma and Dirichlet priors. We use a simple flat prior for Poisson distribution by making all values of α_i and β_i of Gamma distribution equal. For the Dirichlet distribution we set all the values of γ_i to be equal. However, the results of the MAP estimation do not differ more than 0.001% from the results of the ML estimate. This was surprising. Since the data sets are very sparse we expected better results for MAP estimation. Even several combinations of the prior parameters do not lead to a better result. Thus, the remaining experiments are conducted using ML estimation since the model is less complicated during the learning task.

The experiments are performed with different number of clusters. In our experiments, we try different values for the threshold, ξ, of recommendation scores ranging from 0.1 to 0.9. If the threshold is high, then fewer recommendation are produced. If it is small, then irrelevant pages are recommended with a low recommendation score. Our experiments show that setting ξ to 0.5 and *w* to 2 produces few but highly relevant recommendations. We perform the experiments with different numbers of clusters ranging from 4 to 30. These experiments show that normalization of time between 1 and 2 improves the prediction accuracy. For a comparison we only give the results of experiments that are run with the normalization values of 1-2 and 1-10. Table 4.2 and Table 4.3 present the results of the experiments for NASA Kennedy Space Center data set. Table 4.4 and Table 4.5 present the results of C.Net test logs for normalization values 1-2 and 1-10 respectively. Table 4.7 and Table 4.8 present the results of UOS test logs using normalization values 1-2 and 1-10 respectively. As can be seen from the tables, normalization of time between 1-2 increases the prediction accuracy. We define the number of clusters for which we have the highest results as the best number of clusters. The best number of clusters are changing if the normalization time range changes. Table 4.6 summarizes the results of test logs for the best number of clusters where time is normalized between 1-2. Figure 4.1 presents the prediction accuracy of the model for different number of clusters where time is normalized between 1 and 2. As can be seen from the figure, the model is insensitive to the number of clusters in a reasonable range around the best numbers of clusters. The remarkable changes in the number of clusters results in a decrease of the performance of the model. Figure 4.2 presents the prediction accuracy for different normalization values of time.

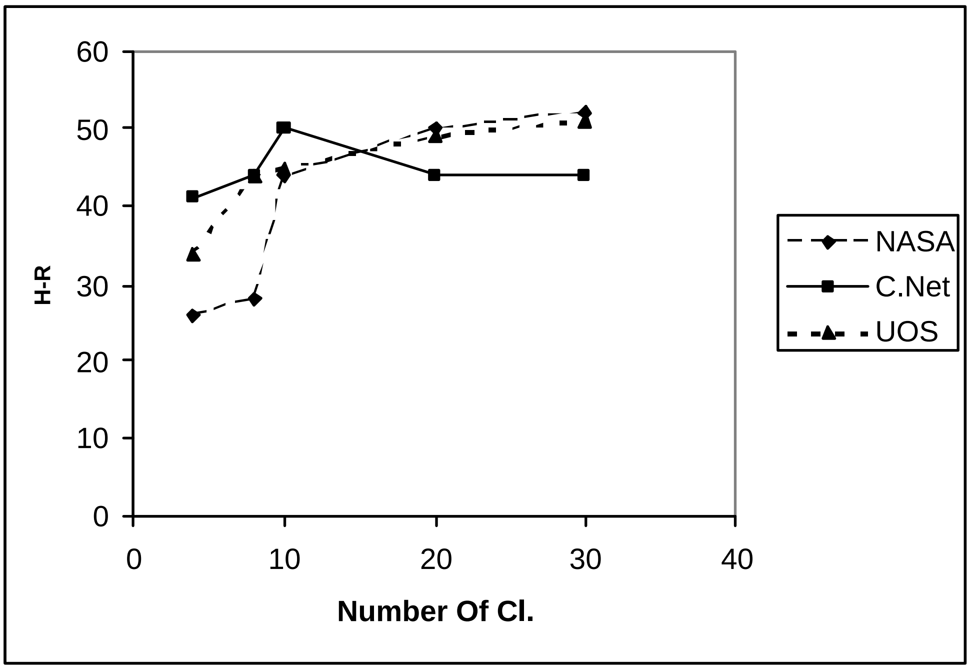


Figure 4.1: Number of clusters-Accuracy(H-R)

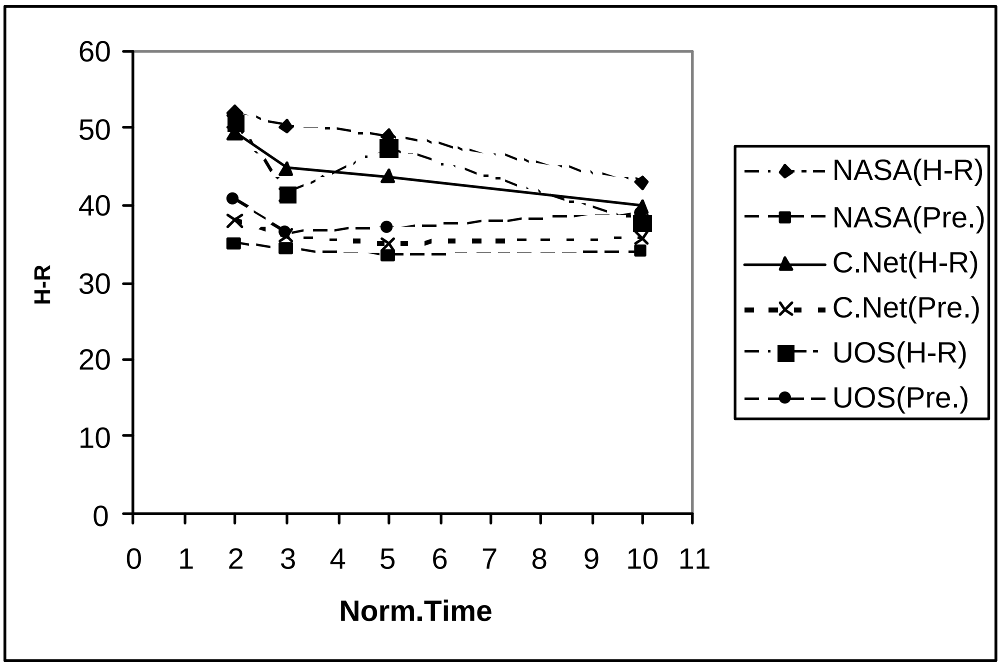


Figure 4.2: Normalization values-Accuracy(H-R)

Table 4.2: Results in % of the NASA data set. Visiting time is normalized between 1 and 2.

No. of Cl.	Method 1			Method 2			Method 3			Method 4			Method 5		
	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R
5	20.96	25.33	51.84	21.2	24.03	49.63	21.2	26	53.2	21.9	25.6	52.4	21	25.3	51.3
10	33.42	43.61	92.66	32.94	43.31	92.85	33.84	43.93	93	32.42	42.87	92.61	30.64	37.46	85.27
15	34.26	45.32	95.8	33.19	45.12	96.14	34.57	45.94	95.99	33.15	43.52	94.28	31.17	39.87	93.14
20	35.1	49.05	97.4	35.53	49.54	98.93	35.37	48.14	98.37	34.14	48.82	97.74	33.20	47.55	96
25	34.15	47.11	91	35.78	46.82	90.18	35.24	47.31	91.34	34.03	46.14	90.48	32.39	43	86.20
30	34.41	51.51	96.4	34.7	51.3	97.03	35	52	96.34	33.76	51.11	96.43	33.76	47.45	92.3

Table 4.3: Results in % of the NASA data set. Visiting time is normalized between 1 and 10.

No. of Cl.	Method 1			Method 2			Method 3			Method 4			Method 5		
	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R
5	24.15	31.33	65.24	24.32	32.24	71.32	27.65	33.55	71.14	28.35	35.06	75.80	24.92	35.29	71.65
10	26.67	36.92	71.12	26.95	41.18	80.26	29.50	38.07	77.16	32.26	40.92	81.09	27.26	39.88	78.77
15	27.06	38.97	75.18	27.41	42.76	80.66	31.54	40.01	77.95	33.15	41.52	81.53	28.17	41.56	79.11
20	29.23	40.76	76.1	28.13	43.05	83.65	32.26	41.42	78.22	34.20	42.89	81.26	30.54	42.80	81.16
25	29.13	41.63	79.21	28.98	42.92	83.14	33	43.10	83.27	34.05	43.12	83.27	30.11	43.18	83.14
30	24.9	33.52	68.32	24.64	35.73	72.16	28.25	34.2	74.11	28.66	36.05	76.43	25.14	35.57	72.14

Table 4.4: Results in % of the ClarkNet data set. Visiting time is normalized between 1 and 2.

No. of Cl.	Method 1			Method 2			Method 3			Method 4			Method 5		
	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R
4	36.29	40.72	99.17	35.83	41.39	99	36.04	40.20	99.54	33.32	39.59	92.17	32.93	39.37	91.19
6	35.92	42.89	99	35.94	42.17	98.87	36.15	41.16	99.54	33.30	49.16	91.19	33.08	39.37	91.16
8	35.88	43.32	99	36.16	43.73	98.54	36.17	42.75	99.55	33.09	40.22	89.54	33.09	39.93	87.37
10	37.9	48.7	94.6	37.6	49.62	95.81	38.18	49.21	95.26	35.4	48.21	94.14	32.88	46.58	51.94
20	37.47	44.32	91.54	37.9	44.08	91.31	37.2	44.15	91.54	35.77	43.60	89.40	32.31	42.18	86.38
30	35.99	43.97	91.44	35.83	43.19	88.94	35.6	44.16	91.01	34.21	43.07	86.79	31.97	41.13	83.63

Table 4.5: Results in % of the ClarkNet data set. Visiting time is normalized between 1 and 10.

No. of Cl.	Method 1			Method 2			Method 3			Method 4			Method 5		
	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R
4	34	35	65.41	30.18	40.13	73.16	36.18	38.14	75.95	36.21	38.64	77.16	33.72	38.05	71.17
6	33.17	32	65.92	29.78	35.88	70.21	35.77	35.07	73.36	34	36	75.21	32	36	72.45
8	32.17	36.47	70.14	29.80	41.78	73.79	33.13	40.80	76.04	33.48	41	77.29	32.56	39	73.43
10	32.01	35.18	70.72	28.40	39.13	73.76	33	35.78	73.74.16	33.32	37	75.43	32.43	35.26	72.12
20	31.47	34.18	71.18	29.41	36.30	72.27	32.29	34.76	71.85	32.33	35.88	72.78	31.79	34.53	70.57
30	31.20	36.92	72.14	30.97	39.79	74.88	32.39	38.01	74.48	32.64	37.87	74.87	31.78	36.91	73.11

Table 4.6: Results (in %) of the model. Visiting time is normalized between 1 and 2.

Data Set	No.Of	Method 1		Method 2		Method 3		Method 4		Method 5	
	Clusters	H-R	Pre.	H-R	Pre.	H-R	Pre.	H-R	Pre.	H-R	Pre.
NASA	30	51.5	34.4	51.3	34.7	52	35	51.1	33.8	47.5	33.8
C.Net	10	48.7	37.9	49.2	37.6	49.6	38.2	48.2	35.4	46.6	32.9
UOS	30	50.8	40.6	50.6	40.7	50.8	40.7	50.5	39.3	50.1	38.7

As can be seen in Table 4.2 and Table 4.3, the accuracy of predictions greatly increases in NASA data set, when we use Hit-Ratio metric. The cause for that may be that we apply further cleaning methods to this data set. Some of the page views in the log data are still available in the NASA Web site. On the other hand, these cannot be applied to others since C.Net Web server does not exist anymore and the URL requests in the UOS log data are not up-to-date. The methods applied during the cleaning tasks produce significant improvement in the prediction accuracy for NASA data set. However, in most cases precision metric performs very poorly compared to the Hit-Ratio metric. This can be due to the fact that for precision metric the active user session is assigned to one of the clusters after the first two requests. The cluster assignment is not calculated for further recommendations to the same user whereas for hit-ratio metric the cluster assignment is calculated after each request of the active user. The recommendation set is created based on this new cluster assignment.

As mentioned in the Chapter 3, we use 5 different methods for calculating recommendation scores. The application of methods that calculate the recommendation scores using popularity term results in marked improvement of the prediction accuracy. This is not surprising, because the popularity represents the common interest among user sessions in each cluster. The results show that using entropy during calculation of recommendation score does not improve the accuracy. This is not surprising for the experiments where page time is normalized in a narrow range. However, even for a wide change in normalized time the entropy does not improve the prediction accuracy as much as we expected. Initially, we assumed this may be due to the fact that EM algorithm learns the best model in terms of model parameters. However, even the results of experiments with different number of clusters reflect the same characteristic. Further examination of the cluster profiles indicate that the popularity of some pages in most of the clusters are zero due to the sparse and scattered nature of the data. Thus, we can not observe the effect of the entropy, since

Table 4.7: Results in % of the University of Saskatchewan data set. Visiting time is normalized between 1 and 2.

No. of Cl.	Method 1			Method 2			Method 3			Method 4			Method 5		
	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R
4	31.96	34.81	81.68	31.98	33.57	81.86	32.05	33.76	82.51	31	33.80	80.34	30.64	33.68	78.72
6	33.12	37.41	86.10	33.14	37.12	86.10	34.17	39	87.11	33.87	38.91	85.74	31.62	38.87	83.79
8	37.33	41.95	99	36.88	44.04	99	36.79	43.58	99	35.28	43.16	96.8	33.93	41.5	93.12
10	37.05	43.01	98.10	36.69	44.98	98.10	36.70	45.32	98.11	35.39	44.32	98	34.08	42.81	97.37
20	38.90	46.10	96.40	36.80	46.48	96.70	37.60	49.06	97.70	37	47.90	96.50	36.90	46	95.80
30	40.64	50.77	98	40.43	50.58	98	40.68	50.82	98.10	39.28	50.5	98	38.74	50.09	98

Table 4.8: Results in % of the University of Saskatchewan data set. Visiting time is normalized between 1 and 10.

No. of Cl.	Method 1			Method 2			Method 3			Method 4			Method 5		
	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R
4	31.12	32.13	72.73	30.13	33.98	71.93	33.86	33.87	72.42	33.93	33.94	74.76	31.35	34.37	70.68
6	32.10	33.75	72.84	32.10	34.21	72.58	35.17	35.51	72.92	36.71	36.81	76.52	32.10	31.16	69.12
8	33.90	35	74.61	33.4	35	73.09	38.47	38	75.10	38.47	39.10	77.51	33.38	35.53	70.90
10	33.61	32	72.49	31.61	39	75.54	36.73	34	72.56	36.22	37.5	75.32	32.62	34.57	69
20	32.80	36	75.69	31.55	41.26	76.12	36.71	39.34	75.49	36.15	40.42	77.10	32.86	37.09	70.45
30	30.37	32.91	72.16	32.58	38.19	74.65	32.92	35.87	72.56	32.53	37.68	74.42	31	35.21	69.13

we multiply the inverse of the entropy with popularity for calculating the recommendationscores. All of our experiments show that in general we can use the third method for calculating recommendation scores, discarding the metric we use for evaluation.

For evaluating the Poisson model in terms of modelling user sessions we use two other distributions: multinomial distribution and binomial distribution. Since multinomial distribution performs very poorly, we only regard binomial distribution. The normalized visiting page time is not considered in that case. The user sessions are modelled only using binary weights for pages, representing existence or non-existence of a page in a user session. The parameters of the binomial distribution are learned using the EM algorithm. The results of these experiments are given in Table 4.9. Method 4 and Method 5 are not used for the evaluation of binomial model. For calculating the recommendation score in Method 4 and Method 5 we need the entropy. Since the weights of the pages are binary the entropy can not be calculated. For this reason, the binomial model is evaluated only using the first 3 methods. As can be seen from the results, Poisson model performs better than the Binomial model, especially Hit-Ratio and Click-Soon metrics outperform by using the Poisson model. The Poisson model performs 38% better than the binomial model for Hit-Ratio metric. These results prove that Poisson model can be used for modelling user sessions and using visiting time improves the prediction accuracy. Besides these experiment and the examination of the histograms as mentioned in the Chapter 3, we examine the occurrence of each of the possible normalized visiting page times. According to our clustering criteria, the normalized visiting page time should not vary among clusters. After clustering user sessions, the histograms are plotted in order to observe the structure of the cluster. These histograms prove that the visiting page times do not vary among user sessions in the same cluster. This may be another evidence that the Poisson distribution could be used for modelling user sessions. However, the model has the flexibility to represent user interest with a mixture of another distribution, e.g., binomial distribution, if one wishes to ignore the visiting time in determining the navigational pattern. This could be considered especially on e-commerce Web sites where the visiting page time is not an indicator of user interest. For this sites, the user interest can be modelled in terms of items that a user buys during her single visit.

We provide some intuitive arguments regarding the advantage of our model in terms of speed and memory usage. The online prediction time correlates strongly with the

Table 4.9: Results (in %) of the model using Binomial distribution.

Data Set	No.Of	Method 1			Method 2			Method 3		
	Clusters	Pre.	H-R	CS-R	Pre.	H-R	CS-R	Pre.	H-R	CS-R
NASA	23	33.33	36.97	49	33.60	36.78	48.77	35.92	36.89	48.84
C.Net	10	34.78	34.17	64.88	34.04	34.19	64.88	38.72	34.16	64.87
UOS	30	34.78	41.22	51.62	34.88	41.22	51.56	38.65	41.27	51.59

model size. The smaller the model size, the faster is the online recommendation. Since we only store the cluster parameters for the prediction of the next page request, our model size is very small. The model size only increases with the number of clusters or the number of pages in the Web site when the Web site has a complex structure. However, it is clear that in that case the application of methods such as sequential pattern mining, association rules or Markov models generate more complex models due to the increasing size of rules or states. Thus, all of these models require some pruning steps in order that they be effective. However, our model provides a high prediction accuracy with a simple model structure.

Table 4.10: Comparison of recommendation models.

Data Set	Poisson Model	Model 1	Model 2	Model 3
NASA	52	4	47.84	52.6
C.Net	49.6	15	49.3	50.08
UOS	50.8	5	44.59	52

For evaluating the effect of the Poisson model, we repeated the experiments with the same training and test sets using three other recommendation methods [11, 70, 71].

The recommendation model proposed in [70] (Model 1 in Table 4.19) is comparable to our model in terms of speed and memory usage. Since the hit-ratio metric has not performed well for the model in [70], we use the precision metric for evaluation. We obtain the best result with 23 clusters for NASA data set and 4 clusters for C.Net data set and 8 clusters for the UOS data set. The C.Net data set has a precision of 15%, whereas the NASA data set has 4% and the UOS has 5%.

Since the model in [71] is based on association rule discovery, it has obviously a greater model size than our model. We select this model in order to compare our results to the results of a model that uses a different approach. For the method in [71] (Model 2 in Table 4.19) we use a sliding window with a window size of 2. The sliding window is the last portion of the active user session to produce the recommendation set. Thus, the model is able to produce the recommendation set only after the first two pages of the active user session. We set the support for association rule generation to a low value (such as 1 %) discarding of increasing the model size in order to have a good prediction accuracy. The hit ratio for the NASA, C.Net and UOS data sets are 47.8%, 49.3%, 44.50% respectively.

For the last set of experiments we use first order Markov models (Model 3 in Table 4.19). The parameters of the Markov model are learned using the EM algorithm. The best results for NASA data set is with 23 clusters, and the hit-ratio and click-soon-ratio are 52.6%. The C.Net data set has, for 4 clusters, a hit-ratio of 50.08%. Finally, the UOS data set has a hit-ratio of 52%. Only the last model has a better prediction accuracy compared with our model. However, the decrease of the prediction accuracy of our models is in an acceptable range considering the model sizes. It is clear that the last model has a greater model size which may lead to a slower online prediction.

These results prove that modelling the user transaction with a mixture of Poisson distributions produces satisfactory prediction rates with an acceptable computational complexity in real-time and memory usage when page time is normalized between 1 and 2.

4.4 Results of the Click-Stream Tree

For each data set we conduct the experiment with a single click-steam tree, without the use of any clusters of user sessions, to compare the performance of the similarity metric and the clustering method. The results obtained by using a single tree (see Table 4.11) gives us the upper bound of the prediction accuracy. In that case we do not have any side effects of the clustering algorithm or the assumptions we made for assigning the active user session to a cluster since the entire tree is searched (with significant run-time overhead, of course).

We repeat the experiments with different number of clusters ranging from 5 to 30 and N ranging from 1 to 3 after the first 2 requests of the user. The experiments are performed for C.Net and UOS data sets with different number of clusters from the experiments of NASA data set. This is done to account for the lower number of sessions and number of pages in the C.Net and UOS data sets. There is a trade-off

Table 4.11: Results in % of the recommendation algorithm with one tree. (NT = Normalized Time, UT = Unity Time)

Data Set	NT		UT		Time(ms.)
	H-R	CS-R	H-R	CS-R	
NASA	61.61	99.63	59.9	95	3.5
C.Net	55.76	100	51.29	92.25	2
UOS	49.49	83.05	53.16	90	1.3

between the prediction accuracy and the time spent for recommendation. When we determine the top-N clusters after the first request of the user, the recommendation is faster, but the accuracy is about 4% lower. Thus, we determine top-N clusters after the first 2 requests, since the time spent for recommendation and the decrease of accuracy are in an acceptable range. In order to study the results we repeat the same experiments without considering the normalized time (Unity Time). If we do not use the time information, the data field of each node in the CST consists of only the page number. The same experiments are then performed by normalizing time between 1 and 3, 1 and 5, and 1 and 10. In the case when the time is normalized between 1 and 3, and the number of clusters is 5, the method with time information performs better for the NASA data set. But in other cases the method without time information outperforms. Since our method for tree construction merges the page number and time information for creating the data field of nodes, the number of data items corresponding to the same page increases if the values of normalized time changes in a wide range. For example for the NASA data set, if the time is normalized between 1 and 10, the number of data items becomes 920, since the number of pages is 92. Thus, for a page we have 10 different data items. Since we only recommend pages to the user, having different data items corresponding to one page makes the recommendation inefficient. For each experiment we register the average time spent to produce the recommendation set. Since the other results are worse, we just present the results of the experiments in which the normalized time has a value between 1 and 2. All Experiments are performed on a Pentium II, 333 MHz PC with a 512 MB main memory running on Microsoft Windows 2000. The programs are coded in Java.

Table 4.12 shows the results of the NASA data set where the visiting time of pages are normalized between 1 and 2. The results in Table 4.13 are obtained without taking time into consideration. Table 4.14 and Table 4.15 show the results of C.Net data set with normalized time between 1 and 2, and without time information, respectively. The

Table 4.12: Results in % of the NASA data set. Visiting time is normalized between 1 and 2.

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
5	57.41	96.47	59.22	98.80	59.79	99.65
10	54.68	91.10	56.15	93.15	57.18	94.53
15	52.61	88.15	54.65	91.15	55.95	92.43
20	50.79	84.45	52.47	86.37	53.51	87.85
25	49.59	81.47	52.17	85.28	53.11	86.50
30	48.75	80.06	51.29	84.92	52.15	85.77

Table 4.13: Results in % of the NASA data set. Time information is ignored.

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
5	56.19	91.17	57.23	92.82	57.95	93.89
10	53.92	88.31	55.07	89.9	56.01	91.27
15	52.3	86.36	53.77	88.21	54.68	89.36
20	48.96	80.69	50.58	83.01	52.10	84.20
25	48.67	80.15	50.02	82.45	50.42	82.98
30	48.33	79.50	49.37	81.17	50.58	82.74

same experiments have been conducted for UOS data set and the results are presented in Table 4.16 and Table 4.17. Figures 4.3, 4.4 and 4.5 present the average time spent to produce one recommendation set using normalized time between 1 and 2 for the three data sets. As can be seen from the figures, using clustering approach reduces the time for producing the recommendation set whereas the prediction accuracy decreases but is still acceptable. As shown in the tables, the method that incorporates time information performs mostly better. These experiments show that normalizing time between 1 and 2 improves the prediction accuracy. However, only with the data set of UOS, the results are worse with normalized time values. As mentioned in Chapter 2, this data set is very sparse such that even the home page is requested only in about 10% of the user sessions. Besides this, as mentioned in the Section 4.1, almost $1/3$ of the user sessions have a length of one page. These characteristics are very unusual for a data set and may arise from the caching effects. However, these characteristics may result in a wrong calculation of visiting page times, which lead to inefficient predictions in case of using time information. In the case of large number of clusters the C.Net data set has a lower prediction accuracy with time information. This is likely due to the fact that C.Net data set is as sparse as the UOS data set and not cleaned to the same extent as the NASA data set. The unique page views are obtained by listing the unique URL's, whereas the pages of the Web site of NASA Kennedy Space Center are retrieved using a *Web crawler* implemented for this work. For the C.Net data set we can not determine different URL's that correspond to the same page in the Web site which causes a lower prediction accuracy for bigger cluster numbers.

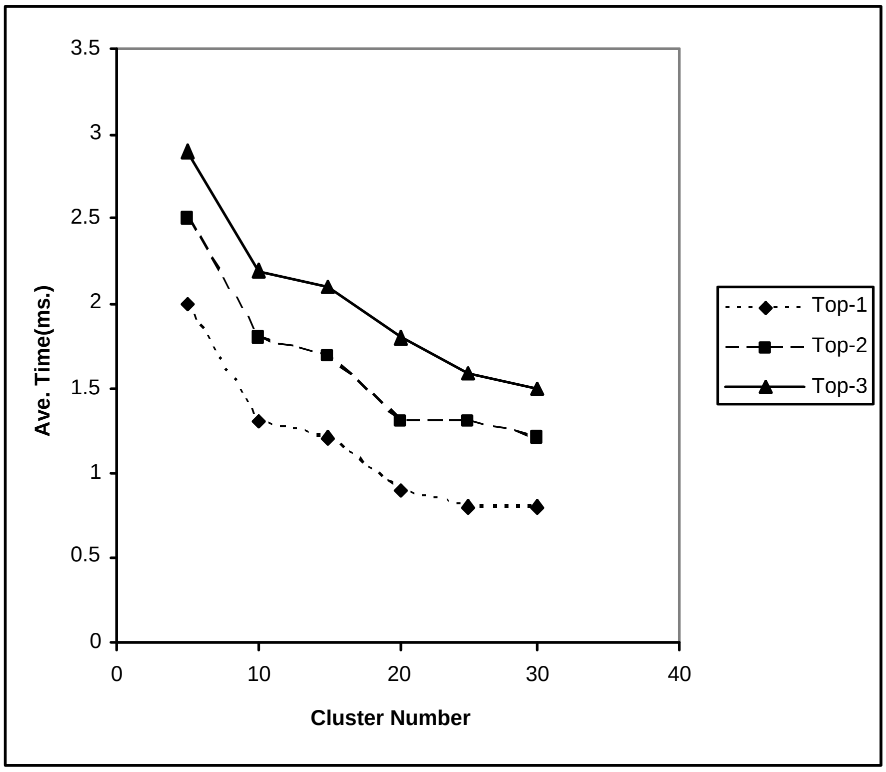


Figure 4.3: Average time in ms. spent to produce one recommendation set for the NASA data set

Table 4.14: Results in % of the ClarkNet data set. Visiting time is normalized between 1 and 2.

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
4	53	100	53.84	100	54.07	100
6	50.53	97.64	50.81	97.86	51.40	98.72
8	49.65	97	50.01	97.20	50.95	97.86
10	48.22	94.07	48.65	94.86	49.01	94.82
20	39.9	76.9	41.51	78.85	42.13	79.50
30	35.65	68.34	37.74	71.75	39.19	73.57

We run the experiments using another similarity metric. The similarity between two user sessions is calculated only using the length of common subsequence and the

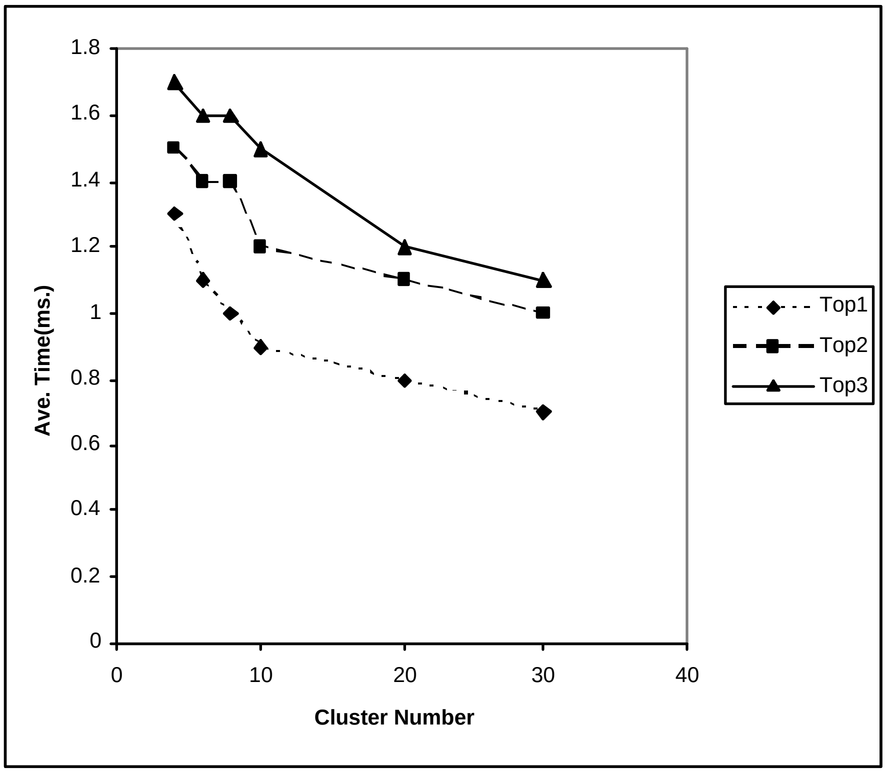


Figure 4.4: Average time in ms. spent to produce one recommendation set for the ClarkNet data set

Table 4.15: Results in % of the ClarkNet data set. Time information is ignored.

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
4	49.94	91.38	51	92.95	51.09	92.96
6	48.69	90.13	49.86	92.19	50.4	92.53
8	48.42	90.05	49.63	92.17	50.22	92.79
10	47.18	88.16	48.64	90.78	48.97	91.32
20	40.95	79.22	43.25	81.80	44.45	83.58
30	37.69	73.29	38.48	74.76	41.23	77.12

length of the user sessions without taking the distance between them into account. For example, let two user sessions be as follows:

$$S_1 = P_1P_3P_2P_4$$

$$S_2 = P_2P_4P_5P_6$$

Table 4.16: Results in % of the University of Saskatchewan data set. Visiting time is normalized between 1 and 2.

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
4	48.07	81.12	48.83	82.17	50.91	82.33
6	47.50	79.45	48.65	80.87	49.81	81.1
8	43.04	73.08	46.23	77.55	48.97	78.39
10	41.83	67.86	44.31	71.12	47.04	72.55
20	38.93	61.82	41.77	64	43.45.48	65.69
30	35.28	64.37	38.87	57.13	42.43	58.67

Table 4.17: Results in % of the University of Saskatchewan data set. Time information is ignored.

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
4	51.07	89.7	52.16	90	53.27	90
6	49.42	87.61	50.78	89.67	51.89	90
8	49.35	88.79	50.30	90	51.32	90
10	48.25	86.11	49.02	87.61	50.26	89.23
20	41.79	62.92	43.92	65.49	44.93	66.74
30	40.60	60.78	42.16	63.28	43.40	64.65

Since the length of matching subsequence is 2 (P_2P_4) and the length of both sessions is 4, the similarity is:

$$sim(s_1, s_2) = \sqrt{(2/4) * (2/4)}$$

Our similarity metric using time information performs about 8% better than this similarity metric. Even without using time information our metric results in about 4% better then this one. For an example, we only give the best results of this experiment in Table 4.18.

Table 4.18: Results with different similarity metric

Data Set	Number of Clusters	Top-3	
		H-R	CS-R
NASA	5	52.1	86.7
C.Net	4	50	95.14
UOS	4	44.28	80.1

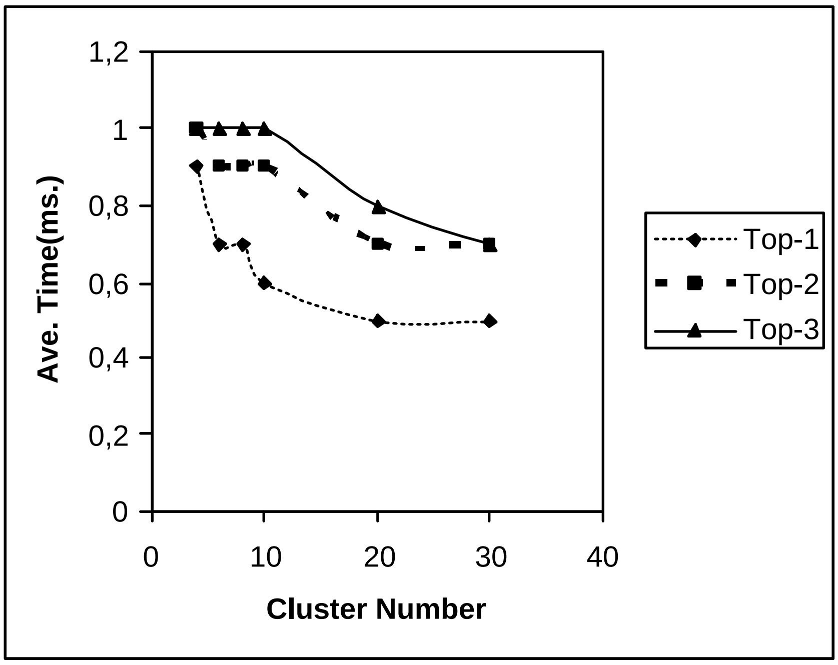


Figure 4.5: Average time in ms. spent to produce one recommendation set for the University of Saskatchewan data set

Table 4.19: Comparison of recommendation models.

Data Set	Metric	Model 1	Model 2	Model 3	C-Str. Tree
NASA	Prec.	4	-	-	48.63
	H-R	-	47.84	-	61.61
		-	-	52.6	59.79
	CS-R	-	84.41	-	99.63
		-	-	86.3	99.65
C.Net	Prec.	15	-	-	49.62
	H-R	-	49.3	-	55.76
		-	-	50.08	54.07
	CS-R	-	92.7	-	100
		-	-	95.12	100
UOS	Prec.	5	-	-	39.9
	H-R	-	44.59	-	49.49
		-	-	52	50.91
	CS-R	-	81.2	-	83.05
		-	-	82.2	82.33

For evaluating the performance of our method, we run the experiments with the same training and test examples using 3 other recommendation methods proposed in [11, 70, 71]. Table 4.19 shows these results in %. Since the experimental conditions for these models are same as in Section 4.3, we only give here the comparison between these models and the CSTM.

Model 1 has a precision for NASA data set of 4%, for C.Net data set of 15% and for UOS data set of 4% respectively.

We set the support for association rule generation to a low value such as 1 %. The hit-ratio and click-soon-ratio for NASA data set is 47.84% and 84.41% respectively. The hit-ratio and click-soon-ratio for C.Net data set is 49.30% and 92.7% respectively. The same model has a hit-ratio of 44.59% and a click-soon ratio of 81.2 for UOS data set. Since this method does not utilize clustering, we can compare these experiments to our experiments in which we use one tree. Clearly our method is superior. Another difference between our model and this one is that our model begins to produce the recommendation set after the first request of the user.

For the last set of experiments, The best results for NASA data set is with 23 clusters, and the hit-ratio and click-soon-ratio are 52.6% and 86.3% respectively. The C.Net data set has for 4 clusters a hit-ratio of 50.08% and click-soon-ratio of 95.12%. Finally, the UOS data set has a hit-ratio of 52% and click-soon-ratio of 82.2%. These results prove that our model performs better than the previous proposed models whether we use one CST or cluster the data set.

Our model has a high click-soon-ratio, in some cases even about 100%. Thus, the model is very useful for a cache prefetching system. Besides this, the clustering approach reduces the search space when working with sites with complex architecture.

5. CONCLUSION

Web usage mining has become an important area of research with the rapid expansion of the World Wide Web. As more data are becoming available, there is much need to study Web user behavior in order to better serve the users and increase business intelligence. In that context, it becomes more important to predict the next action of a Web user. A literature survey reveals that they are many approaches for discovering patterns form Web logs that predict the users' future request based on their current behavior. Despite the fact that visiting time on Web pages is an interesting factor, since it can be employed to measure user's interest in a page, most of previous the studies do not consider the visiting time.

In this work we propose two models to predict a Web user's next request on a Web site based on the time that the user spent on a page and examine the effect of several representations of the visiting page time. The common characteristic of two models is that the discovered patterns do not depend on any personal data about the site users.

For the first model, the mixture of Poisson model is proposed for modelling the behavior of a user in one user session. The main contribution of this model is that user behavior is modelled by the mixture of Poisson distributions without considering the access order of page requests. Experimental evaluation shows that the approach is quite effective in capturing a Web user's access pattern. In order to evaluate the effect of the representation of visiting page time, it is normalized using five different maximum values. These experiments show that normalizing visiting page time in a narrow range improves the prediction accuracy. However, without using the time information, we have a lower prediction accuracy. These results prove that the visiting page time supplies useful information for predicting the user's next requests in case of not considering the order of user accesses. This enables an advantage over previous proposals in terms of speed and memory usage.

However, to improve the prediction accuracy, additional information is needed like the order of user accesses. This leads to a new model that uses two different kinds of information of a user session: Order and visiting page time. We introduce a similarity metric to find pair-wise similarities between user sessions. This similarity metric compares two user sessions by means of visited pages and visiting times. The third significant element of that metric is that it also reflects the distance between matching pages of two user sessions. This second method is aimed at achieving higher prediction accuracy while sacrificing efficiency slightly. To reduce the search space and the time for producing the recommendation set, user sessions are clustered. The main difference between these two models is the time-accuracy tradeoff. Experimental evaluation shows that the model improves the prediction accuracy where visiting time is normalized in a narrow range. Similar to the first model, expanding the upper limits of the normalized time decrease the prediction accuracy.

We are now extending the model in several ways. Since adding the time information as a second dimension to the user sessions improves the prediction accuracy, we are planning to add a third dimension such as the content of Web pages to the user sessions. The similarity between two user sessions can be calculated such that it considers the content of Web pages as well.

Other improvements would be to add a module to the model that updates the model parameters according to the feedback from the users. This feedback will be extracted automatically from the choices of the active user. If the user requests one of the pages from the recommendation set that the model generates it means that the parameters of the model fit to the user's behavior in that session. If the model is unsuccessful in most of the recommendations then the parameters of the model are needed to update.

Ultimately, the field of recommender systems is still young and much work lays ahead. Given the huge amount of information available on the Internet and increasingly important role that the Web plays in today's society, effective Web usage mining will continue to grow as an important tool for analyzing, optimizing, and personalizing Web sites.

REFERENCES

[1] **Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R.**, 1996. Advances in Knowledge Discovery and Data Mining, MIT Press, Cambridge, MA.

[2] **Etzioni, O.**, 1996. The World Wide Web: Quagmire or gold mine,*Communications of the ACM*, **39**, 65-68.

[3] **Borges, J. and Levene, M.**, 1999. Data mining of user navigation patterns, *International WEBKDD Workshop - Web Usage Analysis and User Profiling,* San Diego, CA, USA, August 15, pp. 31-36.

[4] **Madria, S. K., Bhowmick, S. S., Ng, W.K. and Lim E. P.**, 1999. Research issues in Web data mining, *1st International Conference on Data Warehousing and Knowledge Discovery (DaWaK'99)*, Florence, Italy, August 30 - September 1, pp. 303-312.

[5] NETCRAFT, http://www.netcraft.com/.

[6] **Cooley, R., Mobasher, B. and Srivastava, J.**, 1997. Web mining: Information and pattern discovery on the World Wide Web, *9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, Newport Beach, CA, USA, November 03-08, pp. 558-567.

[7] **Kosala, R. and Blockeel, H.**, 2000. Web mining research: A survey, *ACM SIGKDD Explorations*, **2**, 1-15.

[8] **Aggarwal, C. C, Wolf, J. L. and Yu, P. S.**, 1999. Caching on the World Wide Web, *IEEE Transactions on Knowledge and Data Engineering*, **11**, 95-107.

[9] **Shim, J., Scheuermann, P. and Vingralek, R.**, 1999. Proxy cache algorithms: Design, implementation and performance, *IEEE Transactions on Knowledge and Data Engineering*, **11**, 549-562.

[10] **Pitkow, J. and Pirolli, P.**, 1999. Mining longest repeating subsequences to predict World Wide Web surfing, *USENIX Symposium on Internet Technologies and Systems (USITS'99)*, Boulder, CO, USA, October 11-14, pp. 139-150.

[11] **Sarukkai, R. R.**, 2000. Link prediction and path analysis using markov chains, *Computer Networks*, **33**, 377-386.

[12] **Pirolli, P. Pitkow, J. and Rao, R.**, 1996. Silk from a sow's ear: Extracting usable structures from the Web, ACM Conference on Human Factors in Computing Systems (CHI'96), Vancouver, BC, Canada. April 17, pp. 118-125.

[13] **Brin, S. and Pagepp, L.**, 1998. The anatomy of large-scale hypertextual Web search engine, Seventh International World-Wide Web Conference, Brisbane, Australia, April 14-18, pp. 107-117

[14] **Goldberg, D., Nichols, D., Oki, B. and Terry, D.**, 1992. Using collaborative filtering to weave an information tapestry, *Communications of the ACM*, **35**, 61-70.

[15] **Yan, T. W. and Molina, H. G.**, 1999. The SIFT information dissemnination system, *ACM Transactions on Database Systems*, **24**, 529-565.

[16] **Mostafa, J., Mukhopadhyay, S., Lam, W. and Palakal, M.**, 1997. A multilevel approach to intelligent information filtering: Model, system and evaluation, *ACM Transactions on Information Systems*, **15**, 368-399.

[17] **Konstan, J. A., Miller, B. N., Maltz, D. , Herlocker, J. L., Gordon, L. R. and Riedl, J.**, 1997. Grouplens: Applying collaborative filtering to usenet news, *Communications of the ACM*, **40**, 77-87.

[18] **Balabonović, M. and Shoham, Y.**, 1995. Learning information retrieval agents: Experiments with automated Web browsing, *The AAAI Spring Symposium on Information Gathering from Heterogenous, Distributed Resources*, Stanford, CA. USA, pp. 13-18.

[19] **Shahabi, C., Banaei-Kashani, F., Chen, Y.-S. and McLeod, D.**, 2001. Yoda: An accurate and scalable Web-based recommendation system, *6th International Conference on Cooperative Information Systems*, Trento, Italy, September, pp. 318-332,

[20] **Buono, P., Costabile, M. F., Guida, S., Piccinno, A. and Tesoro, G.**, 2001. Integrating user data and collaborative filtering in a Web recommendation system, *8th International Conference on User Modeling*, Sonthofen, Germany, July 13-17, pp. 315-321.

[21] **Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B. Herlocker, J. and Riedl, J.**, 1999. Combining collaborative filtering with personal agents for better recommendations, *The Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, Orlando, FL, USA, July 18-22, pp. 439-446.

[22] **Terveen, L., Hill, W., Amento, B., McDonald, D. and Creter, J.** , 1997. Phoaks: A system for sharing recommendations, *Communications of the ACM*, **40**, 59–62.

[23] **Rucker, J. and Polano, M. J.**, 1997. Siteseer: Personalized navigation for the Web, *Communications of the ACM*, **40**, 73-75.

[24] **Shardanand, U. and Maes, P.**, 1995. Social information filtering: Algorithms for automating word of mouth., *ACM Conference on Human Factors in Computing Systems (CHI'95)*, Denver, CO, USA, May 7-11, pp. 210-217.

[25] **Perkowitz, M. and Etzioni, O.**, 1998. Adaptive Web sites: Automatically synthesizing Web pages, *Fifteenth National Conference on Artificial Intelligence*, Madison, WI, USA, July 1998, pp. 727-733.

[26] **Larsen, J., Hansen, L. K., Szymkowiak, A., Christiansen, T. and Kolenda, T. K.**, 2002. Webmining: Learning from the World Wide Web, *Computational Statistics and Data Analysis*, **38**, 517-532.

[27] **Agrawal, R., Imielinski, T. and Swami, A.**, 1993. Mining association rules between sets of items in large databases, *ACM SIGMOD Conference on Management of Data*, Washington, D.C, USA, May 26-28, pp. 207–216.

[28] **Yang, Q., Zhang, H.H. and Li, I.T.**, 2001. Mining Web logs for prediction models in WWW caching and prefetching, *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, August 26 - 29, pp. 473-478.

[29] **Chen, M.-S., Park, J. S. and Yu, P. S.**, 1998. Efficient data mining for path traversal patterns, *IEEE Transaction on Knowledge and Data Engineering*, **10**, 209-221.

[30] **Yan, T. W., Jacobsen, M., Garcïa-Molina, H. and Dayal, U.**, 1996. From user access patterns to dynamic hypertext linking, *The Fifth World Wide Web Conference (WWW5)*, Paris, France, May 6-10, pp. 1007-1014.

[31] **Shahabi, C., Zarkesh, A., Adibi, J. and Shah, V.**, 1997. Knowledge discovery from users Web-page navigation, *Seventh International Workshop on Research Issues in Data Engineering*, Birmingham, England, April 7-8, pp 20-29.

[32] **Macqueen, J.**, 1967. Some methods for classification and analysis of multivariate observations, *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, **1**, 281-297.

[33] **Fu, Y., Sandhu, K. and Shih, M.**, 1999. A generalization-based approach to clustering of Web usage sessions, *International WEBKDD Workshop - Web Usage Analysis and User Profiling*, San Diego, CA, USA, August 15 pp. 21-38.

[34] **Banerjee, A. and Ghosh, J.**, 2001. Clickstream clustering using weighted longest common subsequences, *Workshop on Web Mining, SIAM Conference on Data Mining*, Chicago, IL, USA, pp. 33–40.

[35] **Cadez, I., Gaffney, S. and Smyth, P.**, 2000. A general probabilistic framework for clustering individuals and objects, *Sixth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 140-149, 2000

[36] **Cadez, I., Heckerman, D., Meek, C., Smyth, P. and White, S.**, 2000. Visualization of navigational patterns on Web site using model based clustering, *Technical Report*, **MSR-TR-0018**, Microsoft Research, Microsoft Corporation USA.

[37] **Anderson, C. R., Domingos, P. and Weld, D. S.**, 2002. Relational markov models and their application to adaptive Web navigation, *Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 23-26, pp. 143-152.

[38] **Zaïane, O. R., Xin, M. and Han, J.** , 1998. Discovering Web access patterns and trends by applying OLAP and data mining technology on Web logs,*Advances in Digital Libraries (ADL298)*, Santa Barbara, CA. USA, April, pp. 19-29.

[39] **Agrawal, R. and Srikant, R.**, 1995. Mining sequential patterns, *International Conference on Data Engineering (ICDE'95)*, Taipei, Taiwan, March, pp.3-14.

[40] **Perkowitz, M. and Etzioni, O.**, 2000. Adaptive Web sites, *Communications of the ACM*, **43**, 152-158.

[41] **Perkowitz, M. and Etzioni, O.**, 1999. Towards adaptive Web sites: conceptual framework and case study, *Computer Networks (Amsterdam, Netherlands: 1999)*, **31**, 1245-1258.

[42] **Spiliopoulou, M. and Faulstich, L. C.**, 1998. , *In International Workshop WebDB'98: World Wide Web and Databases*, Valencia, Spain, March 27-28, pp. 184-203.

[43] **Schechter, S. Krishnan, M. and Smith, M. D.**, 1998. Using path profiles to predict http requests, *Seventh International World Wide Web Conference*, Brisbane, Australia, April, pp. 457-467.

[44] **Agrawal, R. and Srikant, R.**, 1994. Fast algorithms for mining association rules, *20th Conference on Very Large Data Bases (VLDB'94)*, September, pp. 487–499.

[45] **Nanopoulos, A. Katsaros, D. and Manolopoulos, Y.**, 2001. Effective prediction of Web-user accesses: a data mining approach, *International WEBKDD Workshop– Mining Log Data Across All Customer TouchPoints*, San Francisco, CA, USA. August 26.

[46] **Russel, S. and Norwig, P.**, 1995. Artificial Intelligence: A Modern Approach, Prentice Hall, Englewood Cliffs, NJ, USA.

[47] **Joachims, T., Freitag, D. and Mitchell, T.**, 1997. Webwatcher: A tour guide for the world wide Web, *15th International Conference on Artificial Intelligence*, Nagoya, Japan, August 23-29, pp. 770-777.

[48] **Lieberman, H.**, 1995. Letizia: An agent that assists Web browsing, *International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Quebec, Canada, August 20-25, pp. 924-929.

[49] **Etzioni, O. and Weld, D.**, 1994. A softbot-based interface to the internet, *Communications of the ACM*, **37**,72-76.

[50] **Perkowitz, M. and Etzioni, O.**, 1995. Category translation: Learning to understand information on the internet, *International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Quebec, Canada, August 20-25, pp. 930-936.

[51] **Cooley, R., Mobasher, B. and Srivastava, J.**, 1999. Data preparation for mining world wide Web browsing patterns, *Journal of Knowledge and Information Systems*, **1**, 5-32.

[52] **Srivastava, J., Cooley, R., Deshpande, M. and Tan, P. N.**, 2000. Web usage mining: Discovery and application of usage patterns from Web data, *ACM SIGKDD Explorations*, **1**, 12-23.

[53] **Zaïane, O. R.**, 2001. Web usage mining for a better Web-based learning environment, *Conference on Advanced Technology for Education*, Banf, Alberta, Canada, June 27–28, pp. 60-64.

[54] World Wide Web Consortium, http://www.w3.org/.

[55] **Catledge, L. D. and Pitkow, J. E.**, 1995. Characterizing browsing strategies in the World-Wide Web, *Computer Networks and ISDN Systems*, **27**, 1065-1073.

[56] A standard for robot exclusion, http://info.webcrawler.com/mak/projects/robots/norobots.html.

[57] **Han, J., Pei, J. and Yin, Y.**, 2000.Mining frequent patterns without candidate generation, *ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, USA, May 16-18, pp. 1-12.

[58] **Dempster, A. P., Laird, N. M. and Rubin, D. B.**, 1977. Maximum likelihood from incomplete data via the EM algorithm, *Journal of Royal Statistical Society*, **39**, 1-38.

[59] **Hand, D., Mannila, H. and Smyth, P.**, 2001. Principles of Data Mining, The MIT Press, USA.

[60] **Bartoszyński, R. and Niewiadomska-Bugaj. M.**, 1996. Probability and Statistical Inference, John Wiley & Sons, Inc.

[61] **DeGroot, M.H. and Schervish, M.J.**, 2002. Probability and Statistics, Addison-Wesley Publishing.

[62] **Cahrter, K., Schaeffer, J. and Szafron, D.**, 2000. Sequence alignment using FastLSA, *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'2000)*, Monte Carlo Resort, Las Vegas, Nevada, USA, June 26 - 29, pp. 239-245.

[63] **Ding, C., He, X., Zha, H., Gu, M. and Simon, H.**, 2001. Spectral min-max cut for graph partitioning and data clustering, *Technical Report*, **TR-2001-XX**, University of California, Berkeley, CA., USA.

[64] Cluto, http://www-users.cs.umn.edu/ karypis/cluto/index.html.

[65] NASA Kennedy Space Center Log, http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html.

[66] ClarkNet WWW Server Log, http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html.

[67] The University of Saskatchewan Log, http://ita.ee.lbl.gov/html/contrib/Sask-HTTP.html.

[68] **Sarwar, B., Karypis, B., Konstan, J. and Riedl, J.**, 2000. Analysis of recommendation algorithms for e-commerce, *ACM Conference on Electronic Commerce*, Minneapolis, MN, USA, October 17-20, pp. 158-167..

[69] **Cosley, D., Lawrence, S. and Pennock, D. M.**, 2002. REFEREE: An open framework for practical testing of recommender systems using researchindex, *28th International Conference on Very Large Databases*, Hong Kong, August 20-23.

[70] **Mobasher, B., Dai, H., Luo, T. and Nakagawa, M.**, 2000. Discovery of aggregate usage profiles for Web personalization, *International WEBKDD Workshop – Web Mining for E-Commerce: Challenges and Opportunities*, Boston, MA, USA, August 20,

[71] **Mobasher, B., Dai, H., Luo, T. and Nakagawa, M.**, 2001. Effective personalization based on association rule discovery from Web usage data, *Third ACM Workshop on Web Information and Data Management*, Atlanta, USA, November 9, pp. 9-15.

A. Appendix: Prior Distributions

A.1 Prior of Poisson Distribution

When samples are taken from a Poisson distribution, the family of Gamma distribution is a conjugate prior for Poisson distribution. Conjugate prior is a prior that yields a posterior that is identical to the functional form of the prior. Let D be a data set $D = \{X_1, ..., X_n\}$. Suppose $X_1, ..., X_n$ form a random sample from a Poisson distribution for which the value of the mean θ is unknown. Suppose also that the prior distribution of θ is a Gamma distribution with given parameters α and β ($\alpha > 0$ and $\beta > 0$):

$$p(\theta) = \frac{\beta^{\alpha}\theta^{\alpha-1}e^{-\beta\theta}}{\Gamma(\alpha)} \propto \theta^{\alpha-1}e^{-\beta\theta} \tag{A.1}$$

where for positive integers α:

$$\Gamma(\alpha) = (\alpha - 1)!$$

Then the posterior distribution of θ, given that $X_i = x_i$ $(i = 1, ..., n)$ is a Gamma distribution with parameters $\alpha + \sum_{i=1}^{n} x_i$ and $\beta + n$.

Proof: Let $y = \sum_{i=1}^{n} x_i$, then the likelihood function $p(D|\theta)$ satisfies the relation $p(D|\theta) \propto e^{-n\theta}\theta^{y}$.

If X is a random variable from a Poisson distribution, then the probability of observing a particular observation x_i is:

$$p(X = x_i|\theta) = \frac{\theta^{x_i}e^{-\theta}}{x_i!} \tag{A.2}$$

The probability of observing a full data set $D = \{X_1, ..., X_n\}$ is known as the likelihood and is defined as:

$$\begin{aligned} p(D|\theta) &= \prod_{i=1}^{n} p(X = x_i|\theta) \\ &= \prod_{i=1}^{n} \frac{\theta^{x_i} e^{-\theta}}{x_i!} \propto \theta^{\sum_i x_i} e^{-n\theta} \end{aligned} \tag{A.3}$$

The posterior distribution of the parameter θ is:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \propto p(D|\theta)p(\theta) \tag{A.4}$$

Substituting Equation A.1 and A.3 into Equation A.4 we obtain:

$$\begin{aligned} p(\theta|D) &\propto \theta^{\sum_i x_i} e^{-n\theta} \theta^{\alpha-1} e^{-\beta\theta} \\ &\propto \theta^{\alpha+y-1} e^{-\theta(n+\beta)} \end{aligned} \tag{A.5}$$

Thus the posterior distribution of θ is again a Gamma distribution with parameters $\alpha + \sum_{i=1}^{n} x_i$ and $\beta + n$.

A.2 Prior of Multinomial Distribution

Suppose that the random variables $D = \{X_1, ..., X_k\}$ come from a multinomial distribution, then the joint probability function is:

$$p(X_1 = x_1, ..., X_k = x_k) = \frac{N!}{x_1!...x_k!} \tau_1^{x_1} ... \tau_k^{x_k} \propto \prod_{i=1}^{k} \tau_i^{x_i} \tag{A.6}$$

where

$$x_1 + ... + x_k = N$$

$$\boldsymbol{\tau} = \{\tau_1, ..., \tau_k\},\ \tau_1 + ... + \tau_k = 1$$

Then the conjugate prior for $\boldsymbol{\tau}$ is a Dirichlet distribution with parameters $\boldsymbol{\gamma} = \{\gamma_1, ..., \gamma_k\}$ is given by:

$$p(\boldsymbol{\tau}) = \frac{\Gamma\left(\sum_{i=1}^{k} \gamma_i\right)}{\prod_{i=1}^{k} \Gamma(\gamma_i)} \prod_{j=1}^{k} \tau_j^{\gamma_j - 1} \propto \prod_{j=1}^{k} \tau_j^{\gamma_j} \tag{A.7}$$

subject to $\gamma_j > 0$. Then the posterior distribution of $\boldsymbol{\tau}$ is another Dirichlet distribution with parameters $\gamma_1 + x_1, ..., \gamma_k + x_k$.

Proof: Given the Dirichlet prior for $\boldsymbol{\tau}$, suppose we observe data, D, from a multinomial distribution such that there are x_i occurrences of state i for $i = 1, ..., k$. Then the likelihood function satisfies the relation $p(D|\boldsymbol{\tau}) \propto \prod_{i=1}^{k} \tau_i^{\gamma_i + x_i}$.

The probability of observing D is:

$$p(D|\boldsymbol{\tau}) \propto \prod_{i=1}^{k} \tau_i^{x_i} \tag{A.8}$$

The posterior distribution of $\boldsymbol{\tau}$ can be written using Bayes' rule as:

$$p(\boldsymbol{\tau}|D) = \frac{p(D|\boldsymbol{\tau})p(\boldsymbol{\tau})}{p(D)} \propto p(D|\boldsymbol{\tau})p(\boldsymbol{\tau}) \tag{A.9}$$

Substituting Equation A.7 and A.8 into Equation A.9 yields the following result:

$$\begin{aligned} p(\boldsymbol{\tau}|D) &\propto \prod_{i=1}^{k} \tau_i^{x_i} \prod_{j=1}^{k} \tau_j^{\gamma_j} \\ &\propto \prod_{i=1}^{k} \tau_i^{x_i + \gamma_i} \end{aligned} \tag{A.10}$$

which is again a Dirichlet distribution with parameters $\gamma_1 + x_1, ..., \gamma_k + x_k$.

B. Appendix: The EM algorithm for Mixture of Poisson Distributions

B.1 The ML Optimization Framework

To compute the necessary equations used for obtaining ML parameters in the E-step, we should compute the conditional probability of missing class labels given the current parameter set $\mathbf{\Theta}'$. We define this probability as *cluster-posterior* probability, $P_{ig}(\mathbf{\Theta}')$, that the session $\mathbf{x}_i$ arose from the g^{th} cluster. We can write the cluster-posterior probability using Bayes' rule as:

$$\begin{aligned} P_{ig}(\mathbf{\Theta}') &= p(\mathbf{C} = c_g|\mathbf{x}_i) \\ &= \frac{p(\mathbf{C} = c_g)p(\mathbf{x}_i|c_g, \mathbf{\Theta}'_g)}{p(\mathbf{x}_i)} \\ &= \frac{\tau_g p(\mathbf{x}_i|c_g, \mathbf{\Theta}'_g)}{\sum_{j=1}^{G} \tau_j p(\mathbf{x}_i|c_j, \mathbf{\Theta}'_j)} \end{aligned} \tag{B.1}$$

The Q-function can be written as:

$$Q(\mathbf{\Theta}, \mathbf{\Theta}') = \sum_{i=1}^{K} \sum_{g=1}^{G} P_{ig}(\mathbf{\Theta}') \left[\ln p(\mathbf{x}_i|c_g, \mathbf{\Theta}_g) + \ln \tau_g\right] \tag{B.2}$$

In M-step, keeping the cluster-posterior probabilities fixed, we reassign a new set of parameters $\mathbf{\Theta}'(n+1)$ so as to maximize the expected log likelihood of the training data. The Q-function is maximized subject to the constraint that the cluster priors sum to 1. In order to perform constrained maximization, a Lagrange multiplier is used. The estimating equations for cluster priors are as follows:

$$\begin{aligned} \frac{\partial}{\partial \tau_g}\left[Q(\mathbf{\Theta}, \mathbf{\Theta}') - \lambda \sum_{j=1}^{G} \tau_j\right] &= 0 \\ \sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') \left[\frac{1}{\tau_g}\right] - \lambda &= 0 \end{aligned} \tag{B.3}$$

from which it follows:

$$\lambda \tau_g = \sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') \tag{B.4}$$

If we sum Equation (B.4) over g we obtain:

$$\lambda = \sum_{i=1}^{K} \sum_{g=1}^{G} P_{ig}(\mathbf{\Theta}') = K \tag{B.5}$$

Last equation follows from the fact that $\sum_{g=1}^{G} P_{ig}(\mathbf{\Theta}') = 1$. By combining Equation (B.4) and Equation (B.5), we obtain the equation for updating the cluster probabilities:

$$\widehat{\tau}_g = \frac{1}{K} \sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') \tag{B.6}$$

Similarly we can maximize the Q-function with respect to the parameters of Poisson model, $\mathbf{\Theta}_g$, under the independence assumption:

$$\begin{aligned} \frac{\partial}{\partial \theta_{gm}} [Q(\mathbf{\Theta}, \mathbf{\Theta}')] &= 0 \\ \frac{\partial}{\partial \theta_{gm}} \left[\sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}'_g) \left(\ln \prod_{j=1}^{n} \frac{(\theta_{gj})^{x_{ij}}\, e^{-\theta_{gj}}}{x_{ij}!} + \ln \tau_g \right) \right] &= 0 \\ \sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') \left[\frac{x_{im}}{\theta_{gm}} - 1 \right] &= 0 \end{aligned} \tag{B.7}$$

which yields the following update equation for Poisson parameters:

$$\widehat{\theta}_{gm} = \frac{\sum_{i=1}^{K} (P_{ig}(\mathbf{\Theta}') x_{im})}{\sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}')} \tag{B.8}$$

B.2 The MAP Optimization Framework

The E-step of the MAP estimation of parameters is the same as the ML estimation problem where the conditional probability of missing class labels given the current parameter set $\mathbf{\Theta}'$ is computes as in Equation B.1. The Q function of the EM algorithm for the log posterior function is defined as:

$$Q(\mathbf{\Theta}, \mathbf{\Theta}') = \sum_{i=1}^{K} \sum_{g=1}^{G} P_{ig}(\mathbf{\Theta}') \left[\ln p(\mathbf{x}_i | c_g, \mathbf{\Theta}_g) + \ln \tau_g \right] + \ln p(\mathbf{\Theta}) \tag{B.9}$$

where the parameters $\mathbf{\Theta}$ consists of all mixture model parameters:

$$\begin{aligned} \mathbf{\Theta} &= \{\mathbf{\Theta}_1, ..., \mathbf{\Theta}_G, \boldsymbol{\tau}\} \\ \mathbf{\Theta}_g &= \{\theta_{g1}, ..., \theta_{gn}\} \\ \boldsymbol{\tau} &= \{\tau_1, ..., \tau_G\},\ \sum_{g=1}^{G} \tau_g = 1 \end{aligned} \tag{B.10}$$

The prior term $p(\mathbf{\Theta})$ in Equation B.9 consists of Poisson parameter prior and cluster priors. This can be decomposed as:

$$p(\mathbf{\Theta}) = \prod_{g=1}^{G} \prod_{l=1}^{n} p(\theta_{gl}|\alpha_{gl}, \beta_{gl}) p(\boldsymbol{\tau}|\gamma) \tag{B.11}$$

where we use gamma priors with parameters α and β for Poisson parameters and Dirichlet priors for cluster weights :

$$\begin{aligned} p(\theta_{gl}|\alpha_{gl}, \beta_{gl}) &\propto \theta_{gl}^{\alpha_{gl}}\, e^{-\beta_{gl}\theta_{gl}} \\ p(\boldsymbol{\tau}|\gamma) &\propto \prod_{g=1}^{G} \tau_g^{\gamma_g} \end{aligned} \tag{B.12}$$

Then, the Q function in Equation B.9 can be written as:

$$\begin{aligned} Q(\mathbf{\Theta}, \mathbf{\Theta}') &= \sum_{i=1}^{K} \sum_{g=1}^{G} P_{ig}(\mathbf{\Theta}') \left[\ln p(\mathbf{x}_i|c_g, \mathbf{\Theta}_g) + \ln \tau_g\right] \\ &+ \sum_{g=1}^{G} \sum_{l=1}^{n} (\alpha_{gl} \ln \theta_{gl} - \beta_{gl}\theta_{gl}) + \sum_{g=1}^{G} \gamma_g \ln \tau_g \end{aligned} \tag{B.13}$$

To calculate the optimal parameters we maximize the Q function in Equation B.13 subject to the constraint that cluster priors sum to 1:

$$\begin{aligned} \frac{\partial}{\partial \tau_g} \left[Q(\mathbf{\Theta}, \mathbf{\Theta}') - \lambda \sum_{j=1}^{G} \tau_j \right] &= 0 \\ \sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') \left[\frac{1}{\tau_g} \right] + \frac{\gamma_g}{\tau_g} - \lambda &= 0 \end{aligned} \tag{B.14}$$

from which it follows:

$$\lambda\tau_g = \sum_{i=1}^{K} P_{ig} + \gamma_g \tag{B.15}$$

Summing Equation B.15 over g we obtain:

$$\lambda = \sum_{j=1}^{G} \left[\sum_{i=1}^{K} P_{ij} + \gamma_j \right] \tag{B.16}$$

Upon substituting Equation B.16 into Equation B.15 and solving cluster priors, we obtain the update equation for cluster weights:

$$\widehat{\tau}_g = \frac{\sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') + \gamma_g}{\sum_{j=1}^{G} \left[\sum_{i=1}^{K} P_{ij}(\mathbf{\Theta}') + \gamma_j \right]} \tag{B.17}$$

Optimizing the Q-function with respect to Poisson parameters we obtain:

$$\begin{aligned} \frac{\partial}{\partial\theta_{gm}} \left[Q(\mathbf{\Theta}, \mathbf{\Theta}') \right] &= 0 \\ \sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') \left[\frac{x_{im}}{\theta_{gm}} - 1 \right] + \frac{\alpha_{gm}}{\theta_{gm}} - \beta_{gm} &= 0 \end{aligned} \tag{B.18}$$

Equation B.18 can be solved for θ_{gm} to obtain update equation for Poisson parameters as follows:

$$\widehat{\theta}_{gm} = \frac{\sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') x_{im} + \alpha_{gm}}{\sum_{i=1}^{K} P_{ig}(\mathbf{\Theta}') + \beta_{gm}} \tag{B.19}$$

C. Appendix: Source Code for the Construction of CST

```
package ca.uwaterloo.tree;
/**
 * Node of the CST
 * Creation date: (12/30/2002 3:26:05 PM)
 * @author: Sule Gunduz
 */
 import java.util.Vector;
public class Node implements java.io.Serializable, Comparable {
    public Object n_Data; //Data field
    public Vector n_Children = new Vector();
                //indefinite number of children
    public int n_Count = 0; //count field
    public java.lang.Object n_NextNode;
                //links to the next node with the same data field
    public java.lang.Object n_ParentNode; //links to the Parent Node
public Node() {
    super();
    this.n_Count = 0;
    this.n_Data = null;
} /**
 * Creates a node with specified data field and set the count to 1
 * @param data java.lang.Object
 */
public Node(Object data) {
    this.n_Data = data;
    this.n_Count = 1;
} /**
 * set the data and count fields to specified values
 * @param data java.lang.Object
 * @param count int
 */
public Node(Object data, int count) {
    this.n_Data = data;
    this.n_Count = count;
} /**
```

```
 * add a child with data field and returns child
 * @param data java.lang.Object
 */
public Node addChild(Object data) {
    Node tempNode = new Node(data);
    n_Children.add(tempNode);
    tempNode.n_ParentNode = this;
    return tempNode;
} /**
 * add child with data and count fields and return child
 * @return ca.uwaterloo.tree.Node
 * @param data java.lang.Object
 * @param count int
 */
public Node addChild(Object data, int count) {
    Node tempNode = new Node(data, count);
    n_Children.add(tempNode);
    tempNode.n_ParentNode = this;
    return tempNode;
} /**
 * returns the number of children
 * @return int
 */
public int degree() {
    return n_Children.size();
} /**
 * returns the child at index=index
 * @return ca.uwaterloo.tree.Node
 * @param index int
 */
public Node getChild(int index) {
    if (index >= degree() || index < 0)
        return null;
    else
        return (Node) n_Children.get(index);
} /**
 * returns the data field
 * @return java.lang.Object
 */
public Object getData() {
    return n_Data;
} /**
```

```
 * increments the count of the child at index = index
 * Creation date: (12/30/2002 4:10:01 PM)
 */
public void incrementCount(int index) {
    Node tempNode = getChild(index);
    tempNode.n_Count++;
}}/**
 * The Construction of the CST, where each
 * user session is represented as a branch of the CST
 * Creation date: (12/30/2002 3:26:38 PM)
 * @author: Sule Gunduz
 */
 import java.util.Hashtable;
 import java.util.Vector;
 import java.util.Enumeration;
 import java.util.Arrays;
public class Tree implements java.io.Serializable {
    Node t_Root;
    Hashtable t_NodeTable;
    int t_NumberOfPaths;
    int removedElements = 0;

public Tree() {
    super();
    t_NumberOfPaths = 0;  // number of sessions in the tree
    t_Root = new Node();  // root node, data = null
    t_NodeTable = new Hashtable();  // data_table
} /**
 * Each user session is a vector. Adds a user session to the tree
 * @param element java.util.Vector
 * @param node ca.uwaterloo.tree.Node
 */
public void addElement(Vector element, Node node) {
    if (element.size() == 0) {
        boolean found = true;
        return;
    }
    Node current_node;
    int step = 0;
    String data = (String) element.get(0);
    boolean found = false;
    if (node.degree() != 0) {
        while (!found && step < node.degree()) {
            if (node.getChild(step).n_Data.equals(data)) {
                node.incrementCount(step);
                current_node = node.getChild(step);
                element.remove(0);
                found = true;
                ((NodeTableClass)
                            t_NodeTable.get(data)).incrementCount();
                addElement(element, current_node);
            }
            step++;
        }}
        if (!found || node.degree() == 0) {
            current_node = node.addChild(data);
            if (t_NodeTable.get(data) != null)
                 ((NodeTableClass)
                        t_NodeTable.get(data)).addNode(current_node);
            else
               t_NodeTable.put(data, new NodeTableClass(current_node));
            element.remove(0);
            found = true;
            addElement(element, current_node);
        }
} /**
 * Returns the nodes in the tree
 */
public Enumeration elements() {
    return (new Breadth_First_Traversal(t_Root));
}}
/**
 * Performs a Breadth First Search on the tree
 * Creation date: (1/3/2003 4:11:33 PM)
 * @author:
 */
 import java.util.Vector;
 import java.util.Arrays;
        public class Breadth_First_Traversal
                implements java.util.Enumeration {
     private Vector nodes = new Vector();
     private boolean started;
     private Node root = new Node();
```

```
public Breadth_First_Traversal() {
    super();
} /**
 * Starts the search from node
 * @param node ca.uwaterloo.tree.Node
 */
public Breadth_First_Traversal(Object node) {
    root = (Node) node;
    if (root != null)
        nodes.addElement(root);
} /**
 * Tests if this enumeration contains more elements.
 * @return  <code>true</code> if and only if this enumeration object
 *           contains at least one more element to provide;
 *          <code>false</code> otherwise.
 */
public boolean hasMoreElements() {
    if (nodes.size() == 0)
        return false;
    return true;
} /**
     * Returns the next element of this enumeration if this
     * enumeration object has at least one more element to provide.
     * @return     the next element of this enumeration.
     * @exception  NoSuchElementException  if no more elements exist.
     */
public Object nextElement() {
    Object [] array;
    Vector children;
    Node  tempNode = (Node) nodes.elementAt(0);
    nodes.removeElementAt(0);
    children = tempNode.n_Children;
    array = children.toArray();
    Arrays.sort(array);
    children = new Vector();
    for (int i = 0; i < tempNode.degree(); i++){
        children.addElement((Node) array[i]);
        nodes.addElement((Node) array[i]);
    }
    tempNode.n_Children = children;
    return tempNode;
} }
```

BIOGRAPHY

Şule Gündüz Öğüdücü graduated from Istanbul Lisesi in 1987. She received her B.Sc. in 1991 from the Electronics and Communication Engineering Department of Istanbul Technical University, and her M.Sc. in 1994 from the Institute of Biomedical Engineering at Boğaziçi University. She is a research and teaching assistant at the Computer Engineering Department of Istanbul Technical University since 1998 and enrolled in the Ph.D. program of the Institute of Science and Technology of the same university in 1999. She was a visiting research scholar at the University of Waterloo, Waterloo, Canada, in Database Research Group between October 2001 and January 2003.

During her work on Ph.D. thesis, she published the following related papers:

- Şule Gündüz, M. Tamer Özsu, "A User Interest Model for Web Page Navigation," *Proc. of International Workshop on Data Mining for Actionable Knowledge (DMAK)*, Seoul, Korea, Apr. 2003, pages 46-57.
- Şule Gündüz, M. Tamer Özsu, "Recommendation Models for User Accesses to Web Pages", *Proc. of Joint Intl. Conference ICANN/ICONIP 2003*, Istanbul, Turkey, Jun. 2003, pages 1003-1010.
- Şule Gündüz, M. Tamer Özsu, "A Web Page Prediction Model Based On Click-Stream Tree Representation of User Behavior", *Proc. of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, Washington, DC, USA, Aug. 2003, pages 535-540.
- Şule Gündüz, M. Tamer Özsu, "A Poisson Model for User Accesses to Web Pages", *Proc. of Eighteenth International Symposium on Computer and Information Sciences (ISCIS XVIII)*, Antalya, Turkey, Nov. 2003, to appear.