TIME DOMAIN BASED WEB USAGE MINING FOR WEB SITE IMPROVEMENT

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Alpay Erdem

March 2002

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Attila Gürsoy (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Dr. Özgür Ulusoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Uğur Güdükbay

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

TIME DOMAIN BASED WEB USAGE MINING FOR WEB SITE IMPROVEMENT

Alpay Erdem
M.S. in Computer Engineering
Supervisors: Assist. Prof. Dr. Attila Gürsoy and
Assist. Prof. Dr. Uğur Doğrusöz
March 2002

With the increased use of Web, large volumes of click-stream data, embedded inside server logs, has become available for revealing user access patterns especially on specified Web sites. Efficient Web content presentation conveyed through links structure is a very important issue for efficient use of site. Web Usage Mining can be used to improve Web site design by finding deficiencies of the Web site by analyzing user access patterns.

Although Web sites are intended to be designed for efficient usage for typical users, mostly conceptual relations between pages and categorization proposed by Web site designer may not meet expectations of the users. Misleading Web site design leads to users spending much more time for reaching target pages by reasoning redundant paths to be followed or lost in cyber-space without finding the target. Furthermore, changing needs and interests of users by the time require re-structuring of the Web site. Therefore Web sites should be updated according to user expectations. For that reason, most popular pages should be easily accessed, conceptually related pages either should be categorized close enough or should be linked and misleading guidance directing users to different pages other than target should be detected. However, barely finding frequent sequences is not sufficient for improving a Web site. This is because of the fact that explored frequent patterns cover both interested patterns used for reaching popular sites and redundant patterns that are followed previous to reaching target page(s). Frequent backward references embed knowledge of redundant and also related pages according to interest in these pages. In order to interpret backward and forward references in terms of interest we incorporated time domain that finds page viewing timing for each visited page. Relatively spent page viewing time for each page within a session is an important interest criterion for that page.

For that purpose, we proposed a Web usage mining framework that explores deficient points in the web site design according to user expectations. Whether user reached or not indicates misleading-guidance. Besides, jumping to related pages by using long paths, in many cases backtracks, shows that those pages should be linked. However, in order to be able to capture such patterns, page view time of each page is used. This franework advises re-design suggestions for Web site improvement. In the usage processing part of this framework, all user navigation sessions are analyzed and both forward and backward references are obtained by considering cached pages and also page viewing timing is computed for each page. In the mining process and interpretation part, frequent interested and redundant patterns are explored and interpreted for enabling popular pages more visible, linking related pages, reporting misleading categorization and detecting misleading guidance or categorization.

Keywords: Web Usage Mining, Web Site Improvement, Sequence Mining, Web Mining..

ÖZET

WEB SİTE GELİŞİMİ İÇİN ZAMANA BAĞIMLI WEB KULLANIM MOTİFLERİNİN ÇIKARILMASI

Alpay Erdem
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticileri: Yrd. Doç. Dr. Attila Gürsoy and
Yrd. Doç. Dr. Uğur Doğrusöz
Mart 2002

Web kullanımının artması ile birlikte siteler üzerinde sunucu log dosyalarında saklı olan Web kullanıcı verileri kullanıcı motiflerinin incelenmesi amacına yönelik işlenebilir durumdadır. Bağlantılarla gösterilen Web site tasarımı etkin Web içerik sunumu açısından oldukça önemlidir. Web Kullanıcı Motiflerinin işlenmesi kullanıcı motiflerinin yardımıyla Web sitesindeki eksiklikleri tespit ederek Web yapısının iyileştirilmesinde kullanılabilir.

Web siteleri ortalama kullanıcıların etkin bir şekilde kullanabilmeleri için tasarlanmış olsa da sayfalar arasındaki kavramsal bağıntılar ve sayfaların sınıflandırılması kullanıcıların beklentilerini karşılamayabilir. Yanlış yönlendirmeler içeren Web sayfa tasarımı kullanıcıların yanlış yollar denemesine sebep olarak ulaşmak istedikleri hedef sayfalara çok daha fazla zaman harcayarak ulaşması veya "sanal-boşlukta" hedefine ulaşamaması ile neticelenir. Bununla birlikte kullanıcıların zamanla ilgi odaklarının değişmesi ile birlikte kullanıcı ihtiyaçlarına yönelik olarak Web sitesi tekrar düzenlenmelidir. Bu nedenle, en çok ziyaret edilen popüler sayfalar kolaylıkla ulaşabilmeli, kavramsal olarak birbirine yakın olan sayfalar birlikte kategorize edilmeli veya birbirlerine bağlantılarla ulaşabilmeli ve hatalı yönlendirici içeriğe sahip sayfalar tespit edilmelidir. Bununla birlikte yalnızca sık sıralamaların bulunmasi Web sitesinin geliştirilmesi için yeterli değildir. Çünkü sık sıralamalar ilgi duyulan sayfaları içerdiği gibi kullancının hedefine ulaşırken denemiş olduğu gereksiz sıralamaları da içerir. Geri sık sıralamalar gereksiz yere izlenen sıralamaları ve aynı zamanda ilgili sayfaların ilgilenilen veya ilgilenilmeyen sayfalar olmalarına göre farklılaşan bilgileri içerir. Bu tezde, ileri ve geri sık sıralamaların yorumlanması amacı ile her sayfa için sayfanın izlenme süresini bulmada kullanılan zaman boyutunu sıralamalara ekledik. Her sayfada bir sıralama dizisi içinde diğer sayfalara oranla sayfanın izlenmesi için harcanan zaman o sayfaya kullanıcının ilgi kriteri olarak kullanılabilir.

Bu amaçla, Web sitesinin iyileştirilmesine yönelik tekrar tasarlanması için önerilerde bulunan Web kullanım motifleri işleyen bir yöntem önerilmiştir. Kullanıcının hedefine ulaşıp ulaşmadığı hatalı yönlendirmenin bulunması açısından önemlidir. Bununla birlikte, çoğu zaman geri dönüşlerin kullanımı ile uzun yollar izlenerek ilgili diğer sayfalara geçiş bu sayfaların bağlanması gerektiğini gösterir. Ancak bu şekildeki motiflerin yakalanabilmesi için zaman boyutunun kullanılması gerekmektedir. Kullanıcı motiflerinin işlenmesi kısmında tüm kullanıcı sayfa ziyaret sıralamaları ileri ve geri sıralamalar da dahil olmak üzere ve sayfa istem havuzundan getirilen sayfaların bulunması ve her sayfa üzerinde harcanan zamanın bulunması problemlerinin göz önüne alındığı sıralamalar bir bütün olarak incelenmiştir. Popüler sayfalara kısayol ekleme, ilgili sayfaların bağlanması ve hatalı yönlendirici sayfaların bulunması amacı ile sık motiflerin buluması ve yorumlanması kısmında ilgi kullanıcılar tarafından ilgi duyulan motifler ve gereksiz olanlar bulunmuştur.

Anahtar sözcükler: Web Kullanıcı Motifleri, Web Sitesi Gelişimi, Sıralama Çıkarımı, Web Veri Çıkarımı..

Acknowledgement

I would like to express my special thanks and gratitude to Dr. Attila Gürsoy and Dr. Uğur Doğrusöz due their supervision, interest and support during my graduate work.

I am also indebted to Prof. Dr. H. Altay Güvenir for his support during my research and Assoc. Prof. Dr. Özgür Ulusoy and Assist. Prof. Dr. Uğur Güdükbay for accepting to read review this thesis.

I would like to thank my parents for their morale support.

I acknowledge all the honorable faculty members of CS department and my dear friends and colleagues for their help. My special thanks go to Assoc. Prof. Dr. M. J. Zaki for the supply of his Spade implementation and his interest in my subject.

I also would like to thank TÜBİTAK-UEKAE for their invaluable support and tolerance during my graduate education.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Web usage mining process deals with click-stream data in order to extract usage patterns from recorded foot-prints of the users. This process is very crucial for especially e-commerce activity and also for other organizations providing efficient use of cyber-space to users. Efficient use of Web site by users depends on how site content is organized for average user profile. In other words user expectation from a Web site should meet site structure and content design. Structural design of the Web that is hierarchical content organization and links indicating relatedness of these contents determines "content presentation" of the site. Average users expect content to be organized hierarchically and related contents placed close enough or referred by each other. Although site designer tries for a design to provide efficient access, average user expectation due to content presentation issue may not meet with site design. Therefore site should be re-structured according to user expectations.

In order to link related pages and also find misleading pages we have to analyze all navigation process of the user including forward and backward references. Forward references are the paths followed to in forward direction, through hierarchically more detailed levels, and backward references are the vice versa. Frequent backward traversals and points connecting backward and forward references give important information about deficiencies of the site.

In addition to exploring deficiencies in Web site, site structure can be improved by also finding frequently interested pages and shortening paths reaching to such pages. Although favorite pages are also subject to hierarchical placement

inside cyber-space, without spoiling position of such pages we can add guiding link from higher levels to current position. By this way effort for reaching frequently interested pages would be minimized.

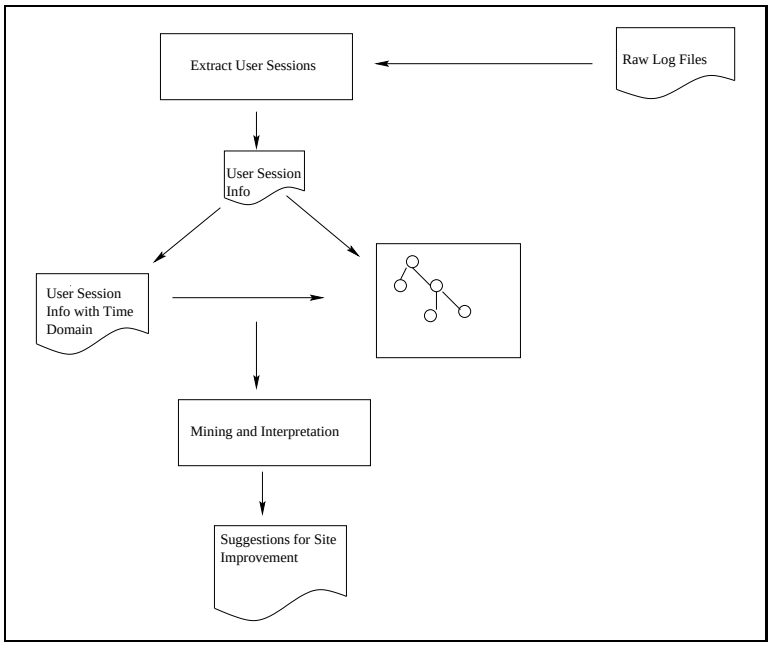

Figure 1.1: Flow Chart of Usage Mining Process.

Although Web sequence data embeds useful temporal knowledge inside it, due to problems stated above there may be redundant traversals or trials since users could not reach their target pages. Besides, some pages are accessed just since they are on the path reaching to target page. Therefore we proposed to distinguish interested and non-interested page accesses, that is "page access" may not indicate interest of a user in that page. In order to find out interest of user in a page we incorporated time domain into temporal knowledge. Relative spent time for viewing a page indicates interest criterion. We interpreted temporal knowledge together with the time domain.

This issue is not addressed in most of the Web usage mining approaches and others directly used page access timings for time domain. However difference between successive page access timings can not be considered as interest criterion since that duration covers both page download time and page view time. Especially this is very crucial if a page has graph dense content. Download time may be much larger than the view time. With previous approach such pages may be considered as interested pages although they may not be. Therefore

page viewing timings should be computed successfully for correct interpretation. However page view time computation and clear extraction of user sessions is difficult task from a limited data source, Web log file. Although it becomes very easy to collect required data by adding some software to client or server part, this is not a common way whereas standard Web server logs are kept for most Web servers and are available.

In this thesis, we attacked problem of interpretation of frequent Web sequence patterns by incorporating time domain and used as interest criterion for higher level of interpretation. We distinguished frequently interested and non-interested paths in order to improve site structure design. We utilized site link map structure together with user session information to find improvement suggestions. Flow chart of the usage mining process is given in Figure 1.1 briefly. In the first step, raw lof files are processed so that user sessions are extracted by considering multi-user sessions. After that, page view timings are computed and embedded into session information. As the next step, site link map is constructed by making a pass over the all extracted sessions for preparing infrastructure to interpretation part. After site link map is constructed, forward and backward traversals are followed over site link map and by using time domain. During this process, forward shortcut info, related page info and misleading guidance info is embedded into the nodes of the site link map. As the last step, this embedded info is processed and a set of suggestions for web site improvement is extracted.

1.2 Outline of the Thesis

Thesis organization is given briefly. We introduce Web usage mining and temporal knowledge approaches in Chapter 2. In Chapter 3, we presented problem definition and motivation for our proposed framework. In Chapter 4, we give implementation details of usage processing and infrastructure for mining and interpretation part which are explained in Chapter 5. Chapter 6 includes experimental results. We conclude our thesis and give suggestions for future work in Chapter 7.

Chapter 2

RELATED WORK

In this section we will briefly overview existing work on Web Usage Mining and Temporal Knowledge Discovery. Web Usage Mining mainly focuses on extracting patterns from Web data and Temporal Knowledge Discovery deals with extracting relations from order of data. We used Temporal Knowledge Discovery as base for modeling Web data.

Web mining can be categorized into three parts [1]: Web content mining, Web structure mining and Web usage mining. Web structure mining involves mining the Web document's structures and links. Web content mining describes the automatic search of information resources available on-line. Web usage mining deals with extraction of user access behaviors from Web server logs.

In general, Web mining can answers some interesting questions:

1. Measuring the completeness of the Web site with respect to that most of the related information are available at the same Web site. For example an airline's home page will have more than local links connecting the "routing" information with air-fares and schedules' than the external links.

2. Measuring the flow of information. We can analyze whether relevant information is kept at the same site.

3. Expressing research papers' ability to cross-reference to other related work. This is used for measuring visibility of the Web documents and ability to relate similar or related documents across different sites.

Providing guidance to user about the most probable paths that can be followed by considering previously followed paths will save timing for searching related documents. Because, for example 68 % of users who previously followed the same path choses to follow Y path. Those pages having greater than a minimum relevance threshold can be displayed for guidance to user.

4. Analyzing flow of information and revealing how they are conceptually related. This can help generalizing flow of information.

5. Analyzing nodes with high in-degree or out-degree may help finding out interesting pages. A node with high-degree may indicate importance of that site, and reverse may be sign of that site a luminous site.

2.1 Web Usage Mining

Web usage mining mainly is composed of General Web Usage Mining, Site Modification, System Improvement and Personalization parts. General Web Usage Mining mainly focuses on extracting general usage patterns from Web logs by analyzing paths followed by user or user groups. Site Modification explores gaps between developers' design and the actual needs of users and suggests modifications on structure and content in order to close this gap. System Improvement deals with optimizing Web traffic with use of Web usage mining. Personalization is the process of customizing Web sites according to needs of specific group of customers. Important aspects of related projects are going to be introduced.

2.1.1 General Web Usage Mining

This subsection contains General Web Usage Mining approaches.

A- WEBMINER

Webminer is implemented as a framework for discovery of association rules and discovery of sequential patterns [2]. In the first part of WebMiner, association rules,presence of some items implies presence of some other items,are extracted.

Examples to intended association rules are like the followings;

60 % of customers who accessed product index page also analyzed `product1` or 40 % of customers who accessed `product1` also accessed the `product2`.

For sequential patterns;

30 % customers who visited `/company/products/product1.html`, had done a search in Yahoo, within the past week on keywords `w1` and `w2`

60 % of customers who placed an on-line order in `product1.html`, also placed an on-line order in `product4` in 15 days.

May be interested in finding common characteristics of all clients that visited a particular site within the time period [t1,t2].

WebMiner uses pruning technique based on site organization information in the following way. Since site organization is previously known and that particular site is organized in a hierarchical way, this information can be used in pruning.For example if search for product index is low, than it can be concluded that search for a specific product belonging that index would be also have low support.

WebMiner also utilize false guidance in order to give guidance for restructuring Web site. For example, If page B is accessed after page B frequently, than there should some information directing users to page B. If page is not the index page than information in A should be moved to higher levels, that is to index page.

In Webminer, for discovering association rules: apriori algorithm is used. For sequential patterns slightly different version in candidate generation phase of apriori algorithm is used. Checks subsequence occurrence in sequences and also webminer integrated data mining query language with data mining process. User may be interested in finding such queries: starting with A, containing B and C in that order. $A * B * C$. This can be also extended and also query can be optimized.

WebMiner suggest that obtained results can be evaluated as the following:
If some patterns are not desirable, then direct links can be placed in corresponding pages for re-directing the traffic.
If a pattern has high confidence, it may be useful to put certain kind of information in the predicted path of clients corresponding to that pattern.

B- Extracting Maximal Forward References

Chen et al. introduce the concept of using the maximal forward references in order to break down user sessions into transactions for the mining of traversal patterns [3]. Approaches on mining traversal paths are mainly decomposed into two category. The first approach does not make utilize backward references in interpretation of sessions in terms of constituting part of the session and considers backward references as beginning of new maximal forward sessions. However, the other approach takes backward references into consideration due to their indication of some sites leading false guidance or having poor context. Approach of Chen et al. falls into the first category. A maximal forward reference is the last page requested by a user before backtracking occurs, where the user requests a page previously viewed during that particular session. Backward references are considered as paths followed in order to reach other Web pages where there is no direct link from the current location.

First deficient point of the algorithm is ignoring backward references although crucial information about Web site design deficiencies is embedded in backward references. Furthermore in constructing conceptual relevance between sites can be found out via combination of forward and backward sequences. So, backward sequences should be considered as a part of user access patterns. Otherwise finding discrete frequent sequences may not reflect the actual user interested path profile. According to the proposed algorithm, forward sequences terminate as backward sequences begin. After a forward sequence terminates, it backtracks to the starting point and a new forward sequence starts. Backward sequences are not utilized. Another deficient point is that there may irrelevant site accesses exist due to misleading pages that provides poor guidance.

In the first phase of the algorithm, original sequence of log data is transformed into a set of traversal subsequences with respect to users. Then large reference sequences are found by an apriori-like algorithm where order is considered. In determination of maximal traversal patterns two different algorithms are used. $FullScan$ (FS) is similar to $DirectHashingPruning$ (DHP) other than considering order of transactions. Selective Scan is the second algorithm which reduces I/O cost by not scanning the database to obtain a large reference set in each pass. Frequent large reference sequences should be consecutive and cannot be subsequences of other sequences.

C- WUM

Although *WebMiner* explores frequent patterns, it does not support introduction of specifications on the structure of patterns. This drawback is addressed by *WUM* by enabling user define interest criteria for navigation patterns dynamically [4].

Interest criteria cannot bu fully determined by simply giving support-confidence and length of the patterns. Because statistical dominance is not always of interest, and statistically dominant patterns rarely brings new knowledge to the expert. Although [32] approach allows user to give instructions to the miner it does not utilize aggregate tree-like structure and that leads severe decrease in performance.

Besides, *WUM*'s approach considers backward traversals as the potential indicators of guided or explorative tours or disorientation instead of interpreting them just start of new sessions. In the first phase of the mining process, log data is divided into transactions representing continuous path trails of each user. These transactions are merged as common prefixes through a trie structure where each node in the aggregated tree represent an URL and number of occurrences of paths from root to the specific node is kept also. After aggregated tree is constructed, user can specify structural, textual and statistical constraints on the mining result interactively in *MINT* syntax. *MINT* syntax also allows specification of beliefs or belief violations as predicates.

A "navigation" tree is a generalization of the aggregate tree constructed according to pattern descriptor. A "pattern descriptor" is a sequence of identifiers and wildcards, where an identifier refers to an occurrence of a Web page. In this part, the structural and textual constraints are considered. The navigation pattern is built by discovering the tree branches in the Aggregated Log that conforms to the pattern descriptor and merging them at the common prefixes and the nodes bearing the identifiers referenced in the descriptor. The support of those new nodes are computed by adding up the supports of the merged nodes. In this part conformance to statistical constraints are checked. In this approach aggregated tree is constructed once and can be updated with new trails. In this way performance of the system is increased instead of processing raw log files and considering much more candidate templates. However as queries become complex, time spend for matching the specified template significantly increases.

D- Suffix Based Approach, WAP (Wap Access Patterns)

Web Access Pattern (WAP) Mine approach extracts frequents access sequences without considering any interest criteria from Wap Access Pattern (WAP) Tree, which is an compact tree constructed from Web log [5]. In construction of $WAPTreeWebLog$ is traversed twice, in the first pass all non-frequent 1 sequences are pruned and frequent subsequences are obtained. In a similar fashion to aggregate log, WAP Tree is constructed by merging frequent access sequences on their common prefixes. In order to ease node traversal in a WAP tree, all nodes with the same level are linked into a queue and the head of each queue is kept in an index.

$WAPMine$ uses conditional search to mine all Web access patterns. Conditional search requires only examination of access patterns with the same suffix. Suffix is used as the condition to narrow the search space, especially as suffix becomes longer, search space becomes more compact. Suffix Heuristic proposes that if an event (page reference) e is frequent in the prefixes of sequences that have a suffix which contain pattern P as a subsequence, then eP is also a pattern.

E- Hypertext Probabilistic Approach

Borges et al. proposed a model of hypertext to capture user preferences as navigating through the Web [6]. Proposed approach makes use of previous knowledge of visited N pages to predict most probable page that would be chosen by user named as N-grammer model. Main motivation for this approach is that user has a limited memory of the previously browsed pages and that the next choice depends only on the last N pages browsed. The user navigation sessions inferred from the log data are modeled as a hypertext probabilistic language generated by hypertext probabilistic grammar (HPG). In HPG model, there exists one-to-one mapping between terminal and non-terminal symbols produced from user sessions where each non-terminal symbol corresponds to a Web page and a production rule corresponds to a link between pages. By using HPG model probabilistic value is computed based on the link corresponding to a production was chosen from the links on a page represented by that state. Authors also propose use of entropy as an estimator of the statistical properties of the grammer.

After the model is constructed, an exhaustive search of all the strings with the required characteristics is performed by employing a special case of a directed

graph Depth-First Search. Deficiency of the algorithm is that although this approach is valid for correctly followed paths, it is not the case that user follows always the most efficient paths. Some of them may be redundant paths just used for reaching target pages. Such redundant paths are not distinguished from the targeted pages in this approach.

F- Generalized Path Approach

Those approaches considering Web log patterns as the problem of finding association rules ignore Web site structure and may find extra relations that does not exist. Maximal Forward Sequence Mining approach concentrates on finding frequent consecutive sequences, however this approach is sensitive to noise that corrupts continuous patterns. Random accesses or dummy paths followed to reach other sites may constitute noise and this may prevent discovering most of the patterns interrupted by such kind of noise. This decreases support of such patterns and they cannot be determined. Mainly these two problems are addressed by Nanopolous et al. and they proposed Apriori-like algorithm that consider such kind of noise and make use of Web structure information in candidate process phase [7].

First distinguishing point of this algorithm from Apriori is that candidates have to be paths in he graph and not the arbitrary combinations of vertices that narrows down search space of candidates. The second point is that generation of next phase candidates conforms to constructing new valid paths in the graph from the existing ones.

G- WEBSIFT

Cooley et al. proposed WebSift system that identifies interesting pattern [8]. They developed a general quantiative model of factors determining interest level of discovered knowledge, and supported automatic generation of an initial set of evidence about a belief set. They also developed specific algorithms for automated discovery of interesting rules in the Web Usage Mining domain.

Pre-processing phase of the algorithm include identifying users, server sessions, removing irrelevant data and completing missing paths due to cache referrals. WebSift also uses the structure and content information about a Web site

to automatically define a belief set. The information filter uses this belief set to identify results that are potentially interesting [8].

Belief Mined Evidence (BME) and BCE (Beliefs with Contradicting Evidence) techniques are used in order to define interest criterion. BME declares itemset that contain pages not directly connected to be interesting. This corresponds to that a belief set of pages are related has no domain or existing evidence, but there is mined evidence. In BCE, the absence of certain frequent itemsets is interpreted by the information filter as evidence against a belief that pages are related. The pages that are linked, but not in the same frequent itemset may also be interesting and such redundant links can be removed.

H- OLAP Approach

Zaine et al. developed WebLogMiner that uses OLAP technology for mining processes through data cube structure [9]. Data accumulated in data cube in different abstract levels that are determined by analyst can be used as input to data mining algorithms.

WeblogMiner considers following issues that have attracted less attention. Web logs do not record backtracking and reloading the same page if the source is cached by the browser or a proxy. However frequent backtracking and reloading indicate deficient design of the site navigation. Second issue is that Web access counts can not be considered only metrics as indicators of user interest or interest of the Web site. Because documents browsed just for reaching another document get hit although they are not the intended pages but just browsed since they are on the path of desired document. Therefore access counts should be considered with some other metrics. Third important point is that date and time collected for each successive request can give interesting clues regarding the user interest by evaluating the time spent by users on each resource.

Data cube, a multi-dimensional array structure, is kept to aggregate hit counts. Each dimension represents a distinct property of data and can be determined according to abstraction levels of data properties. In this way, data cube provides flexibility to process the data and view it from different perspectives.

On data cube $OLAP$ operations such as drill-down, roll-up, slice, dice and data mining techniques association, prediction, classification and time-series analysis can be performed.

I- Clustering of Web Users Based on Access Patterns

Fu et al. proposed clustering Web users based on common properties [10]. Users having same browsing patterns are classified into the same class. For example, if a number of customers spend quite a lot time on browsing pages about "baby furniture", "baby toys", and "diapers", they may be clustered into "expecting parents". Users observed in different classes can be analyzed better and more suitable services can be proposed to each class of users. Deficient point of this approach is that not detailed classification can be expected due to dimensionality problem.

In the first step log data is divided into sessions where each session is represented pairs of as page and time spent on that page. The sessions are identified by grouping consecutive pages requested by the same user together. Due to vector representation of the pages, when number of pages becomes large, it is stated that in [10] quality of clusters and efficiency of the clustering algorithm. In order to overcome this problem, Attribute Oriented Induction method is used through generalization of page hierarchy. Hierarchical structure of pages is constructed according to directory structure of the server. Leaf nodes corresponds to URL's. Level of generalization of sessions is determined by mining expert via Tree Climbing through Attribute Induction method. In this process, pages are replaced with ascendants on page hierarchy.

In the last phase, generalized pages are clustered using $BIRCH$ (Balanced Iterative Reducing and Clustering using Hierarchies). $BIRCH$ builds a Clustering Feature (CF) tree as the result of clustering by incrementally inserting objects. A $CFTree$ is a multidimensional structure like B+ tree. Leaf nodes contain the collection of session vectors constituting clusters and non-leaf nodes store CF vectors where a CF vector contains properties of the clusters belonging to them.

J- Categorization of Users according to Navigation Paths

Shahabi et al. proposed a profiler which collects detailed and accurate Web navigation information of users on a Web site [11]. This approach considers four additional information about users in addition to other approaches in the same category; access page order, link access, cache reference and accurate page viewing time. Page access frequency does not give adequate information to capture user navigation behavior but access order is very important. Because users requesting similar pages may have followed different paths and that indicates different user interests and they should be grouped into different categories and also time spent by user viewing a page crucial in determination of user interest in that page.

Profiler is a Java applet that works on the client side and determines exact viewing time of the pages and catching cache requests that are not recorded on logs otherwise. Paths and time durations are explored through this profiler and they are clustered by Path-Mining method. In addition to this, link information is gained by updating each link of each page in order to convey this information to server side. By this way link information between page requests are added. In clustering phase, visitors with same kind of interests are grouped by using Path-Mining method. This method computes similarities of paths basing on feature space process.

K- Web Sequence Mining

In the approach of Putin et al. Web logs are kept in LOGML and XGMML formats in order to enhance Web mining [12]. Main motivation for this is that graphs can be defined as data objects whose elements are nodes and links. Zaki introduced Frequent Pattern Mining (FPM) which ranges from mining simple sequences to mining trees and graphs [12]. Association mining and sequence mining are both employed in FPM. Finding complex patters like frequent subtrees, frequent DAGs and directed or undirected subgraphs provides more information than finding just frequent itemsets or sequences.

In the proposed algorithm, backward references are also taken into account for extracting complex structures. Backward references does not indicate start of a new session. Each page content is assumed to represent a sequence item and so all page accesses are strictly ordered and there are no two pages that are

viewed by a user. In finding frequent sequences, all user-clicks are defined with unique ID's so that all order information is kept. First, frequent sequences or paths are found and finally frequently traversed subtrees are extracted basing on frequent sequence information.

Deficient point of Zaki's approach is that it just finds frequent sequences or more complex patterns. However, there is no work for finding interesting patterns and also not all frequent sequences are frequent. Besides, there is no work with regard to interpretation of frequent sequences or non-frequent sequences that may have valuable information.

2.1.2 System Improvement

System Improvement deals with increasing quality of services such as Web traffic behavior that can be used for developing policies for Web caching, network transmission, load balancing, response time to user [8].

Almedia et al. [13] make use of locality information that will be used for developing pre-fetching and caching strategies for the proxy server. Locality measure is determined by use of knowledge about user profile, that is users accessing from the same proxy server. Deficient point of this system is that increasing use of dynamic content reduces benefits of caching at both the client and server level.

Schecter et al.[14] proposed algorithms for constructing path profiles from Web logs. Path profile covers set of paths followed by users with counts. These profiles are used to pre-generate dynamic HTML pages by using current user profile. This provides performance improvement due to prevention of latency duration of page generation.

In the proposed system, the paths are stored in a tree structure that is constructed from paths whose maximal prefix is seen in at least T of the paths for optimization memory cost. After path profile construction, guessing of next access of the user is performed. Suffix part of the current user path is tried to be matched a path whose maximal prefix matches with it. When a match is obtained, suffix size is increased. Another method proposed by Schecter et al.

is use of Point Based Prediction method. This approach considers only the last page instead of the whole path.

2.1.3 Site Modification

This section covers approaches usage mining process for improving site design.

A- Adaptive Web Sites

Site modification covers Web site design improvement in terms of both content and structure since attractiveness and speedy access of a Web site are important criterion. Re-design of the sites could be performed as customization for individuals or static changes that can be seen by all visitors.

Perkowitz et al. defined "index page synthesis" problem that is the automatic creation of pages that facilitate a visitor's navigation of a Web site [15]. Their approach to solve this problem, IndexFinder proposes non-destructive transformations on structure of a Web site; adding and highlighting links, promoting links to the front page, cross-linking related pages and creating new pages of related links [15], [16]. In construction of index pages "visit-coherence assumption" is used which states that the pages a user visits during one interaction with the site tend to be conceptually related.

PageGather algorithm finds currently unlinked but conceptually related pages and improves site design by adding automatically created index pages. Instead of attempting to partition the entire space of documents, this approach try to find a small number of high quality clusters. In the first step of this approach, after log is divided into visits co-occurence frequencies of between each pair of pages are computed and similarity matrix is created. Then, a graph is constructed corresponding to similarity matrix. In this graph, pages are represented as nodes and co-occurence values as considered as links between the nodes. Connected components appeared in the graph are assumed to be clusters. However clusters may still have conceptually unrelated pages. To handle this problem, Concept Learning Algorithm finds the most common and basic concepts that summarizes in the cluster [15]. At the end of this process, conflicting pages to

conceptually related pages are removed and unlinked pages that appear conceptually related pages are linked. At the last step, index pages having links to its cluster nodes is constructed.

B- Navigation Analysis Tool based on the Correlation between Contents Distribution and Access Patterns

Nakayama et al. analyzed the gap between user behavior and Web site designer expectation by comparing inter-page conceptual relevance and link connectivity with inter-page access co-occurence [17]. They measured the inter-page conceptual relevance by the vector space model based on word frequency. User traversal ratio of each page in the concept is computed by the ratio of the number of users who visited the start page and the number of users who visited each page in the concept after visiting the start page. In the next step, the correlation coefficient between user inter-page conceptual relevance and user traversal ratio about the each page in the concept are found. If two pages co-occur in many user access patterns, a new hyperlink can be added if the path between these pages is long.User access paths are visualized in polar coordinates. Tool extracts maximal forward references by keeping orders of page accesses. Because
$A \rightarrow B \rightarrow C$, $A \rightarrow C \rightarrow B$ are different sequences for path completion.

2.1.4 Personalization

Anticipating the needs of the user and providing the right information is crucial especially in e-commerce for individualized marketing and also for many other applications. In order to increase attractiveness of a site, understanding the user interests and responding accordingly is required. This process is defined to be Web personalization. In this process, dynamic recommendations are produced based on user profile and Web usage behavior and Web site content is updated to guide each user respectively.

A- WebWatcher: Content Based Filtering

One of the most important problems in Web navigation is the difficulty in finding relevant material efficiently. Therefore Web visitors need some guiding mechanism. *WebWatcher* provides such a guidance for users during navigation based on their interests [18]. At the beginning, *WebWatcher* gets user interests

via keywords and starts with listing potentially interesting titles based on keywords. Page requests are evaluated in a central proxy server in order to easily track user sessions across multiple Web sites and mark interesting links.

WebWatcher learns from its users' actions and becomes more experienced over the time for the types of topics in which previous visitors have had an interest [18]. Previous user actions, the successful ones, are used as training examples in the learning process. *WebWatcher* generalizes from such examples to improve its ability to suggest appropriate hyperlinks for subsequent users.

Three learning methods are proposed; learning from previous tours, learning from hypertext structure and combination of these methods. The most effective learning method is multi-strategy method that rates the significance of a hyperlink based on a combination of factors: the frequency of a hyperlink was taken in the past, the stated interests of previous visitors who took this hyperlink, the words occurring on pages downstream from the hyperlink, and the underlined words associated directly with the hyperlink.

B- Usage Based Web Personalization

There appears two tracks in personalization of Web pages: traditional methods that are static and mostly biased, and dynamic personalization that make use of current user access patterns in addition to old ones. Dynamic approach considers changes in user profiles so it is superior to static personalization approach.

In the offline part of the personalization process, common usage profiles of the Web site are explored by applying data mining techniques on Web logs. In the on-line part, user profile is found based on its usage patterns so far and current Web page is updated according to user's profile.

Analog clusters users in the first step and then recommends links based on pages that retrieved previously by user [19]. Active user sessions are represented as N dimensional vectors where N is the number of distinct pages in the site. Session vector of each user is updated as he retrieves a new page and active user session is matched to existing clusters. Previously non-requested matching clusters are recommended to the user.

Web Personalizer, proposed by Mobasher et al., falls in dynamic generation of personalization [20]. The off-line part includes data preparation tasks and discovery of association rules and the derivation of URL clusters based on two types of clustering methods. In the on-line part, the frequent itemsets and the URL clusters are used to provide recommendations to users based on their current navigational activity. In the first clustering approach, traditional methods are used in which each user session is represented as N dimensional vector. N is the number of distinct URLs that exist in the user sessions. At the end of computation for clustering, URLs with low supports are filtered. The second approach proposes clustering URLs instead of sessions because users that have very different sessions may have common interest to a group of URLs. Association Rule Hypergraph Partitioning (ARHP) is used in order to compensate overhead introduced by size of feature space. In the recommendation process, sliding window technique is used of size n over the active sessions that is lastly visited n pages influence the recommendation value of items in the recommendation set [20]. Besides, physical link path from the current active session window to the candidate URL for recommendation is considered as another factor which is handled by keeping site structure as a directed graph.

At the end, on-line part recommend some links to user with significance scores. This is achieved by finding the match between current user session and aggregate usage profiles.

2.2 Temporal Knowledge Discovery

Especially time dimension in data mining process introduces time-dependent relationships that defines cause and effects. Interpretation of extracted patterns significantly varies with time attribute. However, time attribute becomes more informative when it is added to data mining steps not just as simple numeric values but information source. It also provides interpretation about activity progress of events rather than just revealing states. In a sense, incorporating temporal component leads higher order mining.

Temporal data mining is an important extension as it has the capability rather than just states and, thus, inferring relationships of contextual and temporal

proximity, some of which may also indicate a cause-effect association [21]. We will analyze main branches under temporal knowledge discovery. Taxonomy is done according to "A Survey of Temporal Knowledge" [21].

2.2.1 Apriori-Like Discovery of Association Rules

This part considers extraction of rules based on Apriori-like approaches from temporal data. Apriori-like algorithms bases on basic Apriori algorithm that is proposed by Agrawal.

A- Temporal Associaton Rules

In discovery process of association rules time ordering is not considered as an important information source. As it is stated in [21], temporal association rule mining can be used in: Temporal association mining distinguishes from association mining in that it make use of ordering information. Basically finding just correlation that conveys information that occurrence of one item is related with occurrence of another item gives poor knowledge. Because, in general it is very much different that occurrence of A is before occurrence of B with vice versa and also this interpretation appears to be totally different from considering occurrence of A and B in any order. Factors affecting first order may be different than factors affecting second imposed order:

- previous events may have a role in occurrence of later event,
- there is a third event that causes other two events, and
- the confluence of events is coincidental.

Association rule discovery is especially used in retail-kind of activities where correlations between items included in a large transaction are explored without use of explicit time histories. Temporal association rule considers time histories in determining correlation behavior between items over time periods. This approach finds more detailed and informative relationships. As an example, such a relation can be found between items: if existence of item X tends to disappear while an increase in item Y is observed, than increase in item Y can be correlated with decrease in item X.

A "casual rule" is described as a relationship in which in one part of the modeled reality cause subsequent changes in other parts of the domain [22]. Casual rules are especially crucial in finding candidate factors of that may cause certain medical conditions. Analyzing past history of patients for finding possible factors that may cause current illness may be given an example.

2.2.2 Template-Based Mining for Sequences

Time stamped values from ordinal domains, such as stock prices and population, forms curves and constitutes time-series and provides opportunity to analyze of trends.

A- Describing and Discovering Common Trends in Time Series

Trend discovery based on comparison of continuous time series considering some similarity metric that is domain specific. Basic idea is making predictions about future of one trend by analyzing a similar trend with a relaxed time delay. Time series analyzes has such following applications:

- identify companies with similar pattern of growth,
- determine products with similar selling patterns,
- discover Stocks with similar price movements, and
- find portions of seismic waves that are not similar to spot geological irregularities.

First problem in trend analyze is that time series should be reduced into the same domain in order to compare them. In order to adhere time series in different domains dynamic Fourier transforms are applied [23]. Another issue is exploring significant relations between events in a reasonable time window which is defined as proximity. An example is that relating of trend in decrease of element a to trend in increase in element B in 1 year.

Agrawal et al. decomposed the problem into three parts and proposed three different algorithms. Two sequences are defined to be similar if they have sufficient non-overlapping pairs of similar subsequences. For gap-free sequences this

definition is extended as if one lies within an envelope of a specified width around other, after ignoring outliers [24]. In the first part of the proposed matching system a self join is applied on the data set and an R-tree is used for efficient search. In the second part, a fast algorithm is employed for stitching atomic matches to form long subsequence matches, allowing non-matching gaps exist between the atomic matches. This problem is reduced to finding longest path in an acyclic graph. The third stage linearly orders the subsequence matches to determine if enough similar pieces exist in the two subsequences.

Another work is about finding common trends by defining "shapes". Agrawal et al. proposed a shape definition language (SDL) to describe patterns or shapes occurring in historical data [25]. In this approach, consecutive values in a time series are compared and movement direction is determined.

B- Sequence Mining

Sequence mining is the most general form of finding correlations between events in sequences. Sequence mining algorithms face considering all possible combinations of event sequences mostly without putting any constraint. In this respect they differ from trend analyzes since it defines shapes that narrows search space. Search space of sequence mining algorithms is also much larger than association mining algorithms. Because, in case of m attributes there are $o(m^k)$ potential frequent sequences. Therefore one of the main challenges is reducing the search space of potential frequent sequences as early as possible.

In the pioneering work, Agrawal and Srikant proposed an algorithm that finds sequences of maximal length in a sequence database [26]. A sequence is defined to be an ordered list of elements. An element is also defined as a set of items appearing together in a transaction. The proposed mechanism is based on Apriori algorithm, it finds all frequent sequences by using Apriori algorithm [27]. Then, all non-frequent items are removed. At this point, each transaction consists of as many as entries as there are frequent elements it contains, allowing repetitions. A self join is applied on frequent sequences and non-maximal sequences are removed.

In this approach is extended by postponing the construction of some intermediate results and pruning non-maximal sequences as early as possible [26]. Same authors proposed GSP approach based on previous works. GSP algorithm

is a variation of Apriori algorithm that finds association rules [27]. GSP makes several passes over database. In the first pass, all frequent 1-items are counted. From the frequent items, a set of candidate-2 sequences are formed and their supports are counted. Large 2-item patterns (L_2) are retained and become the seed set for the next pass. In general, candidate sequences of length-k (C_k) are generated by joining large patterns of size $k-1$ from L_{k-1}. A sequence s_1 joins with s_2 if the subsequence obtained by dropping the first item of s_1 is the sane as the subsequence obtained by dropping the last item of s_2. Then the candidate sequence is generated, is just s_1 with the last item of s_2 appended.

In the candidate generation step, a pruning phase eliminates any sequence at least one of whose subsequences is not frequent. For fast counting, the candidate sequences are stored in a hash-tree. For details of constructing and counting process details refer [27]. GSP also supports sliding-window that assumes within a specified time interval, all items are assumed to be originating from the same transaction even they are from different transactions.

Mannila et al. proposed an approach that finds frequent episodes within sequences that are discretized by time intervals into several smaller event sequences [28]. An episode is defined to be conjunction of events bound to given variables and satisfying unary and binary predicates declared for those variables. Simple episodes are the ones that not containing binary predicates. A parallel episode is considered to be within an event sequence e, if every event in the episode occurs disjointly within the time window of e with no constraint on the relative order of occurrence. A serial episode is considered to be within the event sequence e, if every event sequence e, if every event in it occurs in the same window of e, in the same sequential order of occurrence as the episode itself. In the proposed algorithm, first frequent sub-episodes are found and frequent episodes defined above are found. In the second phase, confidences of rules are calculated.

This approach differs from others by finding frequent patterns along with a moving time window instead of a single long sequence. In the algorithm, time window and frequency threshold are determined by the user. Since intervals affects the quality of discovered episodes, selection by the user is the deficiency of the algorithm. Zaki et al. [29] proposed a totally different approach for finding frequent sequences. One of the main deficiency of other sequence mining algorithms is scanning database many times, mostly of k times where k is the

length of the maximal frequent sequence length. $SPADE$ makes only three database scans. $SPADE$ also uses vertical database format whereas all of the previous algorithms used horizontal format. Vertical format allows very efficient joins of candidate sequences. Proposed approach also uses lattice-based approach in order be able to decompose original problem into smaller pieces and solve each piece independently from each other in main memory.

Support counting process is based on intersection of vertical id-lists, where an id-list is a list of sequences in which it occurs along with the appropriate time stamps [29]. Any sequence can be obtained by joining the id-list of the atoms. Zaki proposed efficient temporal join which states that we can generate a sequence by joining its lexicographically first two $k-1$ length subsequences, by storing only sequence-id and timestamp pairs for any sequence no matter how many items it has.

Another challenge is the enumeration of all lattice items whereas we have limited main memory. Therefore original lattice is decomposed into independent sub-lattices. This idea bases on constructing equivalence prefix classes. This provides manipulation of sequences with same prefix independent from other equivalence classes. Breadth-first and depth-first searches are employed for enumerating the frequent sequences within each parent class.

In the first phase of the algorithm, all frequent 1 sequences are computed with a single database scan. During this scan id-lists are read to main memory. Support of each item is computed. Computing the frequent two sequences constitutes the most bottleneck of the algorithm. In this step, frequent 1 vertical data is transformed to horizontal format on the fly. Frequent 2 sequences are computed. In the third step, equivalence classes are constructed based on frequent 2 sequences. Frequent sequences are generated by joining id-lists of all pairs of atoms and checking the cardinality of the resulting id-list against minimum support specified by the user. Although a pruning step is implemented in the candidate generation, it did not work.

Finding frequent sequences may not give appropriate solution for all the domains. One example to this is the error discovery where frequency of such events is not high so they cannot be discovered frequent sequence mining algorithms. Error discovery can be found ensuring existence of high confidence between error and event leading to that error. $PlanMine$ is proposed by Zaki et al. to model

detection of plan failures [30]. *PlanMine* first eliminates normative rules that are consistent with background knowledge that corresponds to normal operation of a plan. Normative knowledge is the redundant part for detecting failures and they are hand-coded by system expert. In the second phase, redundant patterns that have the same frequency as at least one of their proper subsequences are eliminated. Finally only dominating sequences that are more predictive than all of their proper subsequences. A pattern is a dominated if it contains a subsequence with lower support in good plans and higher support in bad plans than the patterns itself. *PlanMine* employees *SPADE* to identify all frequent sequences.

Incorporation of more general more general constraints into the mining process is addressed by Weiss et al. [31]. The proposed approach discovers rare events.

Chapter 3

Problem Statement and Motivation

3.1 Basic Notation

In this part we will define basic notations that we will be referring hereafter. In the following sections, we will introduce the semantic behind these concepts. We assumed data source as Web server log files and we are not given site structure information beforehand that is all site structure information is extracted from the log.

Definition 1.
Web Server Log File is the time sorted log entries where l is element of SL and SL represent Server Log.

Each log entry, $l =< l_{id}, t_{acc}, P_{ref}, P_{tar} >$, consists of an IP identifier (l_{id}), request time of the page (t_{req}), referrer page (P_{ref}) and the target page (P_{tar}).

After log files processed, sessions are extracted with also considering requests made by user but not written to log files because of caching mechanism of the system. There may exist multiple concurrent users or one user may use simultaneously multiple browsers. Each different set of accesses of a user for each browser are assumed to be different sessions.

Identification of page view is crucial for constructing the site structure and interpreting sequences with respect to this site map structure. Difficulty of the problem is because of that some pages are in of multiple frame structure. Although content is presented to user as a single view, structure in this case is composed of more than one piece. In order to handle multi-frame case we treated multi-fames differently in structure construction and sequence interpretation. Semantically we assumed that even a page is consisted of multiple frames, it is consisted of a main frame covering all information may be other than links, top and left frames having links to other pages. Core content is presented at the main frame and others are used for just jumping to other pages. Therefore it is reasonable to assume the main frame as representer of that multi-frame page. From the site structure construction perspective, since as such a page is loaded first index page is called and then all frames constituting the page are loaded, we treat index page as the parent node and others as the children of the index page. (The reason for this is that we have to keep stack and history for path completion that we will explain in the next chapter in detail.)

Definition 2.
Page view (P) is the unique content presented as a whole view of pieces constituting the requested page where
$P^i = < P^i_{id}, P^i_{ref}, t^i_{req}, t^i_{down}, t^i_{view}, S^i >$.

Each page view is composed of page identifier (P_{id}), referring page (P_{ref}), request time (t_{req}), download time (t_{down}), view time (t_{view}) and size (S) properties. Computation of download time depends on current download rate. Page view time is assumed to start after page is downloaded and become visible to user and end with a new page request. Page viewing time is assumed as criterion interest of user for that page.

Definition 3.
Each user session, S is a sequence of page views, such as, $S =< S_{id}, P^1_S, ..., P^n_S >$ where S_{id} is session identifier and P^i_S is the page view.

In a session S, $t^{i+1}_{req} - t^i_{req} < \delta t$ where δt is the maximum allowed interval in a session.

Definition 4.
ForwardPath is a path that consists of successively linked pages that shows drill-down pattern in hierarchical page organization.

Definition 5.
BackwardPath is a path that consists of successively linked pages going toward higher abstraction levels of hierarchical page organization.

Definition 6.
ForwardLink is defined as the link jumping from a node to another one on the forward path instead of following longer paths between those nodes.

Definition 7.
BackwardLink is defined as vice versa of *ForwardLink*.

*Definition*8
$Cross - links$ is defined as the link connecting nodes of related pages that are physically not very close on the site structure map.

3.2 Session Interpretation

Main goal of the work is the Web site improvement by interpreting the frequent sequences embedded inside Web logs. Although in many sequence mining algorithms frequent sequences are found, they are not efficiently utilized in interpretation of Web usage mining. Frequent sequences may consist of both frequent paths followed for reaching popular pages and mis-leaded redundant paths due to poor Web content presentation. Therefore, there appears need for interpretation of whole frequent navigation patterns so that poor design points can be detected. Such sequences would be the ones which contradicts with content presentation expected by average users. In order to explore such patterns and interpret frequent ones, we will develop a model (*WebModel*) that processes and interprets mining results into meaningful suggestions about a Web site. With this framework we aimed following points;

- enabling users to find and reach popular pages easily,
- shortening paths reaching frequently interested pages,

- linking conceptually related pages, and
- detecting mis-leading guidance.

First and second goals are for speedy access and other goals are for preventing redundant traversals. In the *WebModel*, we make use of all backward references since they generally indicate false guidance which shows a deficiency in the design. However most of the previous Web usage mining approaches did not consider backward references as a data source during interpretation of frequent sequences. If user uses "back button" after reaching a specific site, there may be mainly two reasons for that; first, he has reached the desired site and tries to reach another site related to first site, level of relatedness can be determined by considering number of back moves, second possibility is that user could not find the targeted site and still tries to find it. At this point, we use timing in order to distinguish such cases, because if a user spend much time on a Web site it generally indicates that he found the desired page. If he could not find the desired page, as soon as he realizes that it is not a related page, he jumps to other pages. Jumping occurs in form of back move, since details of currently found page does not meet user expected content.

3.2.1 Content Presentation

Web content presentation, -how page contents are organized and are related-, is a crucial issue for enhancing users reach their target page(s) easily and provide "Session Coherence" as long as they do not change the target. "Session Coherence" can be identified as visiting conceptually related pages during the user session. Conceptual categorization and relations between pages is determined by page contents and site link structure. Any mis-categorization or wrong guidance increases user trails for reaching target page and so redundant paths are navigated previous to hit. Backward referrals may be indicators of such redundant paths or any other related page. This problem can be solved by analyzing frequent backward references by using time domain as a interest criterion for visited pages. By detecting such deficiencies, Web site organization can be improved by analyzing user behaviors.

3.2.2 Data Source

Although there appears many ways of collecting usage information of users such as getting keywords, filling registration forms, adding software to client or server side for tracing more detailed information, these are not commonly applicable due to privacy reasons or installation difficulties of additional software to each Web client or server. On the other hand, Web logs are very common and can be accessed easily. Therefore, Web logs appear to be most common data source for Web usage analysis. However, data presented by log files is rather limited from several aspects and this situation makes Web usage analysis difficult. First limitation is the cache referrals. Since pages accessed within a specified period is kept at caches, in case of such pages are re-requested, they are loaded from the cache and this user request is not recorded at log file. During user navigation, all such cache referrals should be found and added to the navigation path. Especially most of the backward referrals are loaded from the cache.

3.2.3 Page Types/Content Structure

Typical navigation activity of the user depends on hierarchical organization of the pages. Higher level pages near to root of the site generally contain index pages. Index pages include links to topics that should be categorized under that page. Index pages may contain crucial or currently hot content information. Depending on the subject, index pages generally constitutes a few top level of the site structure. Below index pages, semi-content pages appear presenting both content information and but also related links with the current topic. Leaf levels are generally consisted of content pages that only include content information and rarely links to related topics. A sample Web site is given in Figure 3.1.

At Web sites having common Web content presentation, typical navigation activity occurs as follows; user first choose the general topic of interest and then continues reaching more detailed sub-topics. When user reached the target sub-topics, he reads the content information under that sub-topic and it may be consisted of several pages. During navigation of interested pages, user spent some time for dealing with the content and he may also jump to related sites. If he could not find the desired content under the current sub-topic, he navigates

back enough so that he can make any other trail for the target content. In this case user does not spent much time at non-interested pages.

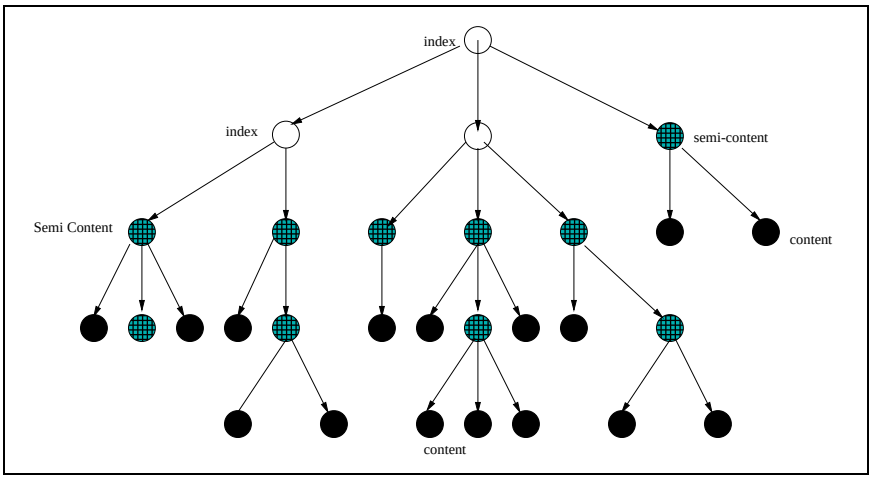

Figure 3.1: Page Types are: Content Page: Black Nodes, Semi-Content: Nodes with Cross-Lines and Index Pages: Empty Nodes.

In Figure 3.1, index pages are represented with white nodes, semi-content pages with cross lines and content pages with black color.

3.3 Site Improvement Recommendations

In this part we will explain improvement suggestions proposed by $WebModel$.

3.3.1 Putting Shortcuts for Popular Pages

First type of recommendation is for making easier to find and reach popular pages. If forward paths are frequent ones, it is very highly possible that such paths are followed in order to reach popular pages. This case is illustrated in Figure 3.2. A link is added to page P to reach directly page Q. By this way guidance information for reaching frequently accessed page Q is added to higher levels. When user notifies existence of shortcut to page Q, all the path from page P to Q is passed by a single click. By this way, effort for finding and reaching to page Q is minimized. For example, when a new homework is assigned that page is requested frequently and all the path to reach homework page is followed

each time by users. In the first place, users face with the problem of finding the related homework with minimum effort. In order to decrease this effort, a link to a frequently accessed page can be put in higher levels (Figure 3.2). When frequent access to homework site decreases, added link should be removed. Therefore, remove recommendation should also be included in $WebModel$. We assumed that user interest in a site if he spends relatively more time compared with other pages in that session.

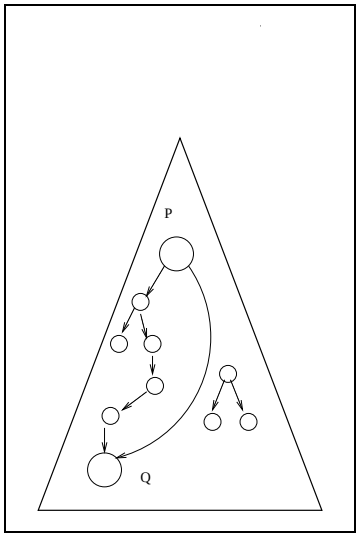

Figure 3.2: Forward Link Adding: Shortcut Add from Page P to page Q.

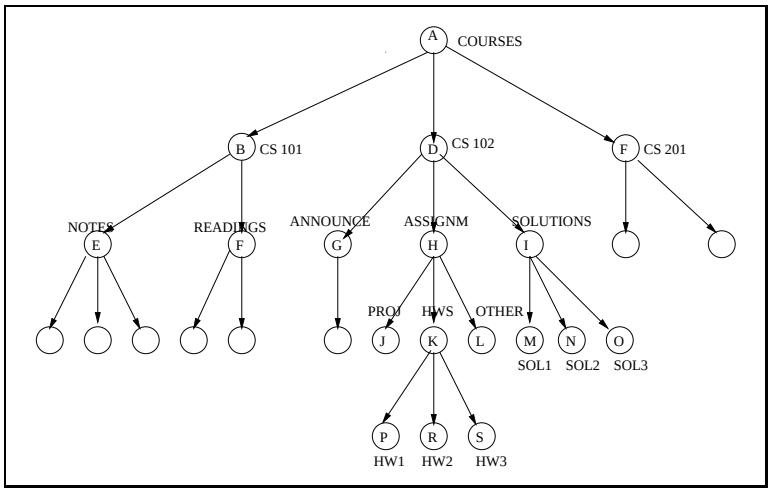

Figure 3.3: Forward Path Shortening Example: Shortcut Add from Page D to Page S, Linking Related Pages: Cross-link from Page S to Page O.

However interpretation of frequent backward and cross links is different from frequent forward links. In Figure 3.3, path A, D, H, K, S is frequently accessed

and page S is of interest. Therefore a more indicative link to this page should be inserted at higher levels. This may be: A to S, D to S or H to S.

If we assume that A corresponds to `COURSES` and D to `CS102` and S to `HW3`, D to S appears as the best choice. In addition, user may interest in more than one site. For example, after user accesses `hw3` page, he may want to see solutions of `hw3` also. Therefore, he will probably traverse the following paths; `PATH1`: A, D, H, K, S and `PATH2`: I, O. When we consider backward reference after reaching `hw3` site, and fill the missing part, we obtain `PATH2` as S, K, H, D, P, I, O (O is solutions 3). Therefore,a guiding link should be added to the page D to be able to directly reach page O.

Until this point, knowledge of frequent accesses to `hw3` and `sol3` pages are converted to guidance as putting extra links to higher level. However we still miss the temporal knowledge of paths `hw3` and `sol3`. If we assume that, users generally access first `hw3` and than `sol3`, (obviously he uses backward references) we should also put a guiding link from `hw3` to `sol3`.

3.3.2 Linking Related Pages

Second kind of recommendation targets linking conceptually related pages according to user. If several users follows a path starting with page R and interested in firstly page P and successively interested page Q, Figure 3.4, this indicates pages P and Q are conceptually related and cross-link can be added to page P, Figure 3.5. If user is not interested in at least one of the pages, interpretation differs. Interest for a page can be understood by comparing page viewing time for that page with other pages. These issues will be addressed in the next chapter in detail. User backtracks in order to reach the related page. Conceptually related page patterns can be explored by successive interested pages with backtracking between those pages. Similar idea is also discussed in [33]. In that work backtracking is also considered as navigation between related pages. In Figure 3.3, if there appears frequent jump from `hw3` to `sol3`, these pages are linked since they are conceptually related.

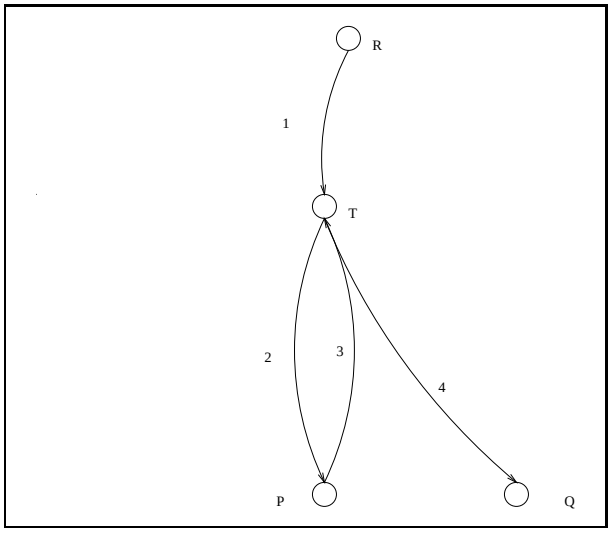

Figure 3.4: Back Traversal Indicates Conceptually Relatedness of Page P and Page Q.

3.3.3 Detecting Misleading Guidance

Third kind of recommendation is revealing mis-guidance information. If a cross reference is made, and this link is followed by many users to reach the referenced site, it indicates that referenced page has incorrect guidance, because users generally go another site before reaching the desired page. Therefore, some guidance should be added at a place above of both referencing and referenced pages.

In previous cases we assumed that user found target pages. However, what happens when user can not find the desired page in the first trial ? In this case, he backtracks to the point where he thinks that he is misled by the wrong or poor guidance and tries other possible paths until he finds the target or ends the navigation activity for that target. Since user could not find the target in first trial, we assume that he does not spent much time on non-interested page. We can figure out whether the user found target page by computing page viewing timings after backtracking. Mis-guidance point can be assumed the page where the user repeat backtracking for the same target. At that point, more detailed information should be put for frequently accessed target page.

As an example to this case is given in Figure 3.6. When a new homework is assigned, user may search this in `announcement/new/` part. If he could not find it, he probably backtracks to main course home page (of CS 102) and chooses in

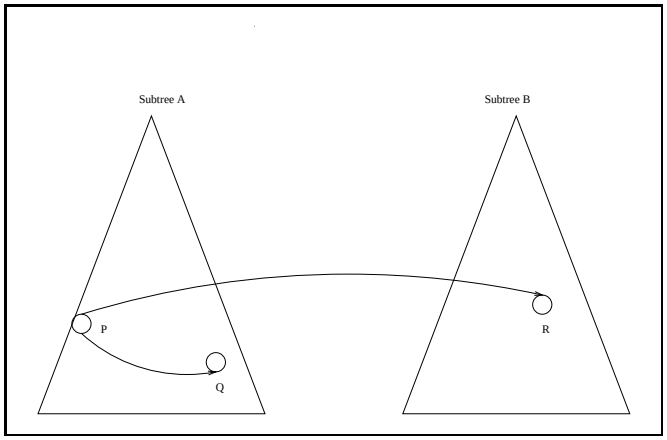

Figure 3.5: Linking Related Pages.

this case `homework` section. The first path is: A, C, G, K and the following path is H, N, V. This case is illustrated in Figure 3.6. Since the first trial is the result of false guidance, either a more indicative note should be appended to ancestor pages, in this case under CS 102, or an extra link should be added to the page where backtracks starts(in this case at page K, link K to V).

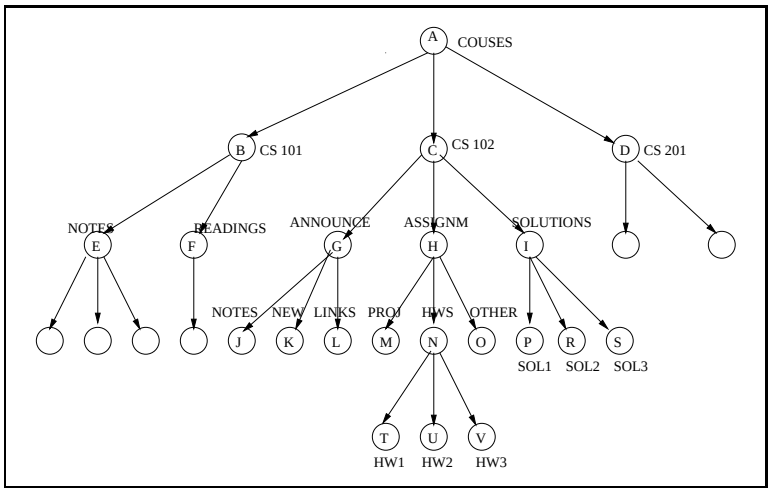

Figure 3.6: Detecting Misleading Guidance: Upon not Finding Target at Page K, Page V Is Tried.

3.4 Interest Criterion

In the proposed system *WebModel*, site improvement recommendations are based on interest criterion on a page because interpretation differs with respect to interest in a page. Even though we extract all navigation path of user correctly, we are not able to correctly interpret these sequences without incorporating time domain. We can understand whether user reached or still searching target page by deciding by measuring the relatively spent time on that page. Although page request timings are recorded on log file, we can not use these timings directly for estimating page view timings but we have to process request timings. There are several issues that need to be solved;

- estimating page download time for single page,
- estimating page download time for multiple frames,
- considering browser type for figure download policy,
- computing page view timings of cache referrals, and
- detecting changing network transmission rates and user navigation speed.

We can define page download time as the time beginning with page request time and ending with the time as requested page become visible to user. Timing steps for a single page is illustrated in Figure 3.7.

In Figure 3.7 successive request timings of two pages are shown. Both pages assumed to contain no figure. Although client requested page at time t_0, page request information is written at time t_1 to server log. Even though first page download duration is $t_4 - t_0$, only new page request time t_6, request time of the second page, is written as the next record to server log. First page becomes visible to user at time t_4 and actual page viewing time is $t_5 - t_4$. At t_5 second page is requested and and this point is assumed as the end of page viewing of the first page.

However, in the sample case server log records only t_1 and t_6. Other timings should be computed by using other information in the log. For the simplest case, if page contains no figures, download time can be calculated by knowing size

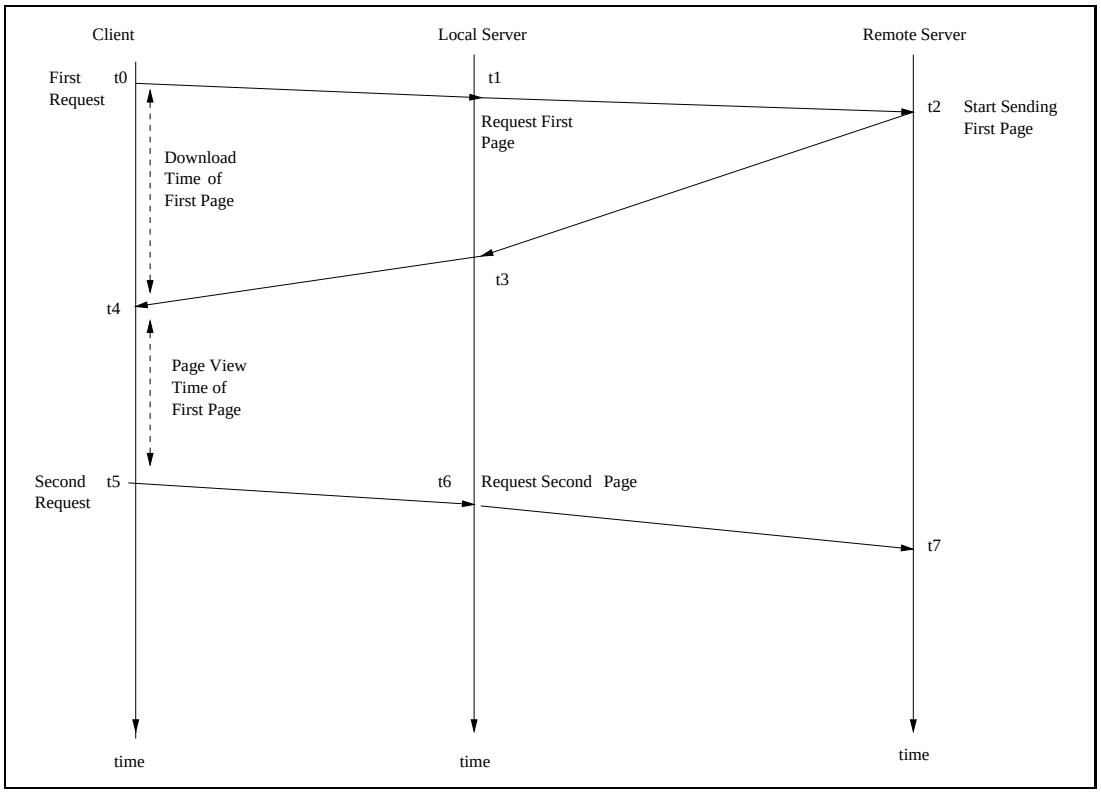

Figure 3.7: Timing Steps For a Single Page Download.

and current download rate. Although size is known, download rate is subject to change.

If page contains some figures, then we have to calculate download time of each figure of this page. Consider page `/courses` has figures `fig1Cours.jpg` and `fig2Cours.jpg`. This case is given in Figure 3.8.

The text body of the page is first downloaded and becomes visible at time t_4 and successively figures at t_7 and t_{10}. In the sample case visibility of the page context is assumed to be gradual that is as each part is downloaded, it immediately becomes visible to user. Although this strategy is the common one, some browser types waits all parts of the content is downloaded and makes all parts visible after this step.

Most complex part for page view time computation is the case where page is composed of multiple frames. Each frame may also contain figures. An example case is given in Figure 3.9. In this case when a page is multi-frame page is requested, first index page is loaded and successively frame components with

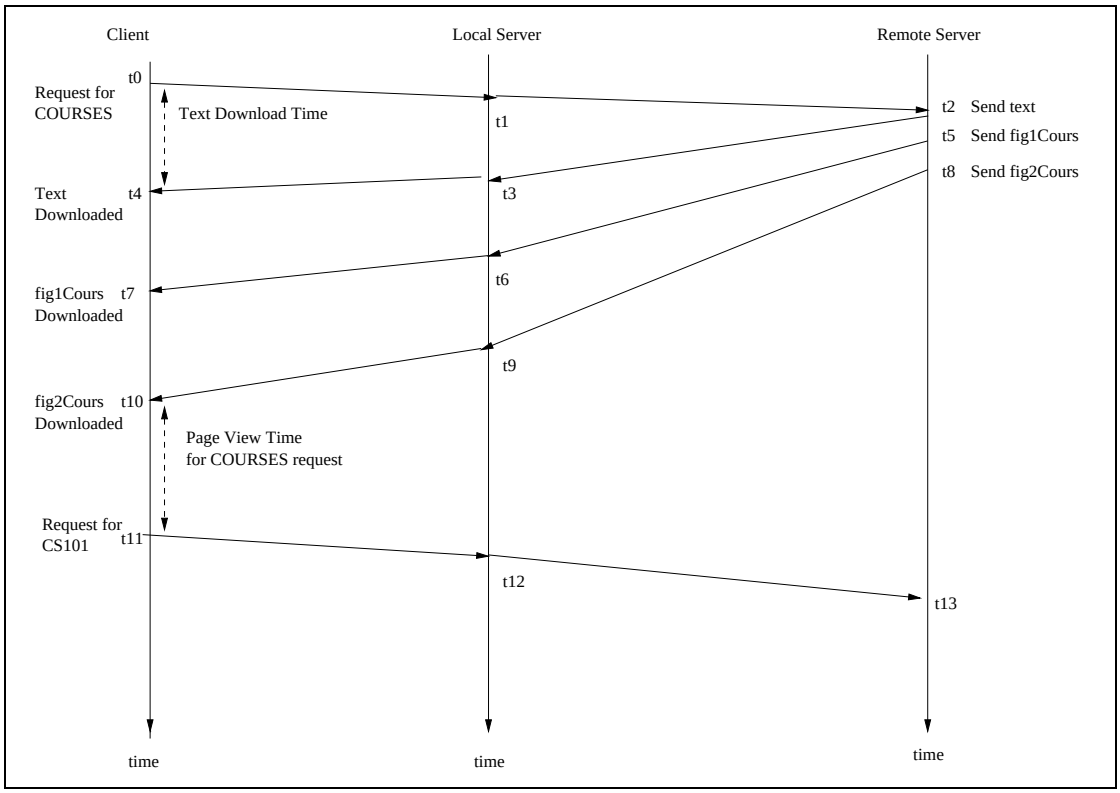

Figure 3.8: Timing Steps For a Single Page With Figures.

text bodies are loaded. Figures on pages are loaded after all these operations are completed. Therefore, we have to detect multi-frame pages and catch figures belonging these frames.

For cache referrals case, since such pages are not recorded in logs, only information that can be used is the previous page timings and new page request as the end of page view of cache referrals. Pages are loaded very fast from cache and we can estimate page viewing time as the large fraction of the difference between previous request and later request. More detailed algorithms for computing cache referral viewing time will be given in the next chapter.

Another parameter required for computation of download timing is download ratio that is the network transmission speed represented as the number of bytes transmitted per second. Since this value is subject to fast change, its value should be continuously updated. Download ratio depends on bandwidth rate of the system, network congestion and internal/external connection. Whether connection

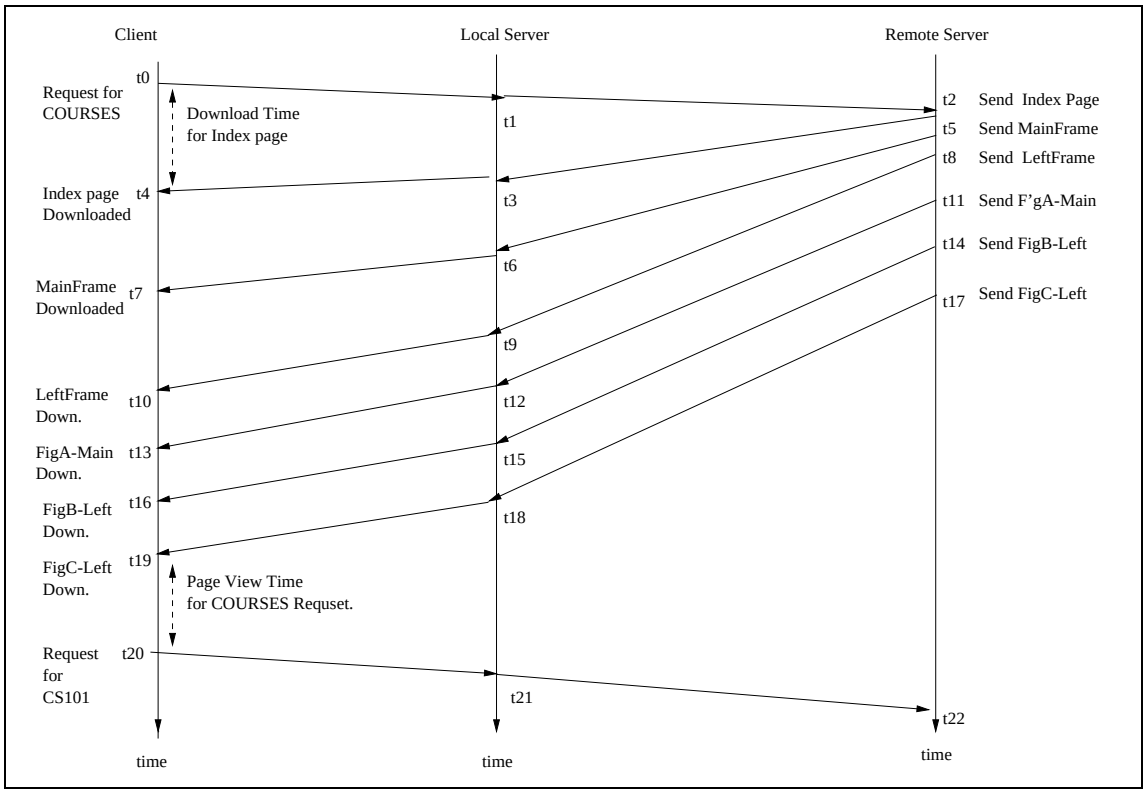

Figure 3.9: Timing Steps For a Page With Multiple Frames and With Figures.

is internal or not can be understood by looking domain/IP of the session. However, other parameters can be estimated by analyzing sample download timings from the real log data.

Still network congestion problem could not be addressed so far. Download of the same sized pages may differ very much at different times due to changing transmission rate. In order to handle this problem, we used relative time approach. Relative spent time on a page gives weight of that page compared to other pages. As stated previously, we assumed that user stays for a minimal time if he does not interested in current page.

3.5 Comparison of Traditional Data with Web Data

In most of the Data Mining applications so far traditional input data is considered to be customer data and this traditional data format is used as a standard format [27]. Although both customer and Web data involves temporal knowledge, Web navigation activity differs from customer purchase from several aspects. First distinguishing point is that since Web pages are accessed one at a time, Web sequences are strictly time ordered. For example there appears no such sequences; $A \rightarrow BC \rightarrow E \rightarrow G$ since there is no possibility of accessing pages B and C simultaneously. For customer data, it is possible that customer purchase several items at the same time, so there are both itemsets and sequences. Although it can be suggested that some pages are composed of multiple frames and if each frame is considered as one item, some pages can be considered as consisting of several items, such pages are mainly composed of single main page and other pages are designed for links to reach other sites.

Second difference is that although during a customer purchase activity if one item appears more than once, it is generally counted once and adds support of value 1 where each occurrence of page is important and should be added to support of that page. Because multiple occurrence of an item during a session indicates interest for that page.

Third difference is that, since each step in navigation activity probes according to Web site link structure, at each point of the session possible moves of user is determined although in customer sequence no such limitation exists. If we consider Figure 3.10, if user has reached page E by following the path $A \rightarrow B \rightarrow E$ there are three possible moves at page E: backtracks to page B, or follow one of the forward paths G or H. In customer sequence there exist as much as number of items in the store. We will utilize this property in sequence intersection optimization part.

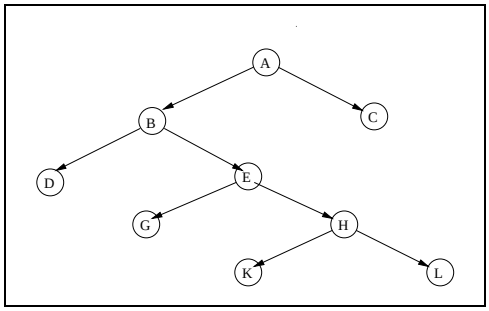

Figure 3.10: Web Structure Determines Path Possibilities of the User.

Chapter 4

Usage Processing

The goal of usage processing is to convert raw log data into useful knowledge format so that it can be used for finding frequent sequences and interpretation of them. As previously discussed in Chapter 3, in order to be able to interpret navigation behaviors of users each session information should be distinguished successfully from other sessions, missing page access information should be extracted and time domain should be incorporated into sequence information. In this framework for realizing these goals, we extracted session information and site link map structure in order to both embed mining information and also to help interpretation of frequent sequences. Page view time computation heuristics are used for deciding whether a page is interested or not. In session information completion process all cache referrals are handled. Since backward movements are considered as well as forward movements, or the combination of two are considered. Processing module mainly is composed of the following steps:

1. removing useless log records and parsing,
2. extracting user sessions,
3. handling cache referrals,
4. site structure extraction, and
5. page view time computation.

4.1 Removing Useless Log Records and Parsing

Data cleaning and parsing step gets server log file of a specific site and filter outs unnecessary data, parse requests, normalize Web request strings. In our case, we will deal with NCSA combined log format since server logs of CS domain of Bilkent University is kept in this format.

CGI, binary file, executables and similar requests other than graphic files are filtered out since they present no valuable knowledge for usage behavior. These files are removed by checking extension of each request URL. Although other many approaches also stripped out graphic files we make use of them for computing page view timings. Because load time of a page is highly depends on graphic files it includes. On the other hand, since number of graph file requests are very large this increases pre-processing time.

Referrer field is parsed in addition to request field since it is used as a footprint in finding the owner session of that specific page request, that is for finding the predecessor request. With respect to time field of the page request, time field gives the page access time of the request made. Successor time field gives the end of view time of the current page. Difference between successor time and current time gives the total of download time and view time of the page and this does not give the exact page viewing time Especially for graphic dense pages page load time is very high compared to text dense pages. This field is very important in interpretation since sequences having same temporal order may reflect different patterns. In cache referrals and multi-frame pages situation differs from the default case.

Another step of the data cleaning task is to normalize, convert into standard format, the request and referrer URI fields. Requests made for a directory is assumed to be a request of a file ending with `index.html` or `home.html`. The directory request may or not has an ending slash and "www" string is optional at the beginning of the URL. if these parts are missing in the request, should be added. A common convention should be applied to all page requests.

Domain	Time	Target	Code	Size	Referrer	Agent
ihlara	[01/Nov/2001:23:35:44]	GET /	200	775	"-"	Mozilla/4.07C-SGI
ihlara	[01/Nov/2001:23:35:48]	GET / aerdem/courses.html	200	1601	/ aerdem	Mozilla/4.07C-SGI
ihlara	[01/Nov/2001:23:35:50]	GET / aerdem/projects.html	200	1402	/ aerdem	Mozilla/4.07C-SGI
ihlara	[01/Nov/2001:23:35:51]	GET / tugrul/grades.html	200	2456	/ tugrul	Mozilla/4.07C-SGI
ihlara	[01/Nov/2001:23:35:53]	GET / tugrul/projetcs.html	200	732	/ tugrul	Mozilla/4.07C-SGI

Table 4.1: Sample Log File.

4.2 Extracting User Sessions

Record of user foot-print data such as cookies and embedded *Ids* enhances user identification and client-tracking methods helps this identification. These two techniques are the most applicable ones compared to registration and tracking mechanisms provided via browsers. Cookie and embedded Id methods may not always be available due to privacy issues. Therefore, we use IP address and Agent pairs, that are always available in server logs, for user identification. See Figure 4.1 for part of a sample log file.

Main cases to be considered in user identification is listed in [8] as in the following and and solution we used are given afterward:

1. Single IP address/Multiple Server Sessions: Users connecting to the Web from the same proxy servers along with the same time period may use the same IP address.

2. Multiple IP address/Single Server Session: Mostly due to privacy concerns Internet service providers give multiple IP addresses to a single session.

3. Multiple IP address/Single User: If a user accesses from different machines, for each different machine a specific user gets a different IP address and so these are evaluated as different users.

4. Multiple User Sessions/Single User: If a user opens and uses multiple browsers from the same machine, concurrent accesses occur with the same IP address. Since log accesses are written in time order, paths followed by the same user could be misinterpreted.

As stated before, we assumed that the only available source is Web logs. In that case, there is no way to keep track of the same user throughout connections from different machines in case 3. Because user identity should be kept across

multiple machines via registration and this identity information should be made public among those machines.

For proxy servers using privacy policy of the user, different IP addresses are given during a continuous visit. As each time a new IP address is given it would be evaluated as a different user.

Other than these cases, problems presented in case 1 and case 4 could be solved with using only Web log data. For the base case, we assume that each IP/Agent type pair represents a different user. However, in case 1, multiple users use the same IP address through a proxy server and concurrent user browsing activities should be distinguished. For case 4, when a single user uses multiple browsers at the same time from the same machine, since Web log file reflects time ordered page requests of the user, it can be understood as a single session activity which gives wrong path information. In order be able to interpret paths appropriately user sessions should be distinguished clearly.

After determination of users, sessions should be identified. Assumption of IP/Agent pair represents different users also means they represent different sessions. Since we assumed that we are using only Web log of a specific site, it is not possible to know when a browsing activity ends. However thirty-minute time is accepted as a common standard for distinguishing two successor sessions [36]. That means, if the same user does not request any page during thirty-minute time, it is assumed that session is ended.

In log files for each page request; referrer page, the page from which that request is made, and target page, the page which user wants to go, are recorded. If we assume that all requests are recorded during navigation and none of the page is brought from the cache, each referrer page of the current request should be equal to target of the previous page. With this information we can explore user session by matching the referrer page with target of previous access. A simple example given in Table 4.2. In this case requests and path obtained are given. This is defined as "Referrer Page Heuristic" and used as a standard for finding single sessions. For identification of different sessions in case 1 and case 4, $ReferrerPageHeuristic$ can be used. If the referring page is not part of a session then that means request belongs to a different session. Referrer page is searched and including session is assigned as the owner session. However there may be more than one active sessions that the referrer page may belong to. That

is, different users or the same user with multiple browsers may be accessing the same or close set of pages concurrently. Therefore referrer page may exist in multiple open sessions. In this case we have to use find the session owner.

Referrer	Target
-	agursoy
agursoy	agursoy/projects.html
agursoy/projects.html	agursoy/projects/parallel.html
agursoy/projects/parallel.html	agursoy/projects/parallel/loadBalanc.html

Table 4.2: Path Construction by Using Referrer Fields.

Resulting constructed path is given as: $PATH$: {agursoy, projects, parallel, loadBalanc}.

For this purpose we used "Multiple Session Heuristic" defined by [35]. This heuristic algorithm assigns the request to the session that is closest to referring page. "Closeness" is defined as the minimal path that is required to access the referring page from the current page. If there exists multiple sessions containing referrer page, then the closest session is chosen as the owner. In case of existence of multiple sessions having the same closeness to the referring page, the session most recently accessed session in terms of time is chosen. In case of proxy-level cache use for static Web pages, multiple requests for the same page is reflected to server log as a single request although in our heuristic it is assumed that every page request is written to the Web log. However there is no practical solution without extra information.

Session identification algorithm is given in Figure 4.1. In line 4, log data is sorted with respect to time and agent fields. This accomplishes assumption of identifying each IP/agent pair as a different session. Line 5.1.1 checks conditions for opening a new session. These conditions are either session timed out or referrer page could not found in any of the active sessions. In both cases a new session is opened in part 5.1.1. If the referrer page is included at least in one of the active sessions, then closest session to the referrer page is chosen in part 5.1.2.2. In line 5.1.2.1, for each session including the referrer page *Distance* function is called which is given in Figure 4.2. *Distance* function computes the distance from last item of the *History* to lastly accessed referrer page and page access recency respectively in lines 4.1.1 and 4.1.2.

1. Let $H_i = f_1, f_2, ...f_n$ denote the ordered session history.
2. Let $l_j, f_j, r_j and t_j$ denote a log entry, request, referrer and time, respectively.
3. Let T denote the session timeout.
4. Sort data by IP address, agent and time.
5. for each unique IP/Agent pair do
   - 5.1 for each l_j do
     - 5.1.1. if ((t_j - t_{j-1}) $\geq T$ or $r_j \in H_0$,...,H_m)
       - 5.1.1.1. Increment i
       - 5.1.1.2. Add l_j to H_i
     - 5.1.1.2 else
       - 5.1.2.1. assign=Distance(H, r_j)
       - 5.1.2.2. Add l_j to H_{assign}
       - 5.1.2.3. $sessionID = assign$
       - 5.1.2.4. $lastElt = H_sessionID.last$
       - 5.1.2.5. if $r_j \notin H_0, H_1, ...H_n$
         open a new session
         call $ProcessIPSession(topSessIdx)$
       - 5.1.2.6. else
         call $ProcessLogEntry(sessionID, r_j, f_j)$
6. end

Figure 4.1: Session Identification Heuristic.

Function $Distance()$

1 Let H_i denote a time ordered session history.

2 Let f_i denote a page file

3 Set min=∞

4 for each $H_i \in H$

4.1 if $f \in H_i$

4.1.1. $d_i = H_i.size() - H_i.index(f)$

4.1.2. $t_i = H_i.t_n - H_i.t_f$

4.1.3. if $(d_i \leq min)$

4.1.3.1. assign = i

4.1.3.2. min = d_i

4.1.4. else if $(d_i = min)$

4.1.4.1 if $(t_i < t_{assign})$

4.1.4.1.1. assign=i

5 return assign

6 end;

Figure 4.2: Distance Function.

Function $ProcessIPSession(topSessIdx)$

1 for each Session $i < topSessIdx$

1.1 call $ConstructSiteMap(Session_i)$

1.2 call $ComputePageViewTime(Session_i)$

Figure 4.3: Function for Processing Multiple Session Information.

Then, in 4.1.3.1 and 4.1.3.2 it finds the closest session. If there appears multiple closest sessions, the most recent session is chosen in 4.1.4.1. After the owner $sessionID$ is found, the current log entry (referrer-target pair) is processed in that session. For that purpose $ProcessLogEntry(sid, r_i, f_i)$ function is called. Algorithm for this function is given in Figure 4.4. If a new IP is detected or difference between previous access time and current access time is greater than the 30 minute limit condition, then all sessions information kept so far is processed. Number of concurrent session that belongs to current session is given by $topSessIdx$. All sessions processed at this point belong to the same domain. Algorithm for $ProcessIPSession(topSessIdx)$ is given in Figure 4.3.

In $ProcessLogEntry$ function, if request is the first access in the current session (part 6), target page and referrer page -if exists- are added to the current history and stack. If request is not the first access (part 7), referrer page existence is checked. If referrer page is $NULL$ this indicates that page request address is directly written without using any referrer page. For the other case, previous target equals to current referrer, path continues normally and current request is added to stack and history. If referrer page could not be found, then $PathComplete$ function is called, since in this case referrer page is accessed from browser back button or from history directly. For each session a local $SessionTree$ is constructed in order to use in handling cache referrals. This tree is constructed by adding each request as a child of the referrer. For finding easier cache referrals these nodes are enumerated. After $ProcessIPSession$ function is called, each $SessionTree$ constructed for each session is erased. Therefore there is not much extra cost of these local trees. Request URL, parent URL and last access time is kept in all nodes.

Function $ProcessLogEntry(sid, r_i, f_i)$

1. Let sid denote a session of page views.
2. Let r_i denote the referring page and f_i target page
3. Let S_{sid} denote stack of session sid.
4. Let H_{sid} denote history of session sid.
5. set $notInStack = false$; set i to 0 ;
6. if first access in session
   - 6.1. if $r_i = NULL$
     - 6.1.1. add f_i to H_{sid}, S_{sid}
     - 6.1.2. add f_i to $SessionTree_{sid}$
   - 6.2. else
     - 6.2.1. add r_i to H_{sid}, S_{sid}
     - 6.2.2. add f_i to H_{sid}, S_{sid}
     - 6.2.3. add r_i to $SessionTree_{sid}$
     - 6.2.4. add f_i to $SessionTree_{sid}$
7. else if (not first access) and ($r_i = NULL$)
   - 7.1. add f_i to H_{sid}, S_{sid}
   - 7.2. add f_i to $SessionTree_{sid}$
8. else if (not first access) and ($r_i = f_{i-1}$)
   - 8.1. add f_i to H_{sid}, S_{sid}
   - 8.2. add f_i to $SessionTree_{sid}$
9. else if (not first access) and ($r_i \neq f_{i-1}$)
   - 9.1. call $PathComplete(sid, r_i, f_i, S_{sid}, H_{sid})$
10. call $ComputeTimings(\ r_i,\ f_i,\ t_{req}\ )$

Figure 4.4: Process Log Entry Function.

4.3 Handling Cache Referrals

In order to improve page load performance, browsers in general keeps currently accessed pages in their caches. When users try to re-access the same pages by *BACK* and *FORWARD* buttons or via history of the browser, these pages are not requested from the server, instead they are loaded from the browser cache. Because of that pages requested from cache are not written to log files. However, combination of backward and forward paths give crucial information about deficient points or related pages of a site. Therefore we need full navigation requests for each session should be completed.

For completing missing parts of the session requests, different approaches are used. Zaki [29] and Chen et al. [3] used prefix completion approach. Since they concentrated on use of forward references only, they perceived each backward reference as indication of last forward path and backward references are used to just make navigation easier. In addition to this, they did not solve problem of finding page requests from the cache. An example to this approach is given in Figure 4.5. However in this approach we loose backward information with turning points, the points appears as bridge between backward and forward references. For this user, maximal forward references are defined as *ABCD*, *ABEGH* and *ABEGW*. No backward path information is kept.

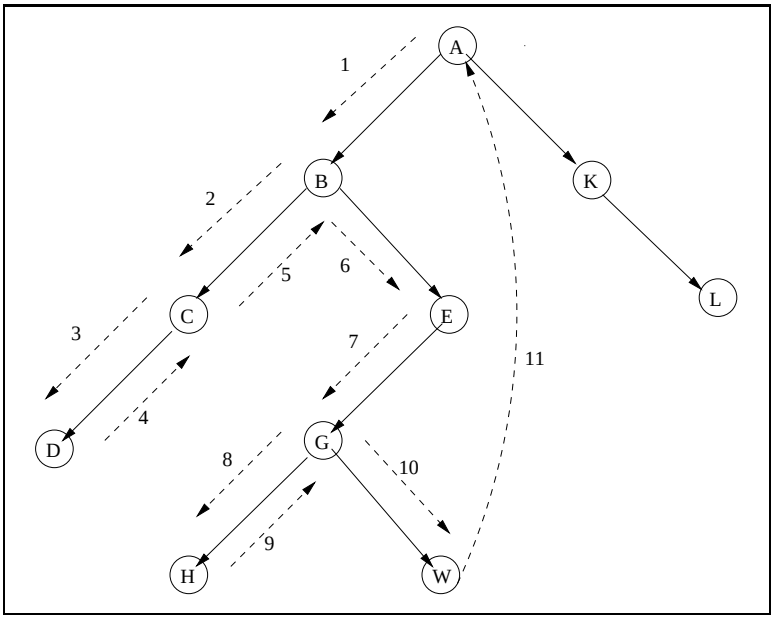

Figure 4.5: Prefix Completion.

For our case, all backward references should be obtained clearly and also forwad references brought from the cache. Nakayama et al. [17], Ramarkish et al. [33] and Cooley [35] found continuous path including forward and backward ones. In all these approaches stack for keeping last page request and session history are kept. Cooley handled problem of finding cache referrals problem.

We improved Cooley's approach for handling multi-passes from same paths and optimized algorithm for fast path finding. For each session we kept local $SessionTree$ that represents tree of a user session and all nodes are enumerated on in this tree.

Our first improvement is $IDIntersection$ for fast finding cached pages, we will give an example. Second improvement is faster handling of multi-passes from the same path. Passing over a path is enough so that one can re-passes over this path by using $BACK$ or $FORWARD$ buttons. If we consider that there are several distinct paths reaching the same node, say P, from a specified node, say Q, and assume that all these paths are cached, traversed at some time. When we realized that user passed from P to Q again, it is somewhat time consuming task to find which path is used lastly. Instead of this, we assigned each node, last page request time that is set as last user access time user and updated upon each user access time. By this way it will be faster to find lastly used path among other alternatives instead of searching browsing history.

In Figure 4.6 this process is explained. In part 6 referrer page is searched in the current session stack and if it is found, all pages on the back path is removed from the stack and added to the history. This case shows that user used $BACK$ button to reach the current target. If referrer could not be found in stack this case shows that user used $BACK$ button and followed some links in one of the pages in $BACK$ path. This will be exemplified later by figures.

An example path is given in $(referrer, target)$ pair format. Sample entries are given in Table 4.3. Resulting constructed graph is given in Figure 4.7.

In `sample1` session, till `cs102` page is requested all previously requested pages are added to Completed Session ($CSession$). Since referrer of first request is empty, only target page is added to $CSession$. Since referrer page of the second request equals to target page of the first entry, path continues in forward direction and target page is added to $CSession$. Other log entries improve the path in

Function $PathComplete(sid, r_v, f_v, S_sid, H_{sid})$

1. Let sid denote a session of page views.
2. Let r_v denote the referring page
3. Let S_{sid} denote stack of session i.
4. Let H_{sid} denote history of session i.
5. set $notInStack = false$; set i to 0 ;
6. while (($notInStack$) and ($i < lastStackElt$))
   - 6.1. if ($S_{sid}.i \neq r_v$)
     - 6.1.1. pop $S_{sid}.i$
   - 6.2. else
     - 6.2.1. $notInStack$ = false
     - 6.2.2. $referIndex = i$
   - 6.3. add $S_{sid}.i$ to $tempList$
7. if ($notInStack$)
   - 7.1. add $tempList$ to H_{sid}
9. else
   - 9.1. $notInHist$ = true; set $j = lastHistElt$
   - 9.2. while (($notInHist$) and ($j > 0$))
     - 9.2.1 if ($H_{sid}.j = r_v$)
       - 9.2.2.1. $notInHist$ = false
       - 9.2.2.2. $referIndex = i$
     - 9.2.3. $i--$
   - 9.3. if ($notInHist$==0)
     - 9.3.1. call $comAncest = findCommonAncestor(r_v, H_i, H)$
     - 9.3.2. call $findBackwardPath(comAncest, H)$
     - 9.3.3. add $BackwardPath$ to $tempPath$
     - 9.3.4. call $findForwardPath(comAncest, H)$
     - 9.3.5. add $ForwardPath$ to $tempPath$
     - 9.3.6. $pathLength = tempPath.length$
     - 9.3.7. $timeInterval = (r_v.time - H_{sid}.j)/pathLength$
     - 9.3.8. for each page in tempPath
       - 9.3.8.1. $tempPath_i.time = r_v.time - (i * timeInterval)$
     - 9.3.9. add $tempPath$ to H_{sid} and $SessionTree_{sid}$

Figure 4.6: Path Completion Heuristic.

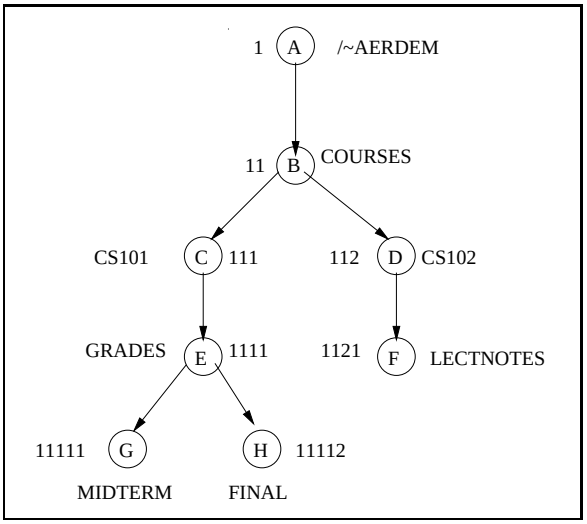

Figure 4.7: First Session Graph.

Referrer	Target
-	aerdem
aerdem	aerdem/courses
aerdem/courses	aerdem/courses/cs101
aerdem/courses/cs101	aerdem/courses/cs101/grades
aerdem/courses/cs101/grades	aerdem/courses/cs101/grades/midterm
aerdem/courses	aerdem/courses/cs102
aerdem/courses/cs102	aerdem/courses/cs102/lectNotes
aerdem/course/cs101/grades	aerdem/course/cs101/grades/final

Table 4.3: Requests with Missing Cache Referrals, `sample1`.

the same way. However referrer page of the sixth entry is not equal to target page of previous entry, referrer is searched in the stack first. `Courses` is found in the *Stack* and all items up to `courses` is popped from the *Stack*. This popped items from the lastly accessed page to the referrer page found in history construct backward path. This backward path is added to the history.

Forward path improvement continues up to `final` page is requested. Referrer page, `grades`, is searched in the *Stack* but it is not found. Because user followed a backward path first by using $BACK$ button, and than followed some links in forward direction. However, this forward path again may be accessed from the cache and not written to the log file. Therefore, for this case referrer may be in subtree, $SubTree$, of which root is the node the backward path ended. This node should be found first and than backward and forward paths should be found. At

this point, path completion heuristic is used to extract these paged used from the cache. Although backward path is found by keeping a temp path from the lastly accessed page to the referrer page in History, there appears many possibility for forward path especially if $SubTree$ is large since it should be found in the subtree $SubTree$.

In order to find the forward path fast we proposed ID Intersection Method. That is, all nodes of the session is enumerated reflecting the parent-child relation and child order. For finding referrer node in the history, ID of lastly accessed (ID_{LA}) page and ID of target page (ID_T) is intersected and common prefix $ID_{ancestor}$ gives the ID of the referrer in the tree. Backward path is found by extracting all IDs between ID_{LA} and ID_T in backward direction. Forward path is found by extracting IDs from $ID_{ancestor}$ to ID_T. By using ID intersection method forward path is found very fast. Eventual session tree is given in Figure 4.7. Completed path is given in Table 4.4.Corresponding path: {`aerdem, courses, cs101, grades, midterm, grades, cs101, courses, cs102, lectNotes, cs102, courses, cs101, grades, final`} .

Referrer	Target
-	aerdem
aerdem	aerdem/courses
aerdem/courses	aerdem/courses/cs101
aerdem/courses/cs101	aerdem/courses/cs101/grades
aerdem/courses/cs101/grades	aerdem/courses/cs101/grades/midterm *
aerdem/courses/cs101/grades/midterm	aerdem/courses/cs101/grades *
aerdem/courses/cs101/grades	aerdem/courses/cs101 *
aerdem/courses/cs101	aerdem/courses *
aerdem/courses	aerdem/courses/cs102
aerdem/courses/cs102	aerdem/courses/cs102/lectNotes
aerdem/courses/cs102/lectNotes	aerdem/courses/cs102 *
aerdem/courses/cs102/	aerdem/courses *
aerdem/courses	aerdem/courses/cs101 *
aerdem/course/cs101	aerdem/course/cs101/grades *
aerdem/course/cs101/grades	aerdem/course/cs101/grades/final

Table 4.4: Completed Path Requests with Cache Referrals.

In `Sample2`, multiple session handling process is exemplified (Table 4.5). Since log entries are time ordered, in case of existence of multiple users accessing from the same IP, as each log entry is processed, owner session of that entry is found

Referrer	Target
-	aerdem
aerdem	aerdem/courses
tugrul	tugrul/grades
aerdem/courses	aerdem/courses/cs101
aerdem/courses/cs101	aerdem/courses/cs101/grades
gudukbay	gudukbay/research
aerdem/courses/cs101/grades	aerdem/courses/cs101/grades/midterm
gudukbay/research	gudukbay/research/tubitak
tugrul/grades	tugrul/grades/final
aerdem/courses	aerdem/courses/cs102
aerdem/courses/cs102	aerdem/courses/cs102/lectNotes
gudukbay/research	gudukbay/research/master
tugrul	tugrul/books
aerdem/course/cs101	aerdem/course/cs101/grades

Table 4.5: Multiple Session Requests, `Sample2`.

by searching referrer page in all active histories. Search starts from the last active session through old active sessions. Once the owner session is found, each entry is processed in its session and path completion algorithm is called as it becomes necessary. Three sessions are constructed for sample2 and corresponding trees are given in Figure 4.7, 4.8 and 4.9. "*" indicates that page is brought from the cache. First completed Session is given in Table 4.6.

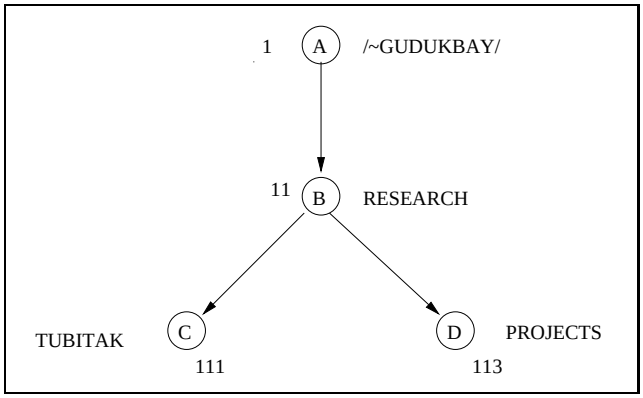

Figure 4.8: Second Session Graph.

First path:{`aerdem, courses, cs101, grades, cs101, courses, cs102, lectNotes, cs102, courses, cs101, grades, final` } . Second completed path is given in Table 4.7. Second path: {`gudukbay, research, tubitak,`

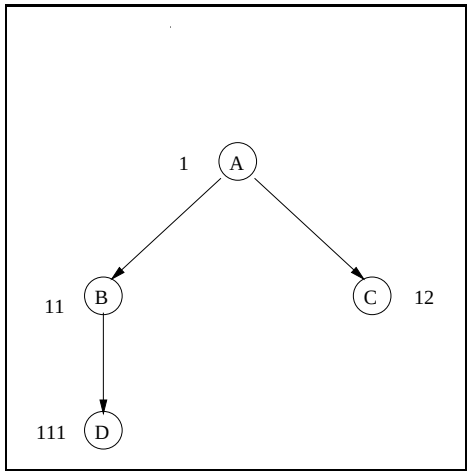

Figure 4.9: Third Session Graph.

Referrer	Target
-	aerdem
aerdem	aerdem/courses
aerdem/courses	aerdem/courses/cs101
aerdem/courses/cs101	aerdem/courses/cs101/grades
aerdem/courses/cs101/grades	aerdem/courses/cs101/grades/midterm *
aerdem/courses/cs101/grades/midterm	aerdem/courses/cs101/grades *
aerdem/courses/cs101/grades	aerdem/courses/cs101 *
aerdem/courses/cs101	aerdem/courses *
aerdem/courses	aerdem/courses/cs102
aerdem/courses/cs102	aerdem/courses/cs102/lectNotes
aerdem/courses/cs102/lectNotes	aerdem/courses/cs102 *
aerdem/courses/cs102/	aerdem/courses *
aerdem/courses	aerdem/courses/cs101 *
aerdem/course/cs101	aerdem/course/cs101/grades *

Table 4.6: First Completed Path Requests with Cache Referrals.

`research, master` } , Third completed path is given in Table 4.8. Third path: {`tugrul, grades, final, grades, tugrul, books` } .

4.4 Extracting Site Link Map

Each path of a specific site by different users should be identified uniquely in order to be able extract frequent sequences. URLs are not used for that purpose since manipulation of strings in mining process would require much larger memory than using IDs. Besides, site structure information will be asked for by Interpretation

Referrer	Target
gudukbay	gudukbay/research
gudukbay/research	gudukbay/research/tubitak
gudukbay/research/tubitak	gudukbay/research*
gudukbay/research	gudukbay/research/master

Table 4.7: Second Completed Session with Cache Referrals.

Referrer	Target
tugrul	tugrul/grades
tugrul/grades	tugrul/grades/final*
tugrul/grades/final	tugrul/grades*
tugrul/grades	tugrul
tugrul	tugrul/books

Table 4.8: Third Completed Session with Cache Referrals.

Module. For all these purposes, after session paths are found, site structure is constructed and each page URL is enumerated during construction process.

Basically graph structure is used for site mapping. Root of the tree is assigned as chosen Web site for analysis. Any incoming URL should belong one of the sub-roots of this root. Sub-roots represent URL categories such as `/~tugrul`, `/~gudukbay`. Category of incoming URL is found by comparing the leftmost part of the current path URL with sub-roots. If any one of the sub-roots is contained by target URL, that indicates that target URL is in that subtree. If category is not found, a new sub-root with the leftmost part of the target URL is added to the tree (part 4.3). Search continues recursively till a match is found. After category is found, full target URL string is searched in that subtree. If it is found, it is already placed to the tree by previous sessions. If it is not found, a new node is created and added to the parent where the search is lastly ended in part 4.5.

Although link graph construction is straightforward for regular links since they have all parent-child relation, situation differs in multi-level link construction process. We define multi-level links as the links between those nodes having no parent-child relation. Cross-links are multi-level links between those nodes that are not ancestor of each other. Forward multi-level link is defined as if there is a direct link from ancestor node P to descendant node Q other than regular

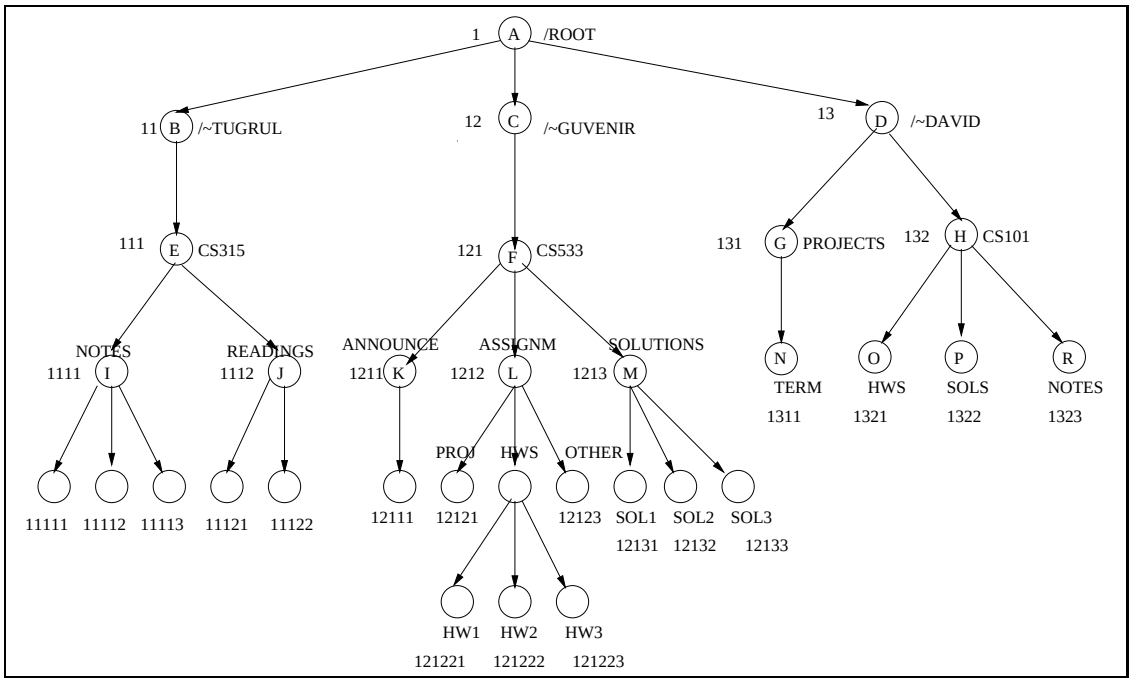

Figure 4.10: Sample Site Structure Map Constructed From Only Regular Links.

path connecting these nodes. Backward multilevel link is defined as vice verse of forward multi-level link. Link types are illustrated in Figure 4.11. Source node is the node having link and referenced node is the node that will be reached as the link is followed.

Since we assumed that site structure information is not given handling multi-level links is troublesome compared to regular links. During site link graph construction process when a multi-level link, assume cross-link in this case, is to be added to the graph. If the node where the link points already exists in the graph that node is searched in the graph and assigned cross-link of the other node having cross-link pointer. However if referenced node is node added to the graph previously, some assumptions should be made about the path connecting the source node and the referenced node. Although backward path part of the connecting path is already constructed, forward path should be added to the graph since the referenced node should exist in the graph in order to be referenced.

We assumed that each part between slashes of the referenced node URL represent one level of link in the graph and for each level we add the nodes represented by that levels. However, Web site designers may not use such URL

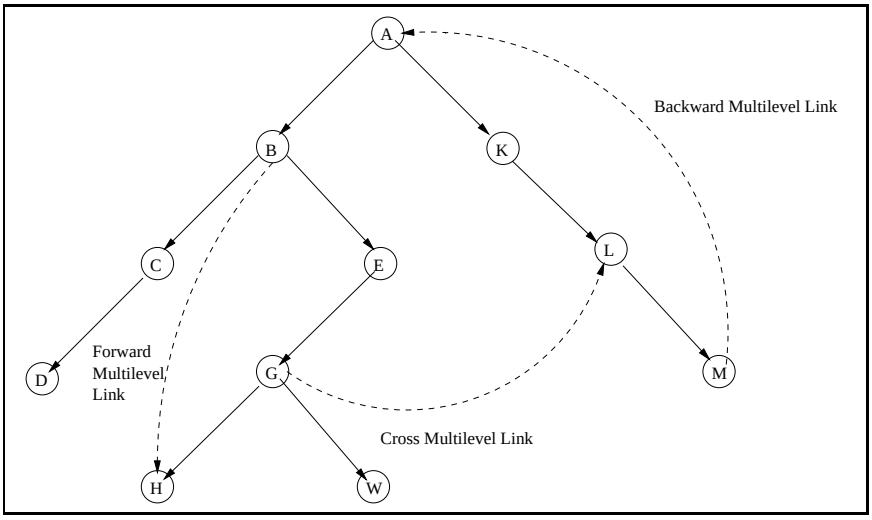

Figure 4.11: Link Types For Site Map Construction.

naming convention. For that case some extra nodes would be added to the graph and the paths would be represented longer then it is. However, eventual interested paths would be obtained correctly. Same argument runs for forward-multiple links but no path construction is required for backward-multiple links. In Figure 4.12 lintype is determined and regular paths are handled. For multi-level links $processMultLink$ function is called. Pseudo-code for this function is given in Figure 4.13.

$ProbeBackDirect$ probes in backward direction until $addURL$ is found. $ProbeForwDirect$ probes in forward direction until $addURL$ is found. If $addURL$ is found in forward the path, there is no need any path construction. However, if $addURL$ is not found, that indicates this path is followed first time until so far and addURL is tokenized and added by $ConstructForwPath$ function.

If $linkType$ is backward multiple link, just $addURL$ is found and added as backward link. if $linkType$ is forward multiple link, algorithm probes until a match found or at least for the common part of $addURL$ and $currentURL$, and if a match is not found remaining path is constructed. In case of cross-link, first common ancestor of $addURL$ and $currentURL$ is found and this node is found in the backward path first. Then procedure of forward path construction is followed. As an example to cross-link, consider Figure 4.11. Assume that nodes L and M are not added to the graph yet and we are to add G to L cross-link.

Function $ConstructSiteMap(CS_1, CS_2, ..., CS_n)$

1 Let CS_i denote ith completed session.

2 Let P_j denote jth page file in current completed session.

3 Set i to 0; $rootID \leftarrow 1$

4 for each Page P in CS_i

4.1. set j to 0

4.2. $currentNode \leftarrow$ find owner subroot for page P_j

4.3. if $currentNode = NULL$

4.3.1. create a new $subroot$

4.3.2. $currentNode \leftarrow subroot$

4.3.3. $currentNodeID \leftarrow rootID \bigoplus j$

4.4. $notFound \leftarrow true$

4.5. $linkType = FindLinkType(currentNode, newURL)$

4.5. if $(linkType = multilevelLink)$

4.5.1. call $processMultLink(newURL, G)$

4.6. while($(notFound)$ and $(currentNode.Child < lastChild)$ and $(linkType = RegularLink)$)

4.6.1. if $(currentNode.child \subseteq P_j)$

4.6.1.1. $currentNode \leftarrow currentNode.child$

4.6.2. if $(currentNode.child = P_j)$

4.6.2.1. $currentNode \leftarrow currentNode.child$

4.6.2.2. $notFound \leftarrow false$

4.6.3. else

4.6.3.1. $currentNode \leftarrow currentNode.sibling$

4.6.4. $childCount++$

4.7. if $notFound = true$

4.7.1. create a NewNode

4.7.2. $notFound = false$

4.7.3. $NewNodeID \leftarrow parentID \bigoplus childCount$

Figure 4.12: Site Map Construction Algorithm

Function $ProcessMultLink(linkType, sourceNode, addURL, G)$

1 $if(linkType = BackwardLink)$

1.1. $tempNode = ProbeBackwDirect(currentNode, addURL)$

1.2. set $currentNode = tempNode$

1.3. add $currentNode$ to $sourceNode$ as $BackwardLink$

2 $if(linkType = ForwardLink)$

2.1. call $tempNode = ProbeForwDirect(currentNode, addURL)$

2.2. call $currentNode = ConstructForwPath(tempNode, addURL)$

2.3. add $currentNode$ to $sourceNode$ as $ForwardLink$

3 $if(linkType = CrossLink)$

3.1. $comAncest = commonancestorofcurrentURLandaddURL$

3.2. call $tempNode = ProbeBackwDirect(comAncest, addURL)$

3.3. call $currentNode = ConstructForwPath(tempNode, addURL)$

3.3. add $currentNode$ to $sourceNode$ as $CrossLink$

Figure 4.13: Site Map Construction Algorithm With Multi-level Links.

We first probe to node A in backward direction and A to K in forward direction. After this point we construct K to M path and assign node M as the cross-link of node G.

4.5 Page View Time Computing

Computation of page view timings is important issue for interest criterion. However several issues should be handled. Main issues are listed below:

- handling figures embedded inside pages,
- handling multi-frame pages,
- handling cache referrals, and
- computing changing download rate.

In order to be able to compute viewing time of a page, we need following parameters:

- text size of the page: used to compute text download time,
- total figure size included: used to compute figure download time,
- request time of the page: used as starting time of download time, and
- following page request time: used as terminating time of page view time.

We can explore text size directly from log entry during parsing process. Figures embedded inside, say page P, is requested one by one after text part of the page is loaded by using referrer page as P. Therefore we keep size of all figures with referrer with P inside page P. Although in general figures are immediately requested after text body of page is loaded, for multi-frame case figures are requested after all frame text bodies are loaded.

As to ending time of page views, we assumed new page request time as end of view time of the current page. End time of multi-frames needs special care.

Although frames constituting the page are requested successively and loaded, end time of each frame excluded index page is new page request not the successive request frame time. However patterns of multi-frame page request and trial kind of page requests are the same. Since both pattern have the same parent-child relation between first request and the other requests. Two similar patterns are given in Figure 4.10 and Figure 4.9. Page in Figure 4.10 is composed of three frames where example in 4.9 represents user requesting pages under directory `tugrul`.

Referrer	Target
-	tugrul
tugrul	tugrul/grades
tugrul	tugrul/projects
tugrul	tugrul/homeworks

Table 4.9: Trial Kind of Page Requests.

Referrer	Target
-	tugrul
tugrul	tugrul/mainFrame
tugrul	tugrul/leftFrame
tugrul	tugrul/topFrame

Table 4.10: Multi-frame Page Request Pattern.

After such a pattern is captured, in order to distinguish these two cases we proposed a simple heuristic, $multFrameHeuristic$. For these patterns name the first request as the *parent* and the others as *child*. This heuristic considers page request timings and propose that if this pattern represents multi-frame structure, frames should be immediately requested in a pre-determined interval,smaller than $timeLimit$, after index page is loaded without requesting any other page or any figure belonging to children. If time difference between any two page requests is measured to be greater than this $time_limit$ this pattern is considered as trial kind of path. For this purpose we first captured such patterns and distinguished them in $computeTimeLine$ function and assigned computed all parameters stated previously. $computeTimeLine$ function is given in Figure 4.14.

In this function, in order to determine that how many pages have multi-frame structure and we kept status of the multi-frame. At any point spoiling

condition of being multi-frame structure algorithm turns back to default case, -1, and continues to process the following pages. Status 0 represents multi-frame having only single frame, status 1 having two frames and status 2 having three frames respectively. Status change conditions are checked according to the $multFrameHeuristic$. Download rate is updated inside this function for each page request if possible. If there appears successive load of frames or figures, time between successive such requests give only load time of previous request. Arithmetic mean is calculated of lastly measured download rate with the current one.

All time-line entries are processed in $computePageViewTime$ function. This function computes text and figure download time. Total duration time is computed as difference of new page request time minus current page request. Total load time is decreased from total duration of the page and assigned as page view time. Cache referrals time-lines are not computed previously so done here. Total duration of each cache referral between pages at starting and ending page request is assigned as total duration between start and end pages divided by number of cache referrals in that interval. Since these pages are loaded very fast, we considered some small fraction f of total duration of each cached page is used for load time and remaining part as the view time.

Function $ComputeTimeLine$

1 status = -1 ; topPageIdx = 0

2 for each page P in the session

3.1. if (status = -1)

3.1.1. if $requestisFigure$
assign $requestfigInfo$ to $prevPage$

3.1.3. else if $prevTargetURL = Referrer$

3.1.3.1. $t_{end}^{prevPage} = t_{req}^{P}$

3.1.3.2. set $prevURL$ to $parentURL$

3.1.3.3. set $pages[0]$ to P info

3.1.4. else if ($notFigure$)

3.1.4.1. $t_{end}^{prevPage} = t_{req}^{P}$

3.1.4.2. set $TempSess[++topPageIdx]$ to P info

3.2. if (status = 0)

3.2.1. if $referrer = parentURL$
if ($t_{req}^{P} - t_{req}^{parent} > MultFrameThrsh$)
$t_{end}^{prevPage} = t_{req}^{P}$
set $TempSess[++topPageIdx]$ to $pages[0]$ info
set $TempSess[++topPageIdx]$ to P info
else
set $pages[1]$ to P info

3.2.2. else if $prevTargetURL = Referrer$

3.2.2.1. $t_{prevPage}^{end} = t_P$

3.2.2.2. set $pages[0]$ to

3.2.4. else if $prevTargetURL = Referrer$

3.2.4.1. $t_{prevPage}^{end} = t_P$

3.2.4.2. $setTempSess[++topPageIdx]$ to $pages[0]$ info

3.2.4.3. $setpages[0]$ to P info

3.2.5. else if ($isFigure$)

3.2.5.1. set $figinfo$ to $pages[0]$

3.2.6. else if $prevTargetURL = Referrer$

3.2.4.1. $t_{prevPage}^{end} = t_P$

3.2.4.2. set $TempSess[++topPageIdx]$ to $pages[0]$ info

3.2.4.3. set $TempSess[++topPageIdx]$ to P info

Figure 4.14: Time Line Computation Function.

Function $ComputePageViewTime$

1 Let $multFrameThrsh$ denote time threshold for multi-frame structure

2 Let $downRate$ denote current download rate

3 Let $pageViewTime$ current page view time

4 for each session $i < topSessIdx$

4.1. if $domainisInternal$

4.1.1. $multFrameThrsh = InternThrsh$

4.1.2. $downlRate = InternRate$

4.2. else if $domainisexternal$

4.2.1. $multFrameThrsh = ExternThrsh$

4.2.2. $downlRate = ExternRate$

4.3. $topTIdx = TempSess[i].topIdx$

4.4. for $k = 0$ to $topTIdx$

4.4.1. if $(currRequestnotFigure)$

4.6.1.1. $textDownDur = textSize/downRate$

4.6.1.2. $pageViewTime = totalDuration - textDownDur$

4.6.2. else if $(currRequestisfigure)$

4.6.2.1. $textDownDur = textSize/downRate$

4.6.2.2. if $(browserType = MSIE)$

A. $figDownDur = totFigSize/2 * downRate$

4.6.2.3. if $(browserType = Netscape)$

B. $figDownDur = totFigSize/downRate$

4.6.2.3. $pageViewTime = (newPageReqTime + currPageReqTime) - (textDownDur + figDownDur)$

Figure 4.15: Page Time View Computation Function.

Function $ComputeCacheViewTime$

1 Let $multFrameThrsh$ denote time threshold for multi-frame structure

2 Let $downRate$ denote current download rate

3 Let $pageViewTime$ current page view time

4 for each session $i < topSessIdx$

4.1. $topTIdx = TempSess[i].topIdx$

4.2. for $k = 0$ to $topTIdx$

4.2.1. if $(TempSess[i].page[k].isFromCache = true)$

4.2.1.1. $startTime = prevPageAccTime$

4.2.1.2. k++

4.2.1.3. while $(TempSess[i].page[k].isFromCache = true)$
$numOfCachedPage++$

4.2.1.4. $endTime = currPageTime$

4.2.1.5. $Interval = (startTime - endtime)$ -
$(textDownTime_{prevPage} + figDownTime_{prevPage})$

A. $figDownDur = totFigSize/downRate$

4.6.2.3. for all cached pages and $prevPage$
$pageViewTime = Interval * (viewRatio)$

Figure 4.16: Page View Time Computation Function of Cache Referrals.

Chapter 5

Mining and Interpretation Process

All data to be mined and interpreted is prepared and processed by Usage Processor Module. Sequence data and time data are extracted by Usage Processor together. Although frequent sequence mining from Web data is addressed so far, higher level interpretation of frequent sequences to convert useful knowledge has attracted less focus. Frequent sequence mining for customer data can be used directly for understanding which products are favorite ones since there is no possibility of purchasing products in the wrong order, situation for Web data is different. Although mining frequent sequences from Web data is an important task and also for customer data, frequent sequences themselves represent not exactly useful knowledge. Main reason for this is that if Web site design does not meet user expectations and users makes several trials for reaching their targets some of the frequent sequences mined would contain sequences belonging such redundant trials due to misleading structure of the Web site. Therefore frequent sequences for Web data needs more higher level of interpretation. However we are not able to interpret basing only sequence information, so we incorporated time domain for the interpretation process. In case of each purchase of a customer shows interest of him for that item, Web user may access some pages since he is mis-guided or since that page is on the reaching path to target. Therefore we need to understand whether user is interested in the accessed pages one by one. By this way we can understand whether user still tries to reach his target or found his target or jumping to any other related pages.

Interpretation Module gets processed data prepared by Usage Processor Module and first finds frequent sequences by using WebSpade and also embeds path info site structure and then passes over sessions once more in order to find forward, cross-links and misleading guidance points and these are presented to Web designer.

5.1 WebSpade

We updated efficient sequence mining algorithm $SPADE$ proposed by Zaki [29] in order to extract frequent Web sequences as WebSpade. There are several reasons for choosing $SPADE$ for our interpretation framework:

- $SPADE$ performs fast sequence intersections by using vertical ID-lists. Number of intersections decreases very much as length of intersection sequences increase.
- $SPADE$ decomposes sequences into equivalence classes and processes these classes independently.
- $SPADE$ provides memory-efficient intersections by intersecting only generating subsequences.

For more detailed information about $SPADE$ algorithm please refer to [29]. First reason for updating $SPADE$ is that although during a session each occurrence of an item is added to support of that item once, each page access of Web sequence is added to support of that page. Consider sequence in Table 5.1.

TIME	ITEM
10	A
20	B
30	C
40	B
50	A
60	D

Table 5.1: Support Count For a Sample Sequence.

Although supports of items A and B is 1 for customer data, their supports for Web data is 2 since they occurred twice. For customer data only important thing is existence or not of that item in sequence.

Since page access timings are strictly ordered and user can not access more than one pages at the same time, intersections for Web sequence data are in temporal format only and no itemset to itemset intersection or itemset to temporal sequence intersection occurs. Assume that customer sequences are such that: $A \rightarrow BC \rightarrow D$ and for Web data: $A \rightarrow B \rightarrow C \rightarrow D$. Frequent two and three length sequences are given as: Customer sequences: $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $B \rightarrow C$, BC, $C \rightarrow D$ $A \rightarrow BC$, $BC \rightarrow D$, $A \rightarrow B \rightarrow D$, $A \rightarrow C \rightarrow D$.

Web sequences: $A \rightarrow B$, $A \rightarrow C$ $A \rightarrow D$, $B \rightarrow C$, $C \rightarrow D$, $A \rightarrow B \rightarrow C$, $A \rightarrow C \rightarrow D$, $B \rightarrow C \rightarrow D$, $A \rightarrow B \rightarrow D$.

Input to $SPADE$ is list of timestamps, number of items and items respectively. Since URLs would occupy much space compared to IDs in memory during intersections operations, we used structure ID of each page as representing page ID and page access time for time-stamp of the item. Without dealing with any caching issues, consider the following path and input provided to WebSPADE in Figure 5.1.

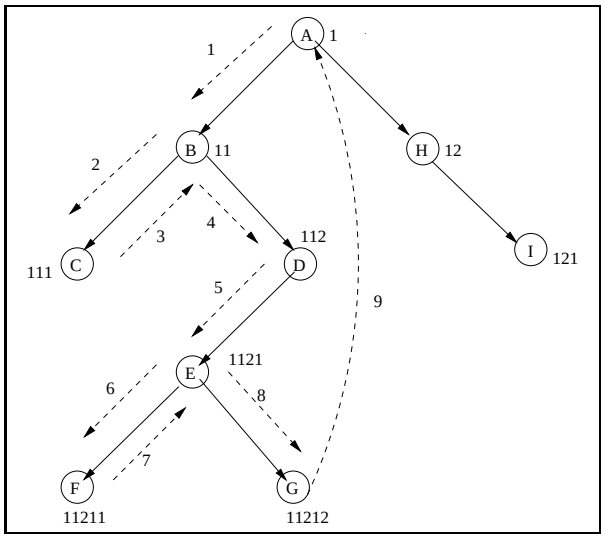

Figure 5.1: Link Types For Site Map Construction.

Input given to spade is (ID, TimeStamp) pairs as: (1, 10), (11, 20), (111, 30), (11. 40), (112, 50), (1121, 60), (11211, 70), (1121, 80), (11212, 90), (1, 100). Timestamps are simplified.

WebSPADE finds and return frequent sequences. These frequent sequences are interpreted by Interpretation Module.

5.2 Embedding Time Domain

Although frequent sequences are mined by $WebSPADE$, still we have to find which sequence items are interested and which ones are not. Because, as discussed in Chapter 3, interpretation of sequences changes with respect to time domain. Therefore we embedded time information into site map structure. Basically, those nodes having frequent sequences reaching to them keeps previous node(s) and their supports from which link should be added. This knowledge is used during distinguishing process of frequent sequences by using site map structure.

5.2.1 Adding Forward Shortcuts

This kind of pattern is understood in such cases that user successively follows links, say beginning from page P, in forward direction and interests in a page, say page Q, and spends relatively larger time in that page. Accesses previous to this interested page are assumed to be for reaching page Q. This implies that if there exists a forward shortcut from page P to page Q user would directly follow this shortcut and number of clicks for reaching page Q would be decreased as much as length of path between these pages. When this pattern is captured this information is embedded into page Q. In node representing page Q is written URL of page P, source node from which link should be added, and its count is increased by one. By this way if there occurs such a pattern greater than minimum support, then "link should be added from P to Q" suggestion is proposed to Web site designer. This kind of pattern is captured previous to user backward navigation since after user begins backward navigation that pattern contains more complicated information. It is addressed in the second part.

5.2.2 Adding Cross-links Between Related Pages and Capturing Misleading Points

When user begins following backward paths there are two possible interpretations: either user could not find the target and still tries other paths and other alternative is that user jumps to related pages. First trial page is the page where user expected to find target and if at this page he could not find the page there should be added a link from this page to the target page. If he tries other alternatives after first trial that indicated that user still could not find and searches the target. This search continues until user finds the target or give ups searching. If there appears at least one interested point previous to give up, a link should be added from first trial point to interested point. This case is illustrated in Figure 5.2. We define backward turning point, BTP, as the page where user shifts from backward navigation to forward navigation and forward turning point, FTP, as the point where user shifts from forward navigation to backward navigation. Session Figure 5.2.

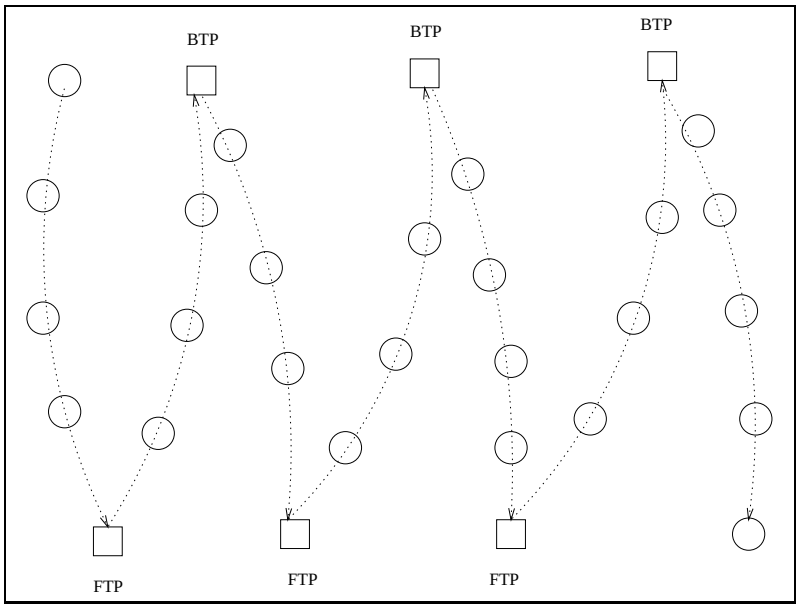

Figure 5.2: Link Types For Site Map Construction.

If user successively interests more than one FTP, these turning points are related concepts and in order to enhance user reach faster to later interested FTP from previous FTP, a cross- link should be added from previous FTP to later FTP. However, cross-links do not have to be added as pointing to always

FTPs. Because user may be interested in pages after last BTPs. Therefore we should keep pointers to all interested pages in forward direction navigation after the last BTP and cross-links from FTPs can be added as pointing to such pages. In order to put all such links we have to keep FTPs and all interested pages in the forward path.

In case of user could not find target, he backtracks and tries other alternatives and for each such case user passes through a BTP. Therefore for each such pass we kept "trial support" indicating how many users passed through this page since they could not find previously target page. If support count is greater than pre-specified min-support value, such pages are considered as mis-guiding pages to Web site designer so that its content should be updated for correct guidance. Algorithm for finding cross-links and misleading points is given in Figure 5.3.

In this algorithm, let FTP denote Forward Turning Point and BTP Backward Turning Point, $interestThrh$ denote interest threshold, $currPage$ current page in the session, $currNode \rightarrow crossLinkInfo[]$ denote cross link information kept in each node, $FTPList[]$ denote Forward Turning Points kept during a session, $lastInterFP$ denote Last Interested Forward Page.

In part 1.1, forward path is considered for putting shortcuts to favorite pages. When an interested page is found in the forward path previous to reaching first Backward Turning Point, if there exists a previously interested page that page is assigned as source node of the link and the current page as the pointed node (1.1.2). If interested node is the first one, starting page is considered as the source node for link (1.1.1). In part 1.5, all BTPs and FTPs are found and cross-link information of each node is assigned at nodes representing FTPs.

As the last step of the interpretation process, we find and traverse over the site structure frequent sequences and we check forward turning points on site structure that are also on frequent sequences. If support of forward or cross-links of this pages are greater than min-support, we specify those pages as forward or cross-links to the current forward turning point and add to suggestion list.

Function $FindCrossLinks()$

1 while not FTP_1 not reached

1.1. if $(t_{view}^{currPage} > interestThrsh)$

1.1.1. if $firstinterestednode$

add $currPage- > shortcutInfo[++k].URL\ startpage$

set $currPage- > shortcutInfo[k].support++$

set $lasInterFP\ currPage$

1.1.2. else

1.1.2.1. add $currPage- > shortcutInfo[++k].URL\ lastInterFP$

1.1.2.2. set $currPage- > shortcutInfo[k].support++$

1.1.2.3. set $lasInterFP\ currPage$

1.2. set $FTPList[0]$ to FTP_1

1.3. if $(t_{view}^{currPage} \geq interesThrsh)$

1.3.1. $FTPList[0].count1++$

1.4. else

1.4.1. $FTPList[0].count0++$

1.5. while ($currPage$ not last session page)

1.5.1. while $currPage$ is not equal BTP

1.5.1.1. pass to next page, $currPage = nextPage$

1.5.2. clear $ForwInterPageList[]$

1.5.3. $BTP[++last]_{count}++$

1.5.4. while $currPage$ is not equal FTP

1.5.4.1. pass to next page, $currPage = nextPage$

1.5.4.2. if $(t_{view}^{currPage} > interesThrsh)$

$ForwInterPages[currPage].count1++$

$ForwInterNodes[++lastNode] = currPage$

1.5.4.3. else $FTPList[0].count0++$

1.5.5. $FTPList[lastFTP] = currPage$

1.5.6. $lastInterFTP = currPage- > CrossLinkInfo[lastCross]$

1.5.7. if $(t_{view}^{FTP}$ ¿ $interesThrsh)$

1.5.7.1. set $lastInterFTP = FTP$

1.5.8. if $currPage = lastSessionPage$

1.5.8.1. process cross-link Information of $ForwInterPageList[]$

Figure 5.3: Finding Frequent Forward and Cross-links.

5.3 Experimental Results

In this chapter we illustrate the sample runs of the usage miner framework. We selected Web site of CS department of Bilkent University and used server log files of CS domain Web server. Since most of the Web pages in this domain are continuously used, it is possible to capture patterns of average users. However one disadvantage of CS domain is that Web sites under this domain is not much complicated so extracting very clear picture of user misleadings is difficult task.

5.3.1 Sample Outputs

In this part we give sample suggestions given by usage miner framework. Table 5.2 shows putting forward shortcuts for shortening paths to reach popular pages. *Source* gives the page where link to be added and *Destination* gives the page where this link points. Table 5.3 gives sample suggestions for adding cross-links between related pages.

SOURCE	DESTINATION
/~akman/	/~akman/jour-papers/sigart/
/~canf/cs533/	/~canf/cs533LectureNotes/Week3Notes.html
/~bgenc/	/~bgenc/cs319/projects.htm
/~david/cs492/	/~david/cs492/98asep/98asep.htm
/~canf/	/~canf/cs533/hw2sol.doc
/~will/courses/	/~will/courses/homeworks/homework4.html
/~agursoy/courses.html	/~agursoy/courses/cs543info/projects.html
/~berkant/	/~berkant/courses/hw/hw2.html
/~guvenir/courses/	/~guvenir/courses/cs315/grades02.html
/~gudukbay/	/~gudukbay/cs465/assignments/asst2.html
/~will/courses/	/~will/courses/cs223/examgrades.html
/~ugur/teaching/	/~ugur/teaching/cs319/grading.html
/~akman/courses/	/~akman/courses/cs461/461books2.txt

Table 5.2: Sample Results For Adding Forward Links.

SOURCE DESTINATION
/∼will/courses/cs223/examgrades.html /∼will/courses/cs223/quizgrades.html
/∼canf/cs533/LectureNotes/Week3.html /∼canf/cs533/LectureNotes/Week4.html
/∼guvenir/courses/cs315/grades02.html /∼guvenir/courses/cs315/grades01.html
/∼bgenc/cs319/cs31901grades.html/ /∼bgenc/cs319/cs31902grades.html
/∼will/courses/cs223/homeworks/homework4.html /∼will/courses/cs223/labassignments.html
/∼ugur/teaching/cs319/announce.html /∼ugur/teaching/cs319/assignment.html
/∼ubora/cs223/cs223-2quiz.html /∼ubora/cs223/cs223-2hw.html

Table 5.3: Sample Results For Adding Cross Links.

Chapter 6

Conclusion and Future Work

With this work, we addressed time based temporal sequence interpretation problem on Web log files. Frequent sequences are interpreted so that interesting and uninteresting frequently occurring patterns are distinguished. In this work we make use of backward sequences in addition to forward sequences and incorporated time domain as an interest criterion in interpretation of navigation paths. Backward sequences are considered foot-prints of users indicating jumps since target is not found, or moves to related pages. Different interpretations are based on time domain in addition to temporal knowledge. We calculated page view time for each page by considering cases such as figures embedded inside the page, multi-frame structure and cache referrals. Download timings of each figure belonging to requested page is computed in addition to the text body of the page. Some web pages are designed as containing multi-frames and this structure is captured by analyzing the successive page requests since each frame is requested separately through the index page of this multi-frame. Besides, some heuristics are used in order to distinguish multi-frame structure from forward-backward kind of trials.

Since page view computation is based on download time of the whole body of the page, size is also kept and network transmission rate is computed dynamically through each session by considering successive figure or text downloads of a page. page viewing timings of cache referrals are computed Even though they are not recorded at web logs. Relative page view time is used as a criterion for interest of users for that page. Because download time of the same page may differ at different times depending on the network transmission rate. Time domain is

incorporated to interpretation of backward and forward paths. Clear user session extraction process is very critical for getting useful picture of navigation patterns. During extraction of whole user sessions multi-user cases or multiple browser use of the same user issues are considered. Cache referrals are also found by using efficient methods.

In this work, we proposed a model, WebModel, that interprets the frequent navigation patterns and developed usage mining processing framework extract user sessions, finds frequent sequences and interprets them.

As a future work, incremental version of this framework could be developed. Because links should be continuously updated. Besides, unused links on the site should be removed so that Web site does not become over-linked due to continuous link add. Unused links can be determined by keeping link support for each link. We can use interest criterion for link support. Links on redundant paths should not increment support of these links.

When finding related pages, many links can be suggested for adding between different hierarchical levels of different subtrees. If all of these links are processed sites may become over-linked leading confusion rather than better guiding. Therefore if there exists sufficient number of cross-link suggestions between related page clusters, a higher level link can be added between these clusters.

Bibliography

[1] S. K. Madria, S. S. Bhowmick, W. K. Ng, and E. Lim, "Research Issues in Web Data Mining," *Proc. First Int'l Conf. on Data Warehousing and Knowledge Discovery (DaWaK '99)*, M.K. Mohania and A.M. Tjoa, eds., pp. 303-312, 1999.

[2] B. Mobasher, N. Jain, E. Han, and J. Srivastava. "Web Mining: Pattern Discovery From World Wide Web Transactions", (TR 96-050), 1996.

[3] M. S. Chen, J. S. Park, and P. S. Yu. "Data Mining for Path Traversal Patterns in a Web Environment", *In 16th International Conference on Distributed Computing Systems*, pp. 385-392, 1996.

[4] M. Spiliopoulou and L. C Faulstich, "Wum: A Web Utilization Miner", *In EDBT Workshop WebDB98*, Valencia, Spain, 1998, Springer Verlag.

[5] J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu, " Prefixspan: Mining Sequential Patterns Efficiently by Prefix Projected Pattern Growth", *In Proceedings of International Conference on Data Engineering (ICDE'01)*, 2001, Heidelberg, Germany.

[6] J. Borges and M. Levenue, "Data Mining of Users Navigation Patterns", *In Web Usage Analysis and User Profiling*, volume 1836, pp. 92-111.

[7] A. Nanopoulos, Y. Manolopoulos, "Finding Generalized Path Patterns for Web Log Data Mining", Technical Report, Aristotle University, 2000.

[8] J. Srivastava, R. Cooley, M. Deshpande, P. Tan, "Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data", *ACM SIGKDD Explorations*, Volume 1, Issue 2, Page 12, Jan 2000.

[9] O. R. Zaiane, M. Xin, and J. Han., "Discovering Web Access Patterns and Trends by Applying Olap and Data Mining Technology on Web Logs", *In Advances in Digital Libraries*, pp. 19-29, Santa Barbara, CA, 1998.

[10] Y. Fu, K. Sanhau, and M Shih, "Clustering of Web Users Based on Access Patterns", *International Workshop on Web Usage Analysis and User Profiling (WEBKDD99)*, San Diego, CA., 1999.

[11] C. Shahabi, A. M. Zarkesh, J. Adibi, and V. Shah. "Knowledge Discovery from Users Web-page Navigation", *In Workshop on Research Issues in Data Engineering*, Birmingham, England, 1997.

[12] J. Punin, M. Krishanomoorthy, and M. J. Zaki, "Web Usage Mining: Languages and Algorithms", *Studies in Classification, Data Analysis and Knowledge Organization*, Springer-Verlag, 2001

[13] V. Almeida, A. Bestavros, M. Crovella, and A. Oliveira. "Characterizing Reference Locality in WWW", *Technical Report TR-96-11*, Boston University, 1996.

[14] S. Schechter, M. Krishnan, and M. D. Smith, "Using Path Profiles to Predict http Requests", *In 7th International World Wide Web Conference*, Brisbane, Australia, 1998.

[15] M. Perkowitz and O. Etzioni, "Adaptive Web Sites: Automatically Synthesizing Web Pages", *In Fifteenth National Conference on Artificial Intelligence*, Madison,WI, 1998.

[16] M. Perkowitz and O. Etzioni, "Adaptive Web Sites: Conceptual Cluster Mining", *In Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

[17] T. Nakayama, H. Kato, Y. Yamane, "Discovering the Gap between Web Site Designer's Expectations and User's Behavior", *Proc. of the Ninth Int'l World Wide Conference*, Amsterdam, 2001.

[18] T. Joachims, D. Freitag, and T. Mitchell, "Webwatcher: A Tour Guide for the World Wide Web", *In the 15th International Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[19] S. Turner, "Analog", `http://www.analog.cx`, 2000.

[20] B. Mobasher, R. Cooley, and J. Srivastava, "Creating Adaptive Web Sites Through Usage Based Clustering of URLs", *In Knowledge and Data Engineering Workshop*, 1999.

[21] J. F. Roddick, M. Spiliopoulou, "A Survey of Temporal Knowlege Discovery Paradigms and Methods", *to appear in IEEE Transactions on Knowledge and Data Engineering.*

[22] R. L. Blum, "Discovery, Confirmation and Interpretation of Causal Relationships from a Large Time-Oriented Clinical Database: The RX Project", *Computers and Biomedical Research*, vol. 15, no. 2, pp. 164-187, 1982.

[23] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases", *Proc. Int'l Conf. Foundations of Database Organisation and Algorithms*, (FODO '93), D. Lomet, ed., pp. 69-84, 1993.

[24] R. Agrawal, K. I. Lin, H. S. Sawhney, and K. Shim, "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases", *Very Large Databases*, pp. 490-501, 1995.

[25] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaot, "Querying Shapes of Histories", *Proc. 21st Int'l Conf. Very Large Databases(VLDB '95)*, U. Dayal, P. M. D. Gray, and S. Nishio, eds., pp. 502-514, 1995.

[26] R. Agrawal and R. Srikant, "Mining Sequential Patterns", *Proc.11th Int'l Conf. Data Eng.*, P. S. Yu and A. S. P. Chen, eds., pp. 3-14, 1995.

[27] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases", *Proc. ACM SIGMOD Int'l Conf. Management of Data*, vol. 22, pp. 207-216, 1993.

[28] H. Mannila and H. Toivonen, "Discovering Generalised Episodes Using Minimal Occurences", *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD-96)*, pp. 146-151, 1996.

[29] M. J. Zaki. "SPADE: An Efficient Algorithm for Mining Frequent Sequences", *Machine Learning Journal*,D. Fisher, edt., 42(1/2):31-60, Jan/Feb 2001.

[30] M. J. Zaki, N. Lesh, and M. Ogihara, "Planmine: Sequence Mining for Plan Failures", *Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining*

(KDD '98), R. Agrawal, P. Stolorz, and G. Piatetsky Shapiro, eds., pp. 369-373, 1998,

[31] G. M. Weiss and H. Hirsh, "Learning to Predict Rare Events in Event Sequences", *Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining (KDD '98)*, R. Agrawal, P. Stolorz, and G. Piatetsky Shapiro, eds., pp. 359-363, 1998.

[32] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: Information and Pattern Discovery on the World Wide Web", *In International Conference on Tools with Artificial Intelligence*, pp. 558-567, Newport Beach, 1997.

[33] R. Srikant, Y. Yang, "Mining Web Logs to Improve Website Organization", *Proc. of the Tenth International World Wide Web Conference*, Hong Kong, 2001.

[34] R. Cooley, B. Mobasher, and J. Srivastava, "Data Preparation for Mining World Wide Web Browsing Patterns", *Knowledge and Information Systems*, 1(1), 1999.

[35] R. Cooley, P. Tan, and J. Srivastava, "Discovery of Interesting Usage Patterns From Web Data", *Technical Report TR 99-022*, University of Minnesota, 1999.

[36] L. D. Catledge and J. E. Pitkow, "Characterizing Browsing Strategies in the World-Wide Web", *Proc. Third WWW Conf.*, Apr. 1995.