UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a doctoral thesis by

Robert Walker Cooley

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Dr. Jaideep Srivastava
Name of Faculty Adviser

____
Signature of Faculty Adviser

____
Date

GRADUATE SCHOOL

Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Robert Walker Cooley

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Dr. Jaideep Srivastava, Advisor
May, 2000

Acknowledgments

First and foremost, I'd like to thank my wife Sarah for her support in all phases of my life. I'd also like to thank my advisor Jaideep Srivastava for his advice, guidance, and patience. Mukund Deshpande and Pang-Ning Tan deserve credit for the many hours they put in designing the WebSIFT system, writing code, and running experiments. George Karypis provided the product concept clusters used to evaluate the information filter. Finally, thanks to my parents for bringing me up to be the type of person who actually completes one of these PhD things.

Abstract

Web Usage Mining is the application of data mining techniques to Web clickstream data in order to extract usage patterns. As Web sites continue to grow in size and complexity, the results of Web Usage Mining have become critical for a number of applications such as Web site design, business and marketing decision support, personalization, usability studies, and network traffic analysis. The two major challenges involved in Web Usage Mining are preprocessing the raw data to provide an accurate picture of how a site is being used, and filtering the results of the various data mining algorithms in order to present only the rules and patterns that are potentially interesting. This thesis develops and tests an architecture and algorithms for performing Web Usage Mining. An evidence combination framework referred to as the *information filter* is developed to compare and combine *usage*, *content*, and *structure* information about a Web site. The information filter automatically identifies the discovered patterns that have a high degree of subjective interestingness. The results of experiments with the information filter show that lists of thousands of discovered patterns can be reliably ranked according to interestingness, as defined by the deviation from a predefined set of expectations about the usage of a Web site. Also, the necessary steps for preprocessing clickstream data are identified, and algorithms for each of these steps are developed and tested against data from a large electronic commerce Web site. The results show that proper preprocessing, including the incorporation of knowledge about the structure of a Web site, is required to obtain meaningful usage patterns.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The use of hypertext linked displays continues to expand at an amazing rate as a medium for conducting business and disseminating information via the World Wide Web. Even with evolving standards and technology, there are many challenges that must be overcome in order to thoroughly analyze the usage of a Web site. Design of a Web site centers around organizing the information on each page and the hypertext links between the pages in a way that seems most natural to the site users in order to facilitate their browsing (and possibly purchasing). For small sites, an individual Web designer's intuition along with some straightforward usage statistics may be adequate for predicting and verifying the users' browsing behavior. However, as the size and complexity of a Web site increases, the simple statistics provided by existing Web log analysis tools [7, 8, 14] are inadequate for providing meaningful insight into how a Web site is being used. *Web Usage Mining*, which is the application of data mining techniques to large Web data repositories, adds powerful techniques to the tools available for analyzing Web site usage. Figure 1.1 shows the high level steps involved in the Web Usage Mining Process.

In Web Usage Mining, as with many data mining domains, objective measures such as *support* and *confidence* [20] are often used to limit the number of discovered rules to a manageable number. However, high thresholds on objective measures often result in patterns that only confirm the obvious, and low thresholds usually result in an unmanageable number of rules. For example, products that are offered on the

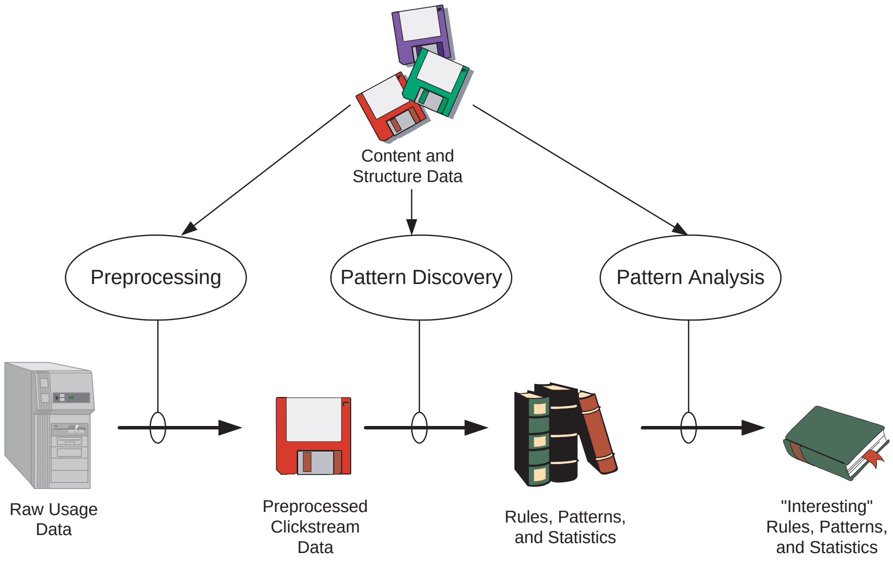

Content and Structure Data
Preprocessing
Pattern Discovery
Pattern Analysis
Raw Usage Data
Preprocessed Clickstream Data
Rules, Patterns, and Statistics
"Interesting" Rules, Patterns, and Statistics

Figure 1.1: High Level *Web Usage Mining* Process

same page view will often appear in discovered patterns as being highly correlated. However, these patterns are not particularly useful for performing a task such as personalization or Web site design. Dynamically adding a recommendation for a product that is already in front of the user does not necessarily justify the expense of data mining. One of the main goals of pattern analysis is to assist an analyst (or process in the case of personalization or some other real-time task) in identifying patterns that are of potential interest. The definition of interestingness is often subjective, and based on the domain knowledge of the particular analyst.

Therefore, in order to be truly useful, a method for ranking the results of Web Usage Mining according to subjective interestingness is required. This interest measure must be able to compare and combine the evidence gathered from the pattern discovery algorithms, with domain knowledge about how a Web site is expected to be used. Along with the challenge of defining a subjective interestingness framework comes the challenge of automatically quantifying domain knowledge. Even an ideal framework is useless without a set of domain knowledge to compare to the discovered patterns. Ideally, data sources other than usage data, that are associated with every Web site, can be used to automatically generate a default set of domain knowledge.

There are many kinds of data that are involved in the Web Usage Mining process. This thesis classifies such data into the following types:

- **Content:** The *real* data in the Web pages, i.e. the data the Web page was designed to convey to the users. This usually consists of, but is not limited to, text and graphics.

- **Structure:** Data which describes the organization of the content. *Intra-page* structure information includes the arrangement of various HTML or XML tags within a given page. The principal kind of *inter-page* structure information is hyper-links connecting one page to another.

- **Usage:** Data that describes the pattern of usage of Web pages, such as IP addresses, page references, and the date and time of accesses. Typically, the usage data comes from an Extended Common Log Format (ECLF) Server log.

- **User Profile:** Data that provides demographic information about users of the Web site. This includes registration data and customer profile information.

The information provided by the data sources listed above can all be used to construct a data model consisting of several data abstractions, notably *users, page views, click-streams, server sessions*, and *episodes.* In order to provide some consistency in the way these terms are defined, the W3C Web Characterization Activity (WCA) [15] has published a draft of Web term definitions relevant to analyzing Web usage. A page view is defined as all of the files that contribute to the client-side presentation seen as the result of a single mouse “click” of a user. A click-stream is then the sequence of page views that are accessed by a user. A server session is the click-stream for a single visit of a user to a Web site. Finally, an episode is a subset of page views from a server session. These definitions will be discussed in more detail in Chapter 3.

In order to create the data model discussed above, significant amounts of preprocessing must be performed. Web Usage Mining preprocessing consists of converting the usage, content, and structure information contained in the various available data sources into the data abstractions necessary for pattern discovery. The practical difficulties in performing preprocessing are a moving target. As the technology used to deliver content over the Web changes, so do the preprocessing challenges. While each of the basic preprocessing steps remains constant, the difficulty in completing certain steps has changed dramatically as Web sites have moved from static HTML served directly by a Web server, to dynamic scripts created from sophisticated content servers and personalization tools. Both client-side tools (e.g. browsers) and server-side tools (e.g. content servers) have undergone several generations of improvements since the inception of the Web. The only piece that has remained relatively constant is the transport mechanism - the Hypertext Transport Protocol (HTTP). The fundamental characteristics of the current HTTP version are relatively unchanged from the original version. However, as the W3C works to develop the "next generation" protocol (referred to as HTTP-NG), even the "constants" such as a stateless connection may disappear (The addition of "persistent connections" is expected to improve performance and will greatly simplify Web Usage Mining preprocessing). The methods described in this thesis try to account for variations in the client-side and server-side technologies, but are heavily dependent on the implementation of transport protocol.

Once the raw usage data has been preprocessed into server sessions or episodes, a number of techniques can be used to discover patterns such as association rules [20], clusters of similar pages or users, or sequential patterns [70]. As will be discussed in Chapter 2, the discovered patterns can then be used for a variety of applications, such as:

- Site Design - reorganization of the link structure or content of the pages to reflect actual usage.

- Business/Marketing Decision Support - determination of common behaviors or traits of users who perform certain actions, such as purchasing merchandise.
- Personalization - customization of page views based on information gained about each user. This can include dynamic pricing or real-time promotions to encourage "cross-sells" and "up-sells."
- Usability Studies - determination of interface quality.
- Security - detection of "unusual" accesses to secure data.
- Network Traffic Analysis - determination of equipment requirements and data distribution in order to efficiently handle site traffic.

Each of theses applications can benefit from patterns that are ranked by subjective interestingness. Even if an entire list of patterns is ranked as subjectively uninteresting, this can be of value. For site design, this confirms that the Web site is being used as intended. For security applications, the lack of unexpected patterns indicates everything is normal. Without an automatic ranking of each rule as interesting or uninteresting, an analyst must manually sift through the entire list of results. The rare subjectively interesting rule or pattern can be easily overlooked among hundreds of uninteresting patterns. As will be shown in Chapter 8, a pattern may not even look interesting at first glance.

This thesis tests a number of hypotheses regarding the use of an evidence combination framework referred to as the *information filter*. The *information filter* is defined in Chapter 7, and automatically ranks discovered patterns according to subjective interestingness by comparing and combining *usage*, *content*, and *structure* information about a Web site. The hypotheses include:

- Frequent itemsets of page views can be reliably ranked according to subjective interestingness by comparing the itemsets to the site structure.
- Clusters of page views can be reliably ranked according to subjective interestingness by comparing the clusters to the site content.

In order to effectively test the information filter, this thesis also defines and tests a number of preprocessing algorithms and heuristics. The preprocessing hypotheses include:

- The information available in an ECLF log is sufficient for accurately identifying server sessions.
- Client-side caching of page references can be reliably inferred from the information in an ECLF log for a static Web site.
- The usage type of a page view can be inferred by the amount of time a user spent viewing the page.

An experimental Web Usage Mining system called the Web Site Information Filter (WebSIFT) system has been designed and implemented to test the hypotheses listed above. The tests confirm the two hypotheses related to the information filter, and show that lists of thousands of discovered patterns can be filtered and ranked according to subjective interestingness as defined by the deviation from a predefined set of expectations about Web site usage. The results of the tests confirm all of the preprocessing hypotheses except the first. For dynamic Web sites such as the type used for electronic commerce, a positive means of session identification, such as cookies or embedded session IDs, is necessary to accurately preprocess the usage data.

The rest of this thesis is organized as follows: Chapter 2 discusses background and related work. Chapter 3 explains the available data sources and data modeling required prior to any analysis. The various algorithms used by the WebSIFT system will be developed in Chapters 4 through 7, and results of experiments using the WebSIFT system will be presented in Chapters 8 and 9. Finally, Chapter 10 provides a conclusion.

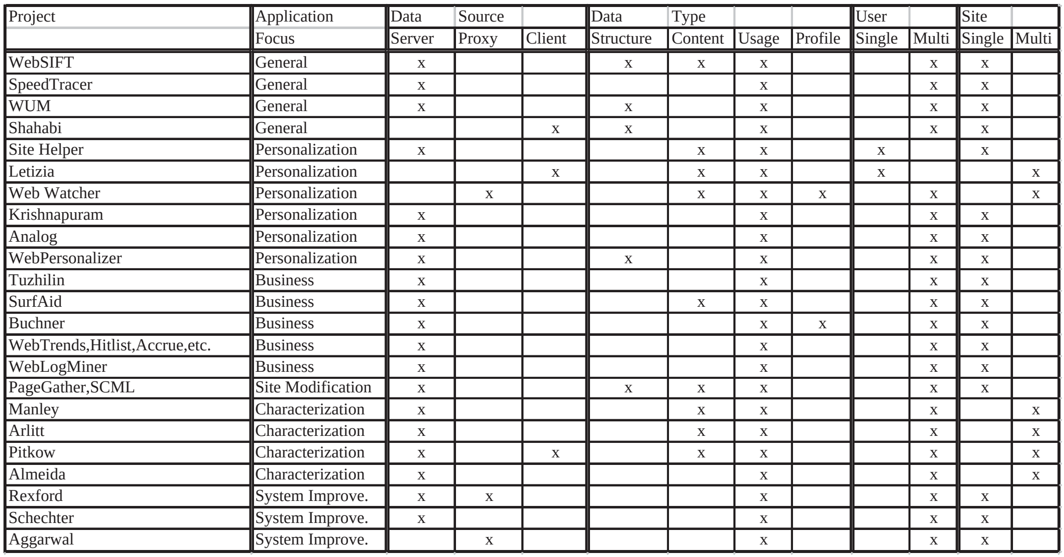
Chapter 2

Related Work

Monitoring and understanding how the Web is used is an active area of research in both the academic and commercial worlds. Web Usage Characterization or Web Metrics focuses on the properties of the entire (or a large portion of) Web and how the Web is used as a whole. Web Usage Mining or Analysis can be broadly defined as being interested in patterns of behavior for Web users. Web Usage Mining is often restricted to a single or small number of sites. In addition to Web Usage Mining, there is also much work being done to characterize the structure and content of the Web. The results of Web Structure Mining or Web Content Mining are often necessary to get meaningful analysis out of any discovered usage patterns. There is an entire body of research devoted to Data Mining and Pattern Discovery algorithms, independent of any particular domain. Since these algorithms are integral to the Web Usage Mining process, they will be addressed separately in Chapter 6. Finally, work dealing with the identification of interesting patterns from the set of discovered patterns, along with reasoning with uncertainty, is relevant to filtering the results of any Web Usage Mining effort.

2.1 Web Usage Mining

While the number of candidate dimensions that can be used to classify Web Usage Mining projects is many, there are five major dimensions that apply to every project



Project	Application	Data	Source		Data	Type			User		Site	
	Focus	Server	Proxy	Client	Structure	Content	Usage	Profile	Single	Multi	Single	Multi
WebSIFT	General	x			x	x	x			x	x	
SpeedTracer	General	x					x			x	x	
WUM	General	x			x		x			x	x	
Shahabi	General			x	x		x			x	x	
Site Helper	Personalization	x				x	x		x		x	
Letizia	Personalization			x		x	x		x			x
Web Watcher	Personalization		x			x	x	x		x		x
Krishnapuram	Personalization	x					x			x	x	
Analog	Personalization	x					x			x	x	
WebPersonalizer	Personalization	x			x		x			x	x	
Tuzhilin	Business	x					x			x	x	
SurfAid	Business	x				x	x			x	x	
Buchner	Business	x					x	x		x	x	
WebTrends,Hitlist,Accrue,etc.	Business	x					x			x	x	
WebLogMiner	Business	x					x			x	x	
PageGather,SCML	Site Modification	x			x	x	x			x	x	
Manley	Characterization	x				x	x			x		x
Arlitt	Characterization	x				x	x			x		x
Pitkow	Characterization	x		x		x	x			x		x
Almeida	Characterization	x					x			x		x
Rexford	System Improve.	x	x				x			x	x	
Schechter	System Improve.	x					x			x	x	
Aggarwal	System Improve.		x				x			x	x	

Figure 2.1: Web Usage Mining Research Projects and Products

- the data sources used to gather input, the types of input data, the number of users represented in each data set, the number of Web sites represented in each data set, and the application area focused on by the project. Usage data can either be gathered at the server level, proxy level, or client level, as will be discussed in Chapter 3. As shown in Figure 2.1, most projects make use of server-side data. All projects analyze usage data and some also make use of content, structure, or profile data. While applying data mining to the structure and content of Web sites is an interesting area of research in its own right, in the context of Web Usage Mining the structure and content of a site can be used to facilitate preprocessing, filter the input to, or output from, the pattern discovery algorithms. Results from Web Structure and Content Mining projects, such as ParaSite [102], the authoritative source and hub work of Kleinberg [53], LIRA [24], WebKB [46], or WebACE [75] can be used to cluster or classify Web pages in order to enhance a Web Usage Mining project.

The algorithms for a project can be designed to work on inputs representing one

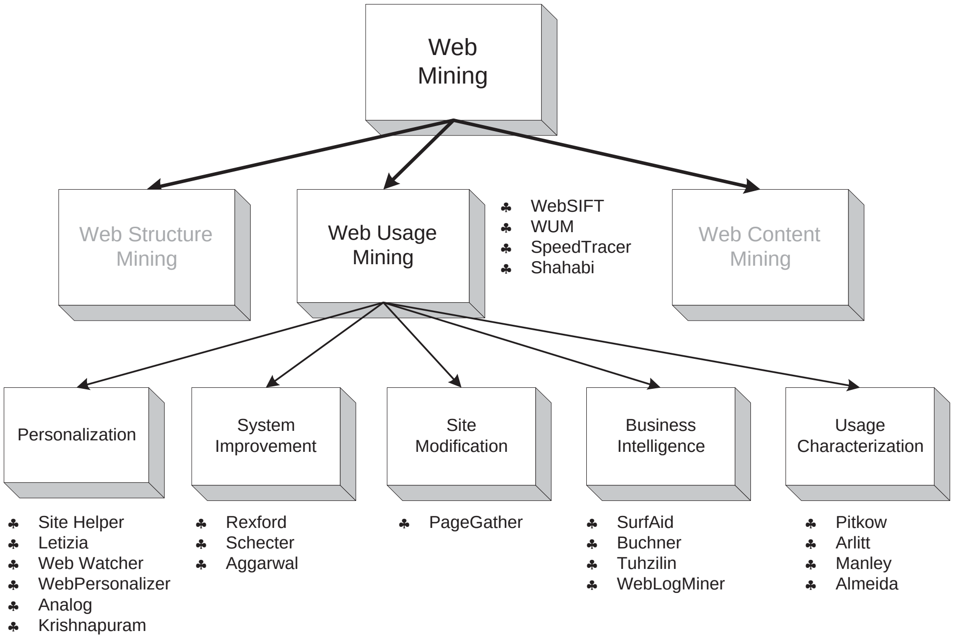


Figure 2.2: Major Application Areas for Web Usage Mining

or many users, and one or many Web sites. Single user projects are generally involved in the personalization application area. The projects that provide multi-site analysis use either client or proxy level input data in order to easily access usage data from more than one Web site. Most Web Usage Mining projects take single-site, multi-user, server-side usage data, i.e. Web server logs, as input.

As shown in Figures 2.1 and 2.2, usage patterns extracted from Web data have been applied to a wide range of applications. Projects such as WebSIFT [41, 43, 44, 45], WUM [103, 104], SpeedTracer [108], and Shahabi's work [99] have focused on Web Usage Mining in general, without extensive tailoring of the process towards one of the various sub-categories. Chen et al. [35] introduced the concept of a *maximal forward reference* to characterize user episodes for the mining of traversal patterns. A maximal forward reference is the last page requested before backtracking occurs during a particular server session. The SpeedTracer project [108] from IBM Watson is built on the work originally reported in [35]. In addition to episode identification, SpeedTracer makes use of referrer and agent information in the preprocessing routines to identify users and server sessions in the absence of additional client side information. The Web Utilization Miner (WUM) system [103] provides a robust mining language to specify characteristics of discovered frequent paths that are interesting to the analyst. In WUM, individual navigation paths called trails are combined into an aggregated tree structure. Queries can be answered by mapping them into the intermediate nodes of the tree structure. Shahabi et. al. [99, 113] have one of the few Web Usage Mining systems that relies on client-side data collection. The client-side agent sends back page request and time information to the server every time a page containing the Java applet, either a new page or a previously cached page, is loaded or destroyed.

2.1.1 Personalization

Personalizing the Web experience for a user is the holy grail of many Web-based applications such as individualized marketing for electronic commerce (e-commerce) [6], and e-commerce sites such as Amazon.com. Making dynamic recommendations to a Web user, based on her/his profile in addition to usage behavior, is very attractive for many applications, e.g. *cross-sales* and *up-sales* in e-commerce. Web Usage Mining is an excellent approach for achieving this goal, as illustrated in the WebPersonalizer system [74]. Existing recommendation systems, such as [9, 11], do not currently use data mining for recommendations, though there have been some recent proposals [19].

WebWatcher [59], SiteHelper [78], Letizia [66], WebPersonalizer [74], and work by Yan et. al. [109] have all concentrated on providing Web Site personalization based on usage information. Web server logs were used by Yan et. al. [109] to discover clusters of users having similar access patterns. The system proposed in [109] consists of an offline module that performs cluster analysis, and an online module which is responsible for dynamic link generation of Web pages. Every site user is assigned to a single cluster based on his/her current traversal pattern. The links that are presented to a given user are dynamically selected based on what pages other users assigned to the same cluster have visited. The SiteHelper [78] project learns a user's preferences by looking at the page accesses for each user. A list of keywords from pages that a user has spent a significant amount of time viewing is compiled and presented to the user. Based on feedback about the keyword list, recommendations for other pages within the site are made. WebWatcher [59] "follows" a user as he/she browses the Web and identifies links that are potentially interesting to the user. The WebWatcher starts with a short description of a user's interest. Each page request is routed through the WebWatcher proxy server in order to easily track the session across multiple Web sites and mark any interesting links. WebWatcher evaluates the interestingness of new pages based on the particular user's browsing plus the browsing of other users with similar interests. Letizia [66] is a client side agent that searches the Web for pages similar to ones that the user has already viewed or bookmarked. The WebPersonalizer [74] page recommendations are based on clusters of pages found from the server log for a site. The system recommends pages from clusters that most closely match the current session. Pages that have not been viewed and are not directly linked from the current page are recommended to the user. [76] attempts to cluster server sessions using a fuzzy clustering algorithm, which allows a page or user to be assigned to more than one cluster.

2.1.2 System Improvement

Performance and other service quality attributes are crucial to user satisfaction from services such as databases, networks, etc. Similar qualities are expected from the users of Web services. Web Usage Mining provides the key to understanding Web traffic behavior, which can in turn be used for developing policies for Web caching, network transmission [39], load balancing, or data distribution. Security is an acutely growing concern for Web-based services, especially as e-commerce continues to grow at an exponential rate [50]. Web Usage Mining can also provide patterns which are useful for detecting intrusion, fraud, attempted break-ins, etc.

Almeida et al. [22, 71] propose models for predicting the locality, both temporal as well as spatial, amongst Web pages requested from a particular user or a group of users accessing from the same proxy server. The locality measure can then be used for deciding pre-fetching and caching strategies for the proxy server. However, the increased use of dynamic content has reduced the benefits of caching at both the client and server level. Schechter et. al. [95] have developed algorithms for creating path profiles from data contained in server logs. These profiles are then used to pre-generate dynamic HTML pages based on the current user profile in order to reduce latency due to page generation. Using proxy information for pre-fetching pages has also been studied by [18] and [39].

2.1.3 Site Modification

The attractiveness of a Web site, in terms of both content and structure, is crucial to many applications, e.g. a product catalog for e-commerce. Web Usage Mining provides detailed feedback on user behavior, providing the Web site designer with information on which to base redesign decisions. Web usage data provides an opportunity to turn every site into an ongoing usability test. While the information is not as complete as the information that can be gathered from a formal usability analysis with videos and trained observers, Web usage data are cheap and plentiful. Access times and measures of lostness can be calculated automatically instead of manually as in [65].

While the results of any of the projects could lead to redesigning the structure and content of a site, the adaptive Web site project (SCML algorithm) [81, 82] focuses on automatically changing the structure of a site based on usage patterns discovered from server logs. Clustering of pages is used to determine which pages should be directly linked.

2.1.4 Business Intelligence

Information on how customers are using a Web site is critical for marketers of e-commerce businesses. Buchner et al. [33, 32] have presented a knowledge discovery process in order to discover marketing intelligence from Web data. They define a Web log data hypercube that consolidates Web usage data along with marketing data for e-commerce applications. Four distinct steps are identified in customer relationship life cycle that can be supported by their knowledge discovery techniques : customer attraction, customer retention, cross sales and customer departure. Blue Martini [17] uses clickstream data gathered directly from the content server to discover patterns such as decision tree rules. There are several commercial products, such as SurfAid [13], Accrue [3], NetGenesis [10], Aria [5], Hitlist [8], and WebTrends [14] that provide Web traffic analysis mainly for the purpose of gathering business intelligence. In addition to straight forward usage statistics, Accrue, NetGenesis, and Aria are designed to analyze e-commerce events such as products bought and advertisement click-through rates . Accrue also provides a path analysis visualization tool and IBM's SurfAid provides On-Line Analytical Processing (OLAP) through a data cube and clustering of users in addition to page view statistics. Padmanabhan et. al. [79] use Web server logs to generate beliefs about the access patterns of Web pages at a given Web site. Algorithms for finding interesting rules based on the unexpectedness of the rule are also developed. Han et al. [112] have loaded Web server logs into a data cube structure in order to perform data mining as well as OLAP activities such as roll-up and drill-down of the data. Their WebLogMiner system has been used to discover association rules, perform classification and time-series analysis. e.g. event sequence analysis, transition analysis and trend analysis.

2.1.5 Web Usage Characterization

While most projects that work on characterizing the usage, content, and structure of the Web don't necessarily consider themselves to be engaged in data mining, there is a large amount of overlap between Web characterization research and Web Usage Mining. Pitkow et al. [34, 87, 85] discuss the results of a study conducted at the Georgia Institute of Technology, in which the Web browser Xmosaic was modified to log client-side activity. The results collected provide detailed information about the user's interaction with the browser interface as well as the navigational strategy used to browse a particular site. The project also provides detailed statistics about occurrence of the various client side events such as clicking the back/forward buttons, saving a file, adding to bookmarks etc. The results of subsequent yearly studies are available from http://www.cc.gatech.edu/gvu/user_surveys/. Pitkow et al. [58] also propose a model which can be used to predict the probability distribution for various pages a user might visit on a given site. This model works by assigning a value to all the pages on a site based on various attributes of that page. The formulas and threshold values used in the model are derived from an extensive empirical study carried out on various browsing communities and their browsing patterns. Arlitt et al. [23] discuss various performance metrics for Web servers along with details about the relationship between each of these metrics for different workloads. Manley [69] develops a technique for generating a custom made benchmark for a given site based on its current workload. This benchmark, which he calls a *self-configuring benchmark*, can be used to perform scalability and load balancing studies on a Web server. Chi et. al. [37] describe a system called WEEV (Web Ecology and Evolution Visualization) which is a visualization tool to study the evolving relationship of Web usage, content and site topology with respect to time.

2.2 Discovery of Interesting Patterns

The notion of what makes discovered knowledge interesting has been addressed in [83, 101, 67, 79]. A survey of methods that have been used to characterize the interestingness of discovered patterns is given in [56]. Four dimensions are used by [56] to classify interestingness measures - *pattern-form, representation, scope,* and *class*[1]. Pattern-form defines what type of patterns a measure is applicable to, such as association rules or classification rules. The representation dimension defines the nature of the framework, such as probabilistic or logical. Scope is a binary dimension that indicates whether the measure applies to a single pattern, or to the entire discovered set. The final dimension, class, is also a binary dimension that can be labeled as subjective or objective. *Objective* measures rate rules based on the data used in the mining process. Common *objective* measures include confidence and support [20], or chi-square [31].

Subjective measures are based on the analyst's beliefs or biases. A common theme among the various criteria for subjective interestingness is the concept of *novelty* or *unexpectedness* of a rule. Results that were previously known by the data analyst are not considered interesting. [79] formally defines the unexpectedness of a rule in terms

[1] [56] uses the term *representation* for the first dimension and *foundation* in place of the second dimension. However, in order to be consistent with the large body of existing *reasoning with uncertainty* literature, the terms have been changed in this thesis.

of its deviation from a set of beliefs. [67] has a broader definition of interestingness that includes discovered rules that are not specifically covered by an initial set of beliefs. In other words, a rule that doesn't contradict an existing belief, but points out a relationship that hadn't even been considered is also interesting. While both [67] and [79] give examples of small sets of manually generated beliefs, neither addresses the issue of automated generation of a realistic belief set from a large amount of data. Also, the [67] framework only applies to classification rules.

While the determination of interestingness does not involve inferring new facts from a knowledge base, the comparison and combination of evidence from different sources has been studied extensively under the heading of *Reasoning with Uncertainty* or *Evidential Reasoning.* The critical differences between Reasoning with Uncertainty systems and interestingness frameworks is that Reasoning with Uncertainty must be concerned with the quality of the knew knowledge that is inferred, and the complexity associated with chaining of rules and facts as knew evidence is added to the knowledge base. One of the earliest systems, MYCIN [100], was fairly ad-hoc but successful in inferring medical diagnoses from a list of symptoms. More recent work has been based on Bayesian inference [80], Dempster-Shafer inference [96], or probability-bounds [27]. The various types of inference will be discussed in more detail in Chapter 7 in order to choose a representation for a subjective interestingness framework.

Chapter 3

Data Sources and Data Modeling

A high level view of Web Usage Mining was shown in Figure 1.1 in the Introduction. A more detailed breakdown of the steps involved is shown in Figure 3.1. The various data sources that can be used as input to the Web Usage Mining process are converted into data abstractions such as *server sessions* and *site structure* by the preprocessing tasks. This chapter discusses both the sources and format of the input data, as well as the abstractions that are required to perform any type of meaningful analysis.

3.1 Web Data Sources

One of the key steps in Knowledge Discovery in Databases [51] is to create a suitable target data set for the data mining tasks. As shown in Figure 3.2, data can be collected at the server-level, client-level, proxy-level, or obtained from an organization's database (which can contain business data or consolidated Web data). Each type of data collection differs not only in terms of the location of the data source, but also in the kinds of data available, the segment of population from which the data was collected, and its method of implementation. The usage data collected at the different sources represent the navigation patterns of different segments of the overall Web traffic, ranging from single-user, single-site browsing behavior to multi-user, multi-site access patterns. Client-level logs generally give single-user/multi-site behavior, server-level logs depict multi-user/single-site behavior, and proxy-level logs

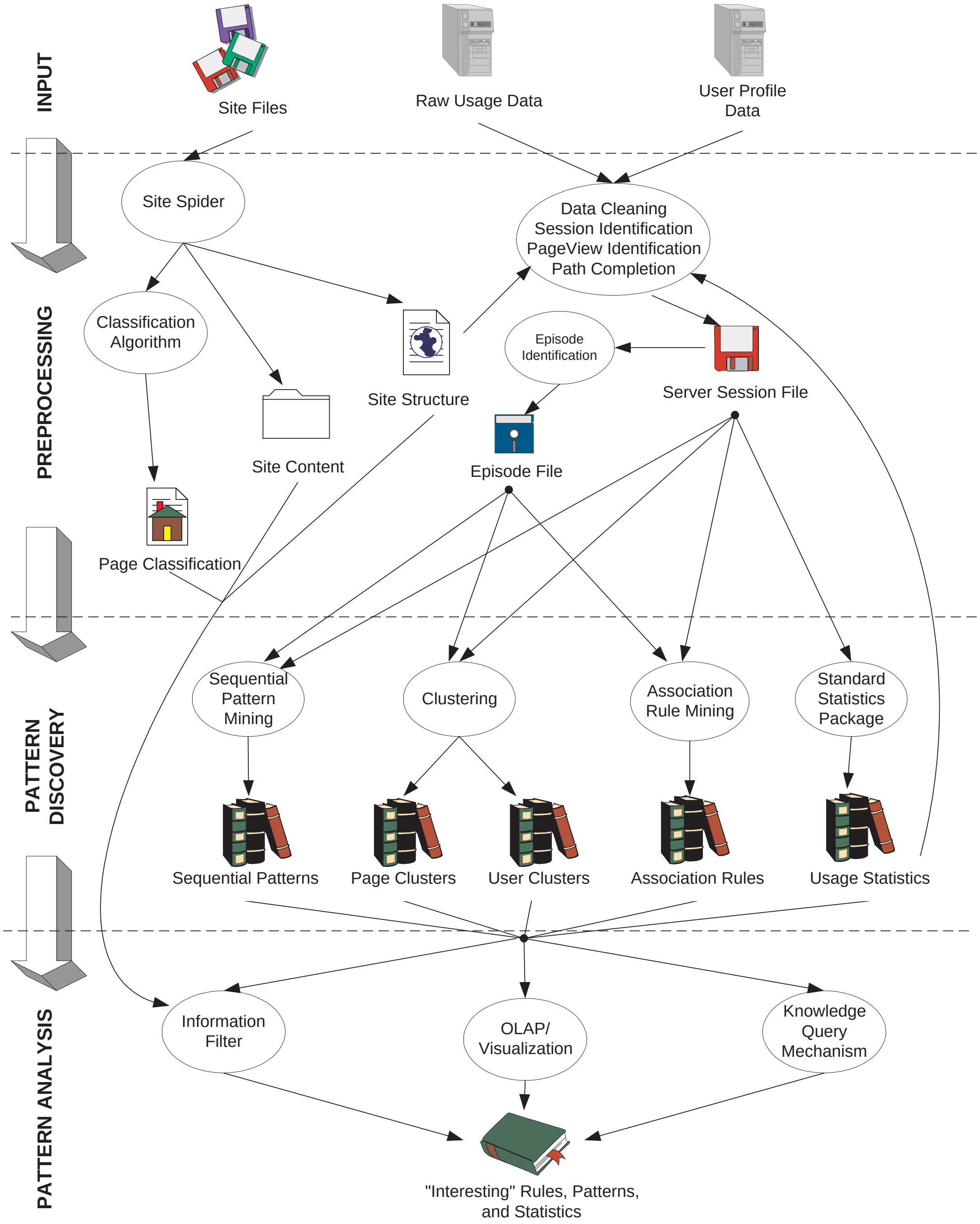


Figure 3.1: Detailed Web Usage Mining Process

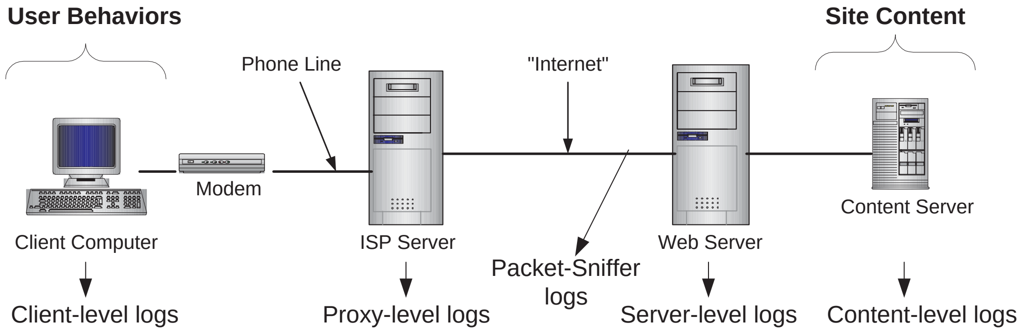


Figure 3.2: Simplified Web Access Diagram

track multi-user/multi-site usage behavior. Figure 3.2 illustrates why reliably tracking user behaviors can be so difficult from data sources such as Web server logs. A Web server log is located in the middle of a communication stream that does not always contain sufficient or accurate information for inferring the behaviors at the client-side as they relate to the pages served from the content server. As an example, Figure 3.3 shows a time-line for three page file requests from a client. For simplicity, only the client-level and server-level events are depicted. A proxy-level or packet-sniffer log would pick up the request for page `A` some time between t_0 and t_1, and the subsequent response from the server between times t_2 and t_3. The content-server level would see the request and it's own response between times t_1 and t_2. As shown in the figure, the user actions at the client-level are not always accurately tracked at the server-level in terms of the number of actions or the times associated with each action. This will be investigated further in the next section.

3.1.1 Server-Level Collection

A Web server log is an important source for performing Web Usage Mining because it explicitly records the browsing behavior of site visitors. The data recorded in server

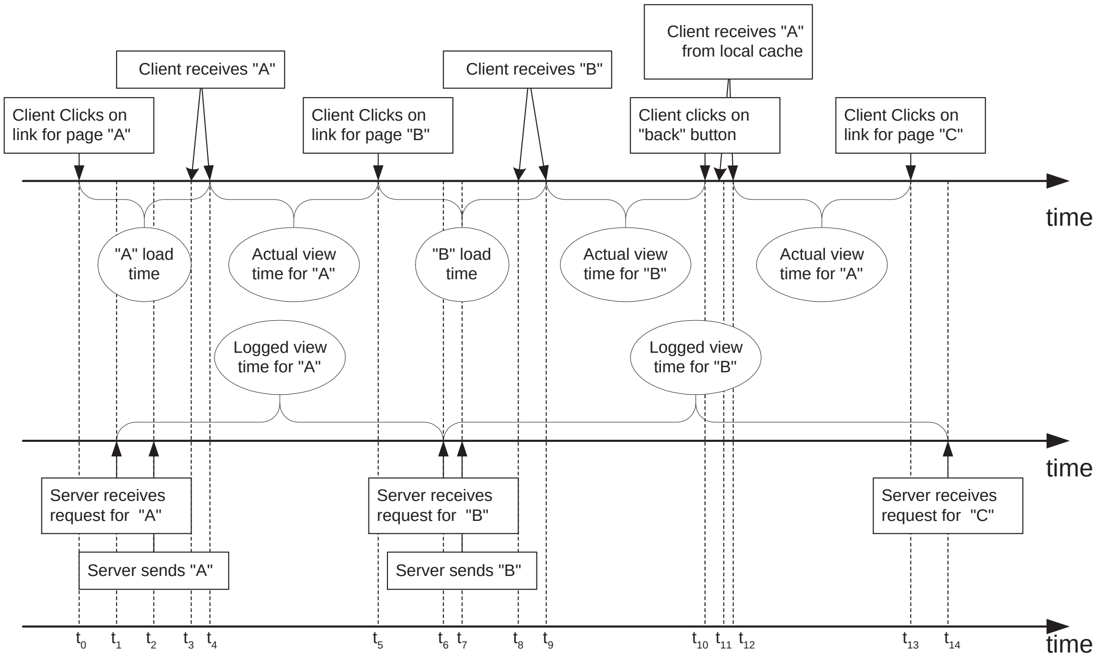

Client receives "A" from local cache
Client receives "A"
Client receives "B"
Client Clicks on link for page "A"
Client Clicks on link for page "B"
Client clicks on "back" button
Client Clicks on link for page "C"
time
"A" load time
Actual view time for "A"
"B" load time
Actual view time for "B"
Actual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

Figure 3.3: Timeline for Page File Request and Response

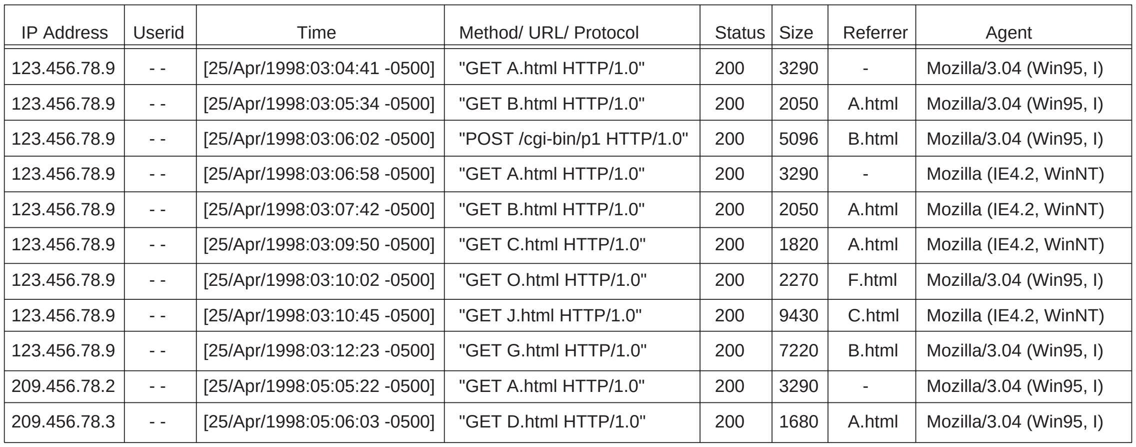
logs reflects the (concurrent and interleaved) access of a Web site by multiple users. These log files can be stored in various formats such as Common Log Format (CLF) or Extended Common Log Format (ECLF) [68]. An example of ECLF is shown in Figure 3.4. The basic data fields are the client IP address, user ID, time/date, request, status, bytes, referrer, and agent. The client IP address is the internet address of the machine that made the request. This may or may not represent the address of a proxy-server. The user ID is filled in only when authentication is required to access the files. The time/date stamp represents the time the request was received by the Web server and the request field lists the method, URI[1], and protocol for the object being requested. The method portion for the request is usually either *GET, POST,*

[1] Uniform Resource Identifier (URI) is a more general definition that includes the commonly referred to Uniform Resource Locator (URL).

or *HEAD.* GET requests an object from the Web server, POST sends information to the WEB server, and HEAD requests just the HTTP header for an object. The URI is either a static file in the local file system, or the name of an executable program that will be called in response to the request. The status field is set by the Web server and indicates the action taken in response to a request. Codes from 200 through 299 generally indicate success, 300 through 399 indicate some form of redirection, 400-499 indicate an error serving the particular request, and 500-599 indicate a problem with the Web server. A common error code is 404, which indicates the requested file was not found. The size field logs the number of bytes returned as a result of a request. The referrer field can contain the URI of the source of the request. For URIs that are typed in or access through a bookmark, the referrer field will be null (As shown in lines 1,4, and 10 of Figure 3.4). Finally, the agent field is a text string that can indicate the operating system and browser software used by the client.

Due to caching and network transmission times, the information in the Server log may not be entirely reliable. As shown in Figure 3.3, the second request for page `A` by the client is served by the local cache, and is not recorded by the server log. Even when caching is not present, the view time for a page as recorded in a server log is often longer than the actual view time on the client-side. Again, as shown in Figure 3.3, the time between requests for pages `A` and `B` is $t_6 - t_1$, but the actual view time for page `A` is only $t_5 - t_4$. Depending on the connection speed of the client, the size of the page file, and network congestion, the difference in recorded and actual view time $(t_4 + t_6 - t_1 - t_5)$ may range from a single second to several minutes.

Tracking of individual users is not an easy task due to the stateless connection model of the HTTP protocol. In order to handle this problem, Web servers can also store other kinds of usage information such as cookies in separate logs, or appended to the CLF or ECLF logs. Cookies are tokens generated by the Web server for



IP Address	Userid	Time	Method/ URL/ Protocol	Status	Size	Referrer	Agent
123.456.78.9	- -	[25/Apr/1998:03:04:41 -0500]	"GET A.html HTTP/1.0"	200	3290	-	Mozilla/3.04 (Win95, I)
123.456.78.9	- -	[25/Apr/1998:03:05:34 -0500]	"GET B.html HTTP/1.0"	200	2050	A.html	Mozilla/3.04 (Win95, I)
123.456.78.9	- -	[25/Apr/1998:03:06:02 -0500]	"POST /cgi-bin/p1 HTTP/1.0"	200	5096	B.html	Mozilla/3.04 (Win95, I)
123.456.78.9	- -	[25/Apr/1998:03:06:58 -0500]	"GET A.html HTTP/1.0"	200	3290	-	Mozilla (IE4.2, WinNT)
123.456.78.9	- -	[25/Apr/1998:03:07:42 -0500]	"GET B.html HTTP/1.0"	200	2050	A.html	Mozilla (IE4.2, WinNT)
123.456.78.9	- -	[25/Apr/1998:03:09:50 -0500]	"GET C.html HTTP/1.0"	200	1820	A.html	Mozilla (IE4.2, WinNT)
123.456.78.9	- -	[25/Apr/1998:03:10:02 -0500]	"GET O.html HTTP/1.0"	200	2270	F.html	Mozilla/3.04 (Win95, I)
123.456.78.9	- -	[25/Apr/1998:03:10:45 -0500]	"GET J.html HTTP/1.0"	200	9430	C.html	Mozilla (IE4.2, WinNT)
123.456.78.9	- -	[25/Apr/1998:03:12:23 -0500]	"GET G.html HTTP/1.0"	200	7220	B.html	Mozilla/3.04 (Win95, I)
209.456.78.2	- -	[25/Apr/1998:05:05:22 -0500]	"GET A.html HTTP/1.0"	200	3290	-	Mozilla/3.04 (Win95, I)
209.456.78.3	- -	[25/Apr/1998:05:06:03 -0500]	"GET D.html HTTP/1.0"	200	1680	A.html	Mozilla/3.04 (Win95, I)

Figure 3.4: Sample ECLF Log

individual client browsers in order to automatically track the site visitors. Cookies rely on implicit user cooperation and thus have raised growing concerns regarding user privacy. Query data, which is typically generated by online visitors while searching for pages relevant to their information needs, can also be logged on the server-side.

The Web server also relies on other utilities such as CGI (Common Gateway Interface) scripts to handle data sent back from client browsers. Web servers implementing the CGI standard parse the URI of the requested file to determine if it is an application program. The URI for CGI programs may contain additional parameter values to be passed to the CGI application. Once the CGI program has completed its execution, the Web server sends the output of the CGI application back to the browser. If the request is made through the hidden POST method, the parameters will not be available in the CLF or ECLF logs. This is shown in the third line of Figure 3.4. The request calls the executable program "/cgi-bin/p1", but any CGI variables that accompanied the request are passed directly to the program and are not logged.

Packet sniffing technology (also referred to as "network monitors")is an alternative method to collecting usage data through server logs. Packet sniffers monitor network traffic coming to a Web server and extract usage data directly from TCP/IP packets (shown in Figure 3.2). If a packet sniffer is placed directly outside a Web server, the segement of usage information available to the packet-sniffer is identical to the information availabel to the Web server. While in practice commercial packet sniffers [3, 5] tend to log hidden CGI variables, whereas CLF and ECLF logs do not, there is no technical restriction that forces this difference. Packet sniffers also are capable of logging information based on the actual content of the data packets in addition to the HTTP headers. However, scalability quickly becomes an issue for packet sniffers monitoring high volume Web sites. Packet sniffers can also be placed "farther out" in a network away from a single server in order to collect usage data for multiple Web servers.

A third server side usage data source is any application that is used to serve dynamic content, commonly referred to as a *content server* or application server. Most commercial content servers provide some sort of logging capabilities for usage data. The main advantage of logging usage data directly from the content server is that data is being collected from the source of the Web site content, as show in Figure 3.2. All of the information required to determine what content is served, such as hidden POST parameters or internal state variables, are available.

Besides usage data, the server side also provides access to the "site files", e.g. content data, structure information, local databases, and Web page meta-information such as the size of a file and its last modified time.

3.1.2 Client Level Collection

Client-side data collection can be implemented by using a remote agent (such as Javascripts or Java applets) or by modifying the source code of an existing browser (such as Mosaic or Mozilla) to enhance its data collection capabilities. The implementation of client-side data collection methods requires user cooperation, either in enabling the functionality of the Javascripts and Java applets, or to voluntarily use the modified browser. As mentioned, client-side collection has an advantage over server-side collection because it is located at the source of the user behaviors, providing relief from caching and session identification problems. However, Java applets perform no better than server logs in terms of determining the actual view time of a page. In fact, it may incur some additional overhead especially when the Java applet is loaded for the first time. Javascripts, on the other hand, consume little interpretation time but cannot capture all user clicks (such as reload or back buttons). These methods will collect only single-user, single-site browsing behavior. A modified browser is much more versatile and will allow data collection about a single user over multiple Web sites. In order to use this method, the users must be convinced to use the modified browser for their daily browsing activities. This can be done by offering incentives to users who are willing to use the browser, similar to the incentive programs offered by companies such as NetZero [12], AllAdvantage [4], and Alexa Internet [16].

3.1.3 Proxy Level Collection

As shown in Figure 3.2, the Internet Service Provider (ISP) machine that users connect to through a modem is a common form of a Proxy server. A Web proxy acts as an intermediary between client browsers and Web servers. Proxy-level caching can be used to reduce the loading time of a Web page experienced by users as well as the network traffic load at the server and client sides [39]. The performance of proxy caches depends on their ability to predict future page requests correctly. Proxy traces may reveal the actual HTTP requests from multiple clients to multiple Web servers, which can serve as a data source for characterizing the browsing behavior of a group of anonymous users sharing a common proxy server. The effect of proxy-level caching on server-side data collection is heavily dependent on the nature of the Web site being accessed. Sites that use predominantly dynamic content, such as e-commerce sites, will not be effected by proxy-level caching since each page served is technically a new file (even if the content is the same). However, Web sites with static content, such as digital libraries, can be very susceptible to proxy-level caching.

3.2 Data Modeling

As mentioned briefly in the Introduction, the W3C has defined several data abstractions for Web usage. These turn out to be the appropriate data model for performing any type of usage analysis, whether it is static aggregation, OLAP, or data mining.

A *user* is defined as a single individual that is accessing files from one or more Web servers through a browser. While this definition seems trivial, in practice it can be very difficult to uniquely and repeatedly identify users. Even with the use of cookies, a user may access the Web through different machines, or use more than one browser on a single machine. Also, several users may use the same machine and browser.

A *page view* consists of the set of files that contribute to the display on a user's browser at one time. Page views are usually associated with a single user action (such as a mouse-click) and can consist of several files such as frames, graphics, and scripts. When discussing and analyzing user behavior, it is really the aggregate page view that is of importance. The user does not explicitly ask for "n" frames and "m" graphics to be loaded into his or her browser, the user requests a "Web page." All information to determine which files constitute a page view is accessible from the site content. This thesis refers to a file that is served to a user as a *page file*. Due to the nature of dynamic content, a page file may exist as several separate sources (e.g.

database table, javascript) until a content server compiles them in response to a user request.

A *click-stream* is a sequential series of page view requests. Again, the data available from the server side does not always provide enough information to reconstruct the full click-stream for a site. Any page view accessed through a client or proxy-level cache will not be "visible" from the server side. A *user session* is the click-stream of page views for a single user across the entire Web. Typically, unless client-side data collection is available, only the portion of each user session that is accessing a specific site can be used for analysis, since access information is not publicly available from the vast majority of Web servers. The set of page views in a user session for a particular Web site is referred to as a *server session* (also commonly referred to as a *visit*). A set of server sessions is the necessary input for any Web Usage analysis or data mining tool. The end of a server session is defined as the point when the user's browsing session at that site has ended. This is concept is very difficult to track reliably with server-side data since the "next-click" at a different Web site is not logged. Also, there is no way of knowing when a user exits a browser when tracking from the proxy or server levels. Any semantically meaningful subset of a user or server session is referred to as an *episode* by the W3C WCA. The definitions and abstractions that are frequently used throughout this thesis are summarized in Table 3.1. The next two chapters will focus on converting the available data sources into the data model necessary for performing Web Usage Mining.

Table 3.1: Frequently Used Definitions and Abstractions.

Term	Definition
User	Single individual that is accessing files from one or more Web servers through a Browser.
Page File	File that is served through HTTP protocol to a User.
Page View	Set of Page Files that contribute to a single display in a Web Browser.
Browser	Client-side software that is responsible for displaying Page Views and making HTTP requests to a Web Server.
Web Server	Server-side software that is responsible for handling incoming HTTP requests.
Content Server	Server-side software that is responsible for serving Page Files in response to requests.
Server Session	Set of page views served due to a series of HTTP requests from a single User to a single Web Server.
Episode	Subset of page views from a single User or Server Session.

Chapter 4

Structure and Content Preprocessing

Preprocessing the structure and content of a Web site are inter-related tasks. The answer to the question of what links are available from a given page view depends on how the page view is defined. The degree of difficulty in performing preprocessing is highly dependent on the technology used to create the Web site content. Conceptually, the problem is simple - create a map or graph of the Web site. However, for dynamically generated Web sites, especially ones utilizing personalization technologies, the question of equality for served content is not trivial. The design or content of the "home page" for a Web site may be different depending on the user. Should different versions of the home page be considered to be different pieces of content for analysis purposes? A simple text matching algorithm for the served files may result in an explosion in the number of "unique" pieces of content in a site map. It may even result in a forest of site maps, instead of a single graph.

There are two major tasks associated with content and structure preprocessing - determining what constitutes a unique page file, and determining how to represent the content and structure of the page file in a quantifiable form. The first task relates to the discussion in the paragraph above. Once the first task is solved, the structure and content of a Web site need to be encoded in a way that is useful for performing the various steps of Web Usage Mining. The task of defining a unique page file in a way that is semantically meaningful is highly dependent on the Web site. For the purposes of this thesis, it will be assumed that this step has already been completed for the Web site being analyzed. In practice, defining the site content is a manual process that not only depends on the technology used to create the Web site, but also the goals of the analysis. The rest of this chapter is concerned with the second task of quantifying structure and content in ways that are suitable for supporting Web Usage Mining.

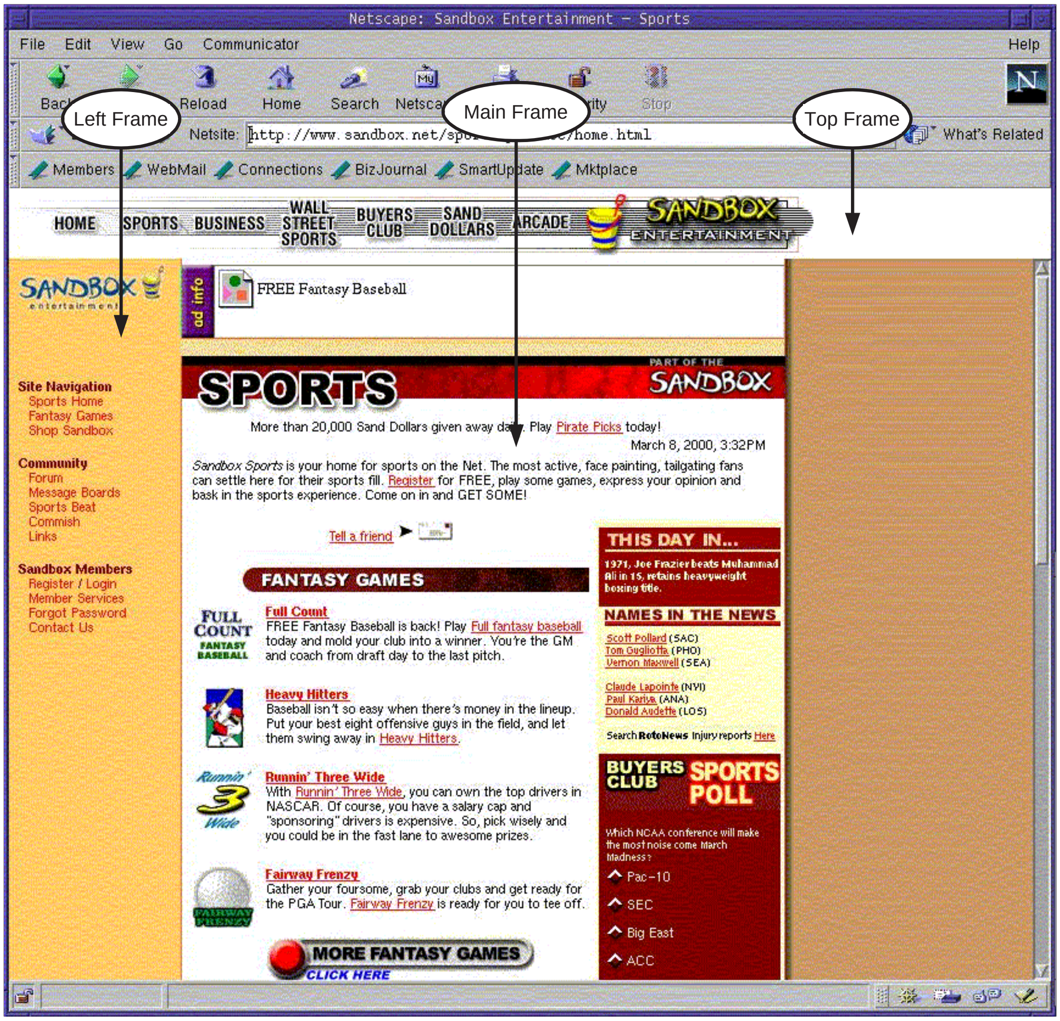
4.1 Structure Preprocessing

The structure of a site is created by the hypertext links between page views and the frame and image tags that populate a particular page view (referred as intra-page structure). Several usage preprocessing steps can not be completed without the site structure. In addition, the site structure is useful for identifying potentially interesting rules. As will be described in Chapter 5, the Web site structure is required for page view identification, and may be needed to identify users in the absence of a unique user identifier such as cookies. Due to the presence of *frames*, the number of potential page views for a Web site can be vast. It is not uncommon for every page view on a site to consist of two or three frames. An example of a common page view structure as displayed in a browser is shown in Figure 4.1. There is a top frame for general site navigation , a left frame for more specific navigation, and a main frame with some content. Assuming that there are x top frames, y left frames, and z main frames for a site, the number of unique page views could be as high as $x * y * z$. This number can quickly become intractable. Therefore, the structure of a Web site needs to be stored as a set of frames, $\mathcal{F}$, with a list of associated links and targets. A target is the page area that the link should be loaded into in the browser display. To further complicate the situation, a single link can lead to the replacement of between one and all of the frames in a page view. A formal definition of a site structure map for use in Web Usage Mining is as follows, where $\mathcal{M}$ is a site map, h_i is an HTML file,

r is a link type, and g_i is a target area:

$$\begin{aligned} \mathcal{M} &= [< \mathcal{F}_1; \ldots; \mathcal{F}_n >] && (4.1) \\ \mathcal{F} &= \{h_f, \mathcal{L}_1, \ldots, \mathcal{L}_m\} && (4.2) \\ \mathcal{L} &= < r, (h_1, g_1) | \ldots | (h_p, g_p) > && (4.3) \end{aligned}$$

The link type indicates how the page file will be requested from the Web server. The most common is *get*, which means the standard GET method will be used to request the page file. Other common types include *post*, *hidden post*, *frame*, *ftp*, and *mail*. The post and hidden post types both use the POST HTTP method for sending data back to the Web server. Although the POST method is theoretically a one way transfer of data from the client to the server, in practice, an application such as a content server will send a page file to the client as a response to the data sent in the POST. The difference between a regular post and a hidden post is in how the client data are sent back to the Web server. A regular post appends the data to the URI in CGI format (a series of text name/value pairs delimited by "&"). A hidden post passes the data in the HTTP header. This is an important distinction since, as discussed in Chapter 3, the URI is logged as part of the CLF or ECLF formats, but hidden POST data are not. If the source of the usage data contains the hidden POST parameters, such as a packet sniffer log, then there is no distinction between the *post* and *hidden post* types. The frame type refers to the use of the HTML "frame" tag. In this case, page files listed as frames are automatically requested from the Web server. The ftp and mail link types are just two common types that do not lead to HTML page files being served. An example of a very simple Web site is given in Figure 4.2. The site is essentially a tree structure, with each page view consisting of one or two frames. The meaning of the three different *usage types* is explained in



More than 20,000 Sand Dollars given away da y. Play Pirate Picks today!

March 8, 2000, 3:32PM

Sandbox Sports is your home for sports on the Net. The most active, face painting, tailgating fans can settle here for their sports fill. Register for FREE, play some games, express your opinion and bask in the sports experience. Come on in and GET SOME!

1971, Joe Frazier beats Muhammad Ali in 15, retains heavyweight boxing title.

Full Count
FREE Fantasy Baseball is back! Play Full fantasy baseball today and mold your club into a winner. You're the GM and coach from draft day to the last pitch.

Heavy Hitters
Baseball isn't so easy when there's money in the lineup. Put your best eight offensive guys in the field, and let them swing away in Heavy Hitters.

Runnin' Three Wide
With Runnin' Three Wide, you can own the top drivers in NASCAR. Of course, you have a salary cap and "sponsoring" drivers is expensive. So, pick wisely and you could be in the fast lane to awesome prizes.

Fairway Frenzy
Gather your foursome, grab your clubs and get ready for the PGA Tour. Fairway Frenzy is ready for you to tee off.

Which NCAA conference will make the most noise come March Madness?

Figure 4.1: Sample Page View

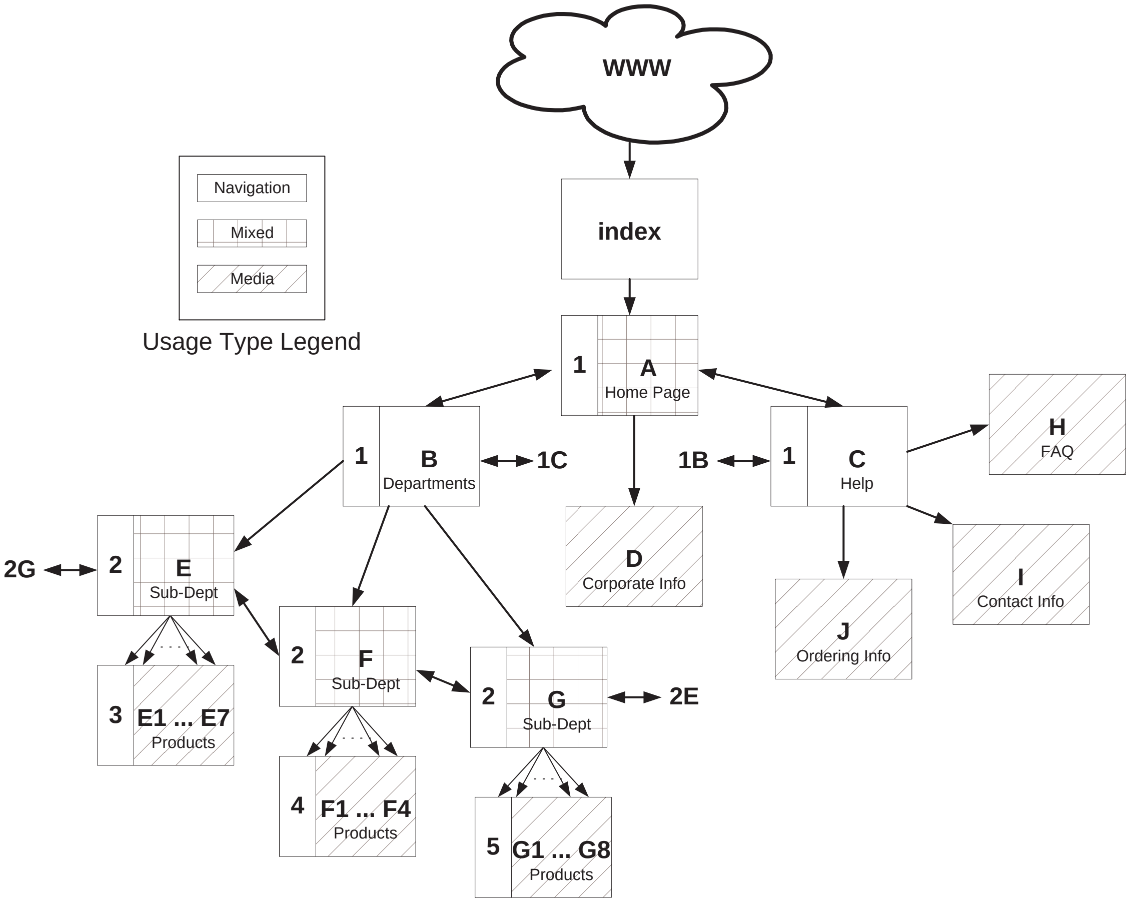


Figure 4.2: Example Web Site

Section 4.2. The site map for Figure 4.2 is as follows, where the possible frame areas are *left, main,* and *top*:

$$\begin{aligned}\mathcal{M} \;=\; & [\{index, (frame, 1, left|frame, A, main)\}; \\ & \{1, (get, A, main), (get, B, main), (get, C, main)\}; \\ & \{2, (get, E, main), (get, F, main), (get, G, main)\}; \\ & \{3, (get, E1, main), \ldots, (get, E7, main)\}; \\ & \{4, (get, F1, main), \ldots, (get, F4, main)\}; \\ & \{5, (get, G1, main), \ldots, (get, G8, main)\}; \\ & \{A, (get, D, top)\}; \\ & \{B, (get, 2, left|E, main), (get, 2, left|F, main), (get, 2, left|G, main)\}; \\ & \{C, (get, H, top), (get, I, top), (get, J, top)\}; \\ & \{D\}; \\ & \{E, (get, 3, left|E1, main), \ldots, (get, 3, left|E7, main)\}; \\ & \{F, (get, 4, left|F1, main), \ldots, (get, 4, left|F4, main)\}; \\ & \{G, (get, 5, left|G1, main), \ldots, (get, 5, left|G8, main)\}; \\ & \{E1\}; \ldots; \{E7\}; \{F1\}; \ldots; \{F4\}; \{G1\}; \ldots; \{G8\}; \\ & \{H\}; \{I\}; \{J\}] \end{aligned} \tag{4.4}$$

The site map lends itself very naturally to an object-oriented schema, where a site object contains a list of frame objects, and each frame object is made up of a file, plus a list of link objects. In order to form page views from the site map, an initial page view needs to be specified. In the case of the example site, the natural initial view is the home page, `1-A`, which will be automatically served as a result of a request for `index.html`[1]. All other page views for the site can be formed by replacing one or more frame areas with any of the candidate links in the active frames. For example, page view `1-B` is formed by following the link to `B` from frame `1`. Page view `2-F` is formed from a single link in `B`, that replaces both frame `1` with `2` and

[1] By default, most Web servers fulfill a request for a directory with a file named index.html or home.html. e.g. A request for www.sample.com will get index.html from the base directory for the Web site.

frame B with F. With 5 left frames and 25 main frames, a maximum of 125 page views could be theoretically formed. However, since in this simple example each main frame is only accessible from one left frame, there are only 25 page views resulting from combinations of left and main frames. With the four single frame page views, this site has 29 possible page views.

4.2 Content Preprocessing

Content preprocessing consists of converting the text, image, scripts, and other multimedia files into forms that are useful for the Web Usage Mining process. Often, this includes performing content mining such as classification or clustering. While applying data mining to the content of Web sites is an interesting area of research in its own right, in the context of Web Usage Mining the content of a site can be used to filter the input to, or output from the pattern discovery algorithms. For example, results of a classification algorithm can be used to limit the discovered patterns to those containing page views about a certain subject or class of products. In addition to classifying or clustering page views based on topics, page views can also be classified according to their intended use [84, 43]. Page views can be intended to convey information (through text, graphics, or other multimedia), gather information from the user, allow navigation (through a list of hypertext links), or some combination uses. The intended use of a page view can also filter the sessions before or after pattern discovery. The content for a site is obtained at the same time the site structure is being mapped. As each HTML file is being fetched and parsed to construct the site map, any text, graphics, or other media types can be cataloged as well.

Table 4.1: Expected Characteristics of Web Page Types

Page Type	Expected Physical Characteristics	Expected Usage Characteristics
Head	• In-links from most site pages • Root of site file structure	• First page in user sessions
Media	• Large text/graphic to link ratio	• Long average reference length
Navigation	• Small text/graphic to link ratio	• Short average reference length • Not a maximal forward reference
Look-up	• Large number of in-links • Few or no out-links • Very little content	• Short average reference length • Maximal forward reference
Data Entry	• "FORM" tag is present	• Followed by a POST request

4.2.1 Content Usage Type

As first detailed in [84], page views are generally intended to be used in different ways. An adaptation of the page usage types defined in [84] is as follows:

- Head Page—a page whose purpose is to be the first page that users visit, i.e. "home" pages.
- Media Page—a page that contains a portion of the information content that the Web site is providing.[2]
- Navigation Page—a page whose purpose is to provide links to guide users on to content pages.
- Look-up Page—a page used to provide a definition or acronym expansion.
- Data Entry Page—a page used to explicitly gather information from the user.

Each of these types of pages is expected to exhibit certain physical characteristics. For example, a head page is expected to have in-links from most of the other pages in

[2] Both [84] and [43] refer to this page type as a *content* page. However, to avoid confusion with the discussions about Web site *content*, this thesis will refer to these pages as *media* pages. This is because the information content will be represented by some form of media such as text or graphics.

the site, and is often at the root of the site file structure. Table 4.1 lists some common physical characteristics for each page type. Note that these are only rules-of-thumb, and that there will be pages of a certain type that do not match the common physical characteristics. There are several valid combinations of page types that can be applied to a single page, such as a head-navigation page or a media-navigation page.

The page classifications should represent the Web site designer's view of how each page will be used. The classifications can be assigned manually by the site designer, or automatically by using supervised learning techniques. In order to automate the classification of site pages, the common physical characteristics can be learned by a classification algorithm such as C4.5 [89] using a training set of pages. Several experiments have been performed by Mitchell et. al. [46, 47] using supervised learning techniques to assign page files to a predefined ontology. Another possibility is that a classification tag can be added to each page by the site designer, using a data structure markup language such as XML (Extensible Markup Language) [30].

4.2.2 Text Preprocessing

While the hypertext link structure of a Web site naturally lends itself to a directed graph, quantifying the content in the page files is not as straight forward. The two techniques that seem to be the most useful for supporting Web Usage Mining are clustering and classification of the page files based on the text in the file. While it would be interesting to take into account the graphics and other multimedia files, research into automated multimedia clustering and classification is still relatively new, and very resource intensive. In order to make use of any data mining algorithm, the text in the page files must first be preprocessed. Once the text is preprocessed, any number of clustering or classification algorithms can be run, such as the Hypergraph clustering method [38] or Support Vector Machine classification method [40] that will be described in Chapter 6.

Unless XML tags are used to add semantic structure to a Web site, the problem of quantifying page files is essentially the same as the quantification of unstructured text documents. While some experiments have attempted to make use of HTML display tags to infer semantic meaning, these methods do not work in the general case. *Wrapper* projects, such as [92] have had some success in using regularities in the HTML tag structure of Web sites, but methods such as these are extremely brittle, and any deviations from the expected tag structure corrupts the algorithm.

Virtually all data mining systems that use the full text of documents rely on some version of Salton's vector space model [93]. The vector space model works by defining each unique word in the corpus of documents as a separate dimension. Each document can then be described by creating a vector where every word that is in the document is represented by a weight. For words that do not appear in a given document, that particular dimension of the vector is set to zero. The two major decisions that must be made when using a vector space model for data mining are, how to reduce the feature space (if any reduction is required) and how to weight the terms.

Feature Space Reduction

Feature space reduction is often performed to improve the performance of data mining or learning algorithms. It can be important for both improving the accuracy of the algorithm and controlling the computation time. Feature space reduction can also be desirable for other reasons such as limits on storage space or providing on-line processing capabilities.

Although not usually referred to as feature space reduction techniques, the use of stopwords and stemming algorithms do have the effect of limiting the number of unique terms in a vector space. Stopwords are defined as words that are known to be generic and have no use in distinguishing documents. Articles and prepositions are frequently included in stopword lists. In addition to removing stopwords, suffixes are removed in order to prevent counting plural or active forms of words separately from the normal form. This is referred to as *stemming.* For instance, the words "running" and "runs" should both be converted to "run" to avoid unnecessary feature space expansion. For text classification, Yang and Pedersen [110], have shown that feature space reduction techniques that favor common terms outperform those that favor rare terms. Information gain (IG) [73] is a term weight that meets this criteria and is calculated using the conditional probability of a document being in a given category, depending on the presence or absence of a term in the document. The information gain, IG of a term, t is calculated as follows, where the categories are c_i's, with $1 \leq i \leq m$:

$$\begin{aligned} IG(t) &= -\sum_{i=1}^{m} p(c_i) log(p(c_i)) \\ &+ p(t)\sum_{i=1}^{m} p(c_i|t) log(p(c_i|t)) \\ &+ p(t)\sum_{i=1}^{m} p(c_i|\neg t) log(p(c_i|\neg t)) \end{aligned} \tag{4.5}$$

Recently, work from the Artificial Intelligence and Natural Language communities has been employed to attempt some "intelligent" feature space reduction. Natural Language techniques, in particular part of speech identification, can be used to pick out the "important" words from a document. For example, nouns, proper nouns, or verbs can be selected from each document, ignoring all other parts of speech. The subset of proper nouns consisting of people, locations, and organizations is referred to as *named entities.* Since named entities are not biased to commonly appearing words, feature space reduction through identification of named entities can result in a similar reduction in the overall size of the data set.

Term Weighting

One of the most common weighting schemes used is referred to as term-frequency/inverse-document-frequency (TFIDF) [94]. This weighting scheme starts with the frequency of a term in a given document (TF), and multiplies this by the "inverse document frequency" (IDF) of the term in the corpus. The IDF of a term is lower the more documents it appears in. The TFIDF formula for a term, t_i is as follows, where n is the number of documents in the corpus and DF is the document frequency for the term:

$$TFIDF(t_i) = TF(t_i) * \log\left(\frac{n}{DF(t_i)}\right) \tag{4.6}$$

The idea is that the more documents a word appears in, the less likely it is to be a good measure for distinguishing one document from another. If the added "information" contained in the IDF is not needed for a particular algorithm, just the term frequency (TF) can be used for a weighting scheme. The advantage of using TF is that it can be calculated from just a single document, without having to maintain and update corpus wide document frequencies. An even simpler term weighting scheme that also has this advantage is binary weighting, where terms that are present are assigned a weight of one, and those that are missing are assigned a weight of zero.

Clustering and Classification

Three properties that are desirable in a clustering algorithm selected for a domain such as unstructured text are relative immunity to a high number of dimensions, no requirement to pre-specify the number of discovered clusters, and a simple method for evaluating the quality of the discovered clusters. One clustering method that meets the first two criteria is hypergraph partitioning of association rules [55]. The third property is difficult to meet quantitatively, because unlike classification there is no predefined "correct value" to use to calculate measures such as *precision* or *recall*. The method employed by the TopCAT (Top Clusters and Associations from Text) system [38] is to use named entity identification as a form of feature space reduction, which also leaves a data set that is very rich in semantic meaning. Clusters of named entities are assumed to be qualitatively easier to understand than a list of stemmed terms, allowing clusters to be evaluated by hand. Experimental results from the use of the TopCAT system on news stories are presented in [38, 40].

For text classification, the number of categories is pre-determined and the quality of the classification can be quantitatively verified. Therefore, the remaining property of the three listed for text clustering, immunity to a high number of dimensions, becomes critical for successful text classification. Support Vector Machines are ideally suited for the problem of text classification since the complexity of the model learned can be independent of the dimensionality of the feature space. Results presented in [60] and [40] confirm this for several set of text documents. The details of the use of Support Vector Machines for classification and hypergraph partitioning for clustering will be presented in Chapter 6.

Chapter 5

Usage Preprocessing

The goal of usage preprocessing is to end up with a set of minable objects for a particular Web site (or set of sites). The most common form of input is a Web server log in the CLF or ECLF format, as described in Chapter 3. However, usage data can also come from HTTP packet sniffers or application logs. Regardless of the source of the usage data, the preprocessing steps shown in Figure 5.1 must be completed at some point in order to create server sessions[1]. Since user episodes are defined as a general subset of server sessions, there is a vast array of techniques that can be used to identify episodes from sessions. This chapter also presents a few methods for episode identification.

5.1 Server Session Identification

As shown in Figure 5.1, there are three major required steps and one optional step to convert raw usage data into server sessions - data cleaning, user/session identification, page view identification, and path completion. The optional steps are indicated by dashed lines in Figure 5.1. This is very similar to the *Clickstream Post-Processor* defined in [64], which separates the user and session identification into two separate steps, and stores the information necessary for page view identification in a module

[1] Depending on the technology used to create the Web site, the difficulty of each step may vary greatly. This chapter describes the transformation of raw usage data into server sessions for commonly encountered scenarios.

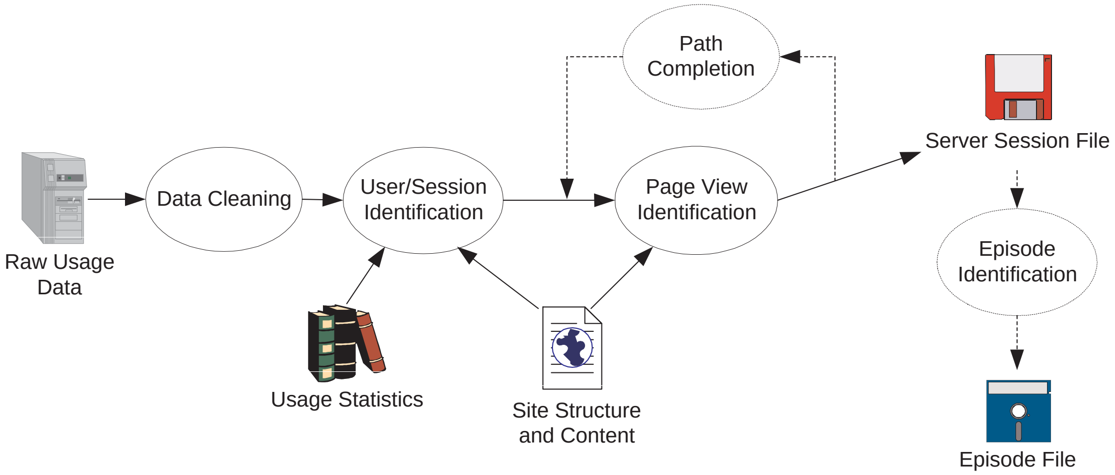


Figure 5.1: Details of Usage Preprocessing

referred to as the *Code Maintenance Application*. However, the [64] architecture does not deal with path completion or episode identification. Formally, a server session ($\mathcal{S}$) is a time ordered set of page views ($\mathcal{V}$) for a single user during a single visit to a Web site, along with some meta data ($\mathcal{A}$).

$$\mathcal{S} = [\mathcal{A} : \mathcal{V}_1, \ldots, \mathcal{V}_n] \tag{5.1}$$

$$\mathcal{V} = \langle v_i, h_j, t^f, t^l, t^e, \{d_1, \ldots, d_m\}, c\rangle \tag{5.2}$$

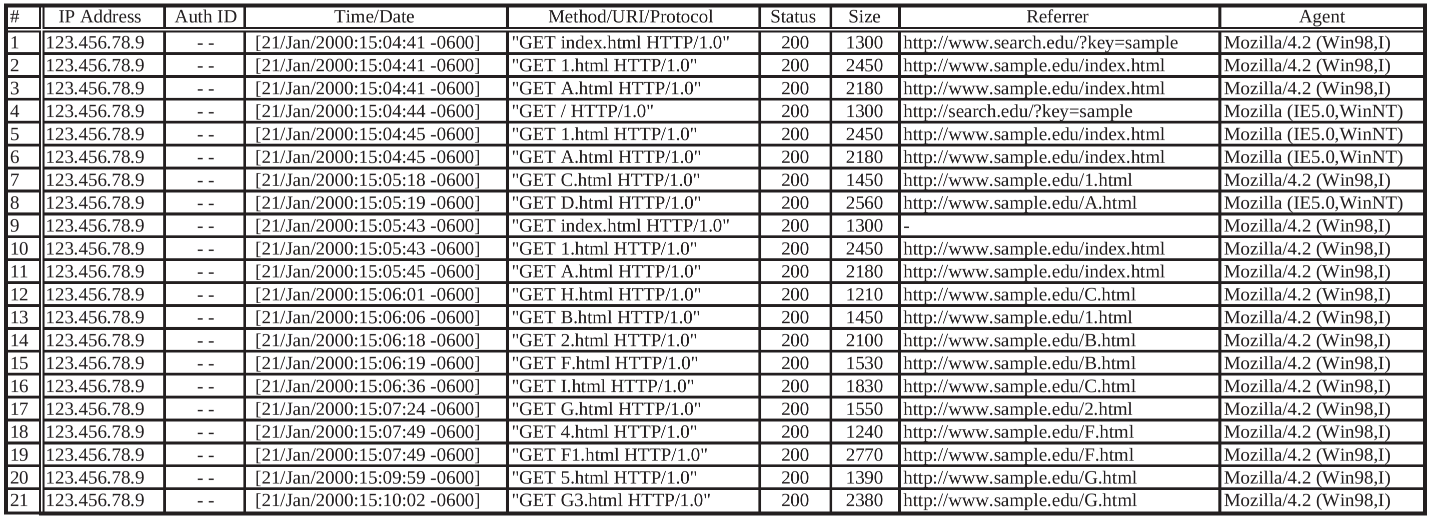
$$\mathcal{A} = \{a_1, \ldots, a_k\} \tag{5.3}$$

Each page view $\mathcal{V}$ consists of a view identifier (v_i), referring page file (h_j), first request time (t^f), last request time (t^l), view end time (t^e), and an optional set of submitted values ($d_1, \ldots, d_m$). The final parameter, c, is a boolean parameter that indicates whether the page view was inferred during path completion. For any page view directly derived from usage data, c will be false. The view identifier maps to the set of page files that make up a particular page view for the Web site. The number and type of the page files is determined by the design of the Web site. The referring page file is the file from which the page view request originated. In cases where the user typed in a URI, selected a bookmark, chose from a history list, or has certain privacy features activated, the referring page file will be null. The first and last times correspond to the time of the first and last file requests for a given page view. The first time field is used to order the page views in a session. Often, for single file page views or fast connections, these times will be the same. The view end time is usually inferred from the first file request time for the *next* page view. However, some client-level collection methods can log the actual page view time, as discussed in Chapter 3. If there is no *next* page view (e.g. the end of a session), the end time will be null. The optional set of submitted values is used for page views that have *form data*. Since not every page view allows data entry, this field can also be null. The meta data, $\mathcal{A}$, collected for each session depends on the technologies being used. For ECLF logs, the session meta data will consist of an IP address (or possibly a set of IP addresses) and an agent (usually the client-side browsing software and operating system). Cookies, embedded session IDs, and other add-on technologies can add meta data to a session.

A sample Web log (ECLF with line numbers added for clarity) is shown in Figure 5.2. It is based on the sample Web site shown in Figure 4.2. Figure 5.2 will be used throughout the remainder of this chapter as a running example.

5.1.1 Data Cleaning

Data cleaning is a site specific step that involves mundane tasks such as merging logs from multiple servers and parsing the log into data fields. Typically, graphics file requests are stripped out at this stage. This is easily done by checking for filename suffixes such as “gif” or “jpg.” Graphics files can be left in the data set and rolled up into page views in a later preprocessing step without any loss of generality. Situations where this is desirable is for checking for *agent* or *spider* traffic through the site. The



#	IP Address	Auth ID	Time/Date	Method/URI/Protocol	Status	Size	Referrer	Agent
1	123.456.78.9	- -	[21/Jan/2000:15:04:41 -0600]	"GET index.html HTTP/1.0"	200	1300	http://www.search.edu/?key=sample	Mozilla/4.2 (Win98,I)
2	123.456.78.9	- -	[21/Jan/2000:15:04:41 -0600]	"GET 1.html HTTP/1.0"	200	2450	http://www.sample.edu/index.html	Mozilla/4.2 (Win98,I)
3	123.456.78.9	- -	[21/Jan/2000:15:04:41 -0600]	"GET A.html HTTP/1.0"	200	2180	http://www.sample.edu/index.html	Mozilla/4.2 (Win98,I)
4	123.456.78.9	- -	[21/Jan/2000:15:04:44 -0600]	"GET / HTTP/1.0"	200	1300	http://search.edu/?key=sample	Mozilla (IE5.0,WinNT)
5	123.456.78.9	- -	[21/Jan/2000:15:04:45 -0600]	"GET 1.html HTTP/1.0"	200	2450	http://www.sample.edu/index.html	Mozilla (IE5.0,WinNT)
6	123.456.78.9	- -	[21/Jan/2000:15:04:45 -0600]	"GET A.html HTTP/1.0"	200	2180	http://www.sample.edu/index.html	Mozilla (IE5.0,WinNT)
7	123.456.78.9	- -	[21/Jan/2000:15:05:18 -0600]	"GET C.html HTTP/1.0"	200	1450	http://www.sample.edu/1.html	Mozilla/4.2 (Win98,I)
8	123.456.78.9	- -	[21/Jan/2000:15:05:19 -0600]	"GET D.html HTTP/1.0"	200	2560	http://www.sample.edu/A.html	Mozilla (IE5.0,WinNT)
9	123.456.78.9	- -	[21/Jan/2000:15:05:43 -0600]	"GET index.html HTTP/1.0"	200	1300	-	Mozilla/4.2 (Win98,I)
10	123.456.78.9	- -	[21/Jan/2000:15:05:43 -0600]	"GET 1.html HTTP/1.0"	200	2450	http://www.sample.edu/index.html	Mozilla/4.2 (Win98,I)
11	123.456.78.9	- -	[21/Jan/2000:15:05:45 -0600]	"GET A.html HTTP/1.0"	200	2180	http://www.sample.edu/index.html	Mozilla/4.2 (Win98,I)
12	123.456.78.9	- -	[21/Jan/2000:15:06:01 -0600]	"GET H.html HTTP/1.0"	200	1210	http://www.sample.edu/C.html	Mozilla/4.2 (Win98,I)
13	123.456.78.9	- -	[21/Jan/2000:15:06:06 -0600]	"GET B.html HTTP/1.0"	200	1450	http://www.sample.edu/1.html	Mozilla/4.2 (Win98,I)
14	123.456.78.9	- -	[21/Jan/2000:15:06:18 -0600]	"GET 2.html HTTP/1.0"	200	2100	http://www.sample.edu/B.html	Mozilla/4.2 (Win98,I)
15	123.456.78.9	- -	[21/Jan/2000:15:06:19 -0600]	"GET F.html HTTP/1.0"	200	1530	http://www.sample.edu/B.html	Mozilla/4.2 (Win98,I)
16	123.456.78.9	- -	[21/Jan/2000:15:06:36 -0600]	"GET I.html HTTP/1.0"	200	1830	http://www.sample.edu/C.html	Mozilla/4.2 (Win98,I)
17	123.456.78.9	- -	[21/Jan/2000:15:07:24 -0600]	"GET G.html HTTP/1.0"	200	1550	http://www.sample.edu/2.html	Mozilla/4.2 (Win98,I)
18	123.456.78.9	- -	[21/Jan/2000:15:07:49 -0600]	"GET 4.html HTTP/1.0"	200	1240	http://www.sample.edu/F.html	Mozilla/4.2 (Win98,I)
19	123.456.78.9	- -	[21/Jan/2000:15:07:49 -0600]	"GET F1.html HTTP/1.0"	200	2770	http://www.sample.edu/F.html	Mozilla/4.2 (Win98,I)
20	123.456.78.9	- -	[21/Jan/2000:15:09:59 -0600]	"GET 5.html HTTP/1.0"	200	1390	http://www.sample.edu/G.html	Mozilla/4.2 (Win98,I)
21	123.456.78.9	- -	[21/Jan/2000:15:10:02 -0600]	"GET G3.html HTTP/1.0"	200	2380	http://www.sample.edu/G.html	Mozilla/4.2 (Win98,I)

Figure 5.2: Sample Usage Data.

amount of automated agents and spider programs traversing links and requesting files from a Web site can often lead to skewed analysis results. The simplest method for dealing with agent traffic is to check the agent field of the usage data. Many agents and spiders voluntarily declare themselves by using an agent string other than one that indicates a browser[2]. A simple string match during the data cleaning phase can remove (or identify for further analysis) a significant amount of agent traffic. Another indicator for identifying agents is an access of a "`robots.txt`" file. Agents that follow the conventions of [1] will always check for the existence of a file named `robots.txt` in order to get instructions about which pages are "off-limits" to agents. However, some agents do not identify themselves through the agent field or by requesting the `robots.txt` file. Heuristics such as looking for sessions that request only HTML files but skip all of the associated graphics requests can be used to filter out agent traffic after the user/session identification step. In order for this to occur, the graphics request must be passed through to the next preprocessing phase. Other reasons for leaving the graphics requests in the data are to check for incomplete page views (users hit the "stop" button on the client browser) or to help identify the content served (the same HTML file with different graphics may represent different content for reporting purposes). The tradeoff involved with the decision to keep or remove the graphics requests in the data cleaning phase has to do with the amount of data. A typical commercial Web site may have ten graphics files for every HTML file served. This results in an order of magnitude increase in the amount of data passed to the next preprocessing step if graphics requests are not stripped out.

The status code and size fields are easily parsed and do not present any preprocessing difficulties. However, the problem with the status codes is that they only reflect the point of view of the Web server. If a separate content server is used, the

[2] The strings "mozilla" or "IE" appear in most browser agent fields.

error codes can be meaningless. A content server can return nonsense in response to a request, and as long as it can be sent via HTTP, the Web server will log the request as a success (code 200). Similar to the status field, the use of a content server can render the size field useless, and a dash will often be logged instead of a number of bytes. While the status code and size of each request should be included in the schema, any analysis needs to take into account the possibility of the values being incorrect.

The final step of the data cleaning task is to normalize the URIs and parse any CGI data. Most Web servers treat a request for a directory as a request for a default file such as "`index.html`" or "`home.html`". The directory request may come in with or without a trailing slash. Also, the "www" in the front of a URI is optional. This means that requests for `sample.edu`, `www.sample.edu`, `www.sample.edu/`, and `www.sample.edu/index.html` are all for the same file. The data cleaning must chose a common form for each page file. As shown in Figure 5.2, the site name is not logged in the request, but is always logged in the referrer field. This is because, by definition, all requests are coming to the same Web server and the site name is known. However, referring files can come from other Web sites, so the full URI is used. The site name (e.g. `www.sample.edu`) must either be stripped off of all requests and internal referring page files, or added on to every request. Finally, any URIs that have CGI data attached must be parsed further to break up the name/value pairs into separate fields.

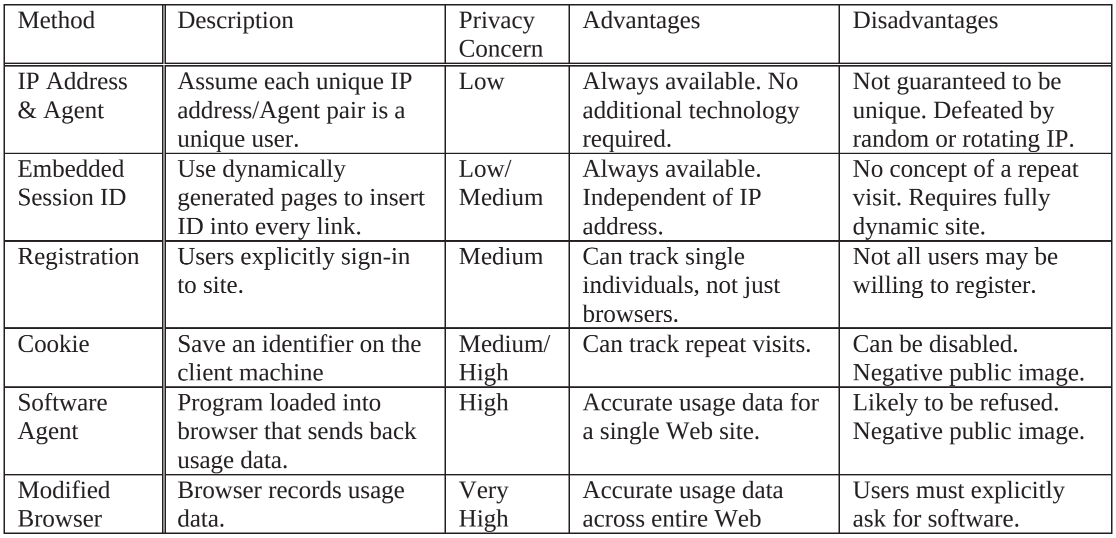
The sample log shown in Figure 5.2 has already been through part of the data cleaning step. In this case, the graphics requests have been stripped out. The only tasks remaining are to fully parse the data fields and normalize the URIs. For the example, this simply means breaking up the request field into it's three component parts and stripping the site name from the internal requests. The request and referrer in line 4 require URI normalization. The CGI data for the referring URIs in lines 1 and 4 must also be parsed.

5.1.2 User and Session Identification

As previously mentioned, common technologies such as cookies and embedded session IDs make user identification almost trivial. Client-side tracking methods also remove most of the difficulties associated with identifying users. However, the level of detail and accuracy of the data collected by the various methods are inversely proportional to the invasiveness of the method in terms of user privacy. Client-side tracking provides by far the most accurate usage data, but is also the largest invasion of privacy. Cookies are next, followed by registration and embedded session IDs. The drawbacks associated with the more intrusive methods is that data are available for a smaller percentage of the visitors to a Web site. Relatively few users are willing to accept a client-side tracking mechanism, and usually only in exchange for substantial benefits such as free internet access or a free computer [4, 16, 12]. In contrast, embedded session IDs will provide data on every user that accesses the Web site. User registration provides a great method for tracking individuals across multiple machines or browsers, but will not necessarily be acceptable to everyone who visits a Web site. A requirement for users to register may end up driving traffic away from a site. Figure 5.3 provides a list of the most common user identification methods along with the pros and cons.

Unless one of the tracking mechanisms discussed above is used, only the IP address, agent, and the server-side click-stream are available to identify users and server sessions. Some of the typically encountered problems are:

1. Single IP address/Multiple Server Sessions - Internet service providers (ISPs) typically have a pool of proxy servers through which users can access the Web.



Method	Description	Privacy Concern	Advantages	Disadvantages
IP Address & Agent	Assume each unique IP address/Agent pair is a unique user.	Low	Always available. No additional technology required.	Not guaranteed to be unique. Defeated by random or rotating IP.
Embedded Session ID	Use dynamically generated pages to insert ID into every link.	Low/ Medium	Always available. Independent of IP address.	No concept of a repeat visit. Requires fully dynamic site.
Registration	Users explicitly sign-in to site.	Medium	Can track single individuals, not just browsers.	Not all users may be willing to register.
Cookie	Save an identifier on the client machine	Medium/ High	Can track repeat visits.	Can be disabled. Negative public image.
Software Agent	Program loaded into browser that sends back usage data.	High	Accurate usage data for a single Web site.	Likely to be refused. Negative public image.
Modified Browser	Browser records usage data.	Very High	Accurate usage data across entire Web	Users must explicitly ask for software.

Figure 5.3: Common User Identification Methods.

A single proxy server may have several users accessing a Web site, potentially over the same time period.

2. Multiple IP address/Single Server Session - Some ISPs or privacy tools randomly assign each request from a user to one of several IP addresses. In this case, a single server session can have multiple IP addresses.

3. Multiple IP address/Single User - A user that accesses the Web from different machines will have a different IP address from session to session. This makes tracking repeat visits from the same user difficult.

4. Multiple Server Sessions/Single User - This occurs when a user opens up more than one browser window, and accesses different portions of a Web site simultaneously.

5. Single Client/Multiple Users - This occurs when more than one individual uses the same computer, such as families or public access machines.

The first two problems only occur for the IP address/agent method of user identification. Embedded session IDs have no notion of a repeat visit, meaning the third problem, multiple IP address/single user, is irrelevant for this method. All of the identification methods except the modified browser are subject to the fourth problem - multiple server sessions/single user. Finally, all of the identification methods except user registration are subject to the last problem, single client/multiple users, unless some sort of browser login is used[3].

Once users have been identified, the click-stream for each user must be divided into sessions. Since page requests from other servers are not typically available, it is difficult to know when a user has left a Web site. A thirty minute timeout is often used as the default method of breaking a user's click-stream into sessions. The thirty minute timeout is based on the results of [34], where 1.5 standard deviations were added to the mean inactivity length of 9.3 minutes to get a cutoff of 25.5 minutes. This cutoff has been rounded up to 30 minutes as an industry standard. When a session ID is embedded in each URI, the definition of a session is set by the content server. Modified browsers which track usage across the entire Web are able to track the *next* click outside of a specific Web site, and can declare a server session to be ended when a user moves to a different site or exits from the browser (Although a timeout must still be used for the case where the browser is left open, but no browsing activity occurs).

For the "worst-case", where only the information in the ECLF Web log is available, *users* and *sessions* should be considered to be the same since there is no way of knowing if a different session from the same IP address/agent pair is the same user. As first presented in [84], a reasonable assumption to start with is that each different agent type for an IP address represents a different session. The next step of the heuristic for session identification is to use the referring page file. If the referring page file for a request is not part of an open session, it is assumed that the request

[3] Both Internet Explorer and Netscape have this feature.

is coming from a different session. The final part of the heuristic is the handling of requests with the same IP address and agent that could have come from more than one active session. The heuristic assigns the request to the session that is *closest* to the referring page at the time of the request. *Closeness* is defined as the minimum number of links that would need to be traversed from the current page view in order to have the referring page file displayed in the client browser. In the case where multiple sessions are the same distance from a page request, the heuristic assigns the request to the session with the most recent referrer access in terms of time. In theory, static sites can also be subject to proxy-level caching. This means that multiple sessions can access the same page file with only a single request making it back to the Web server log. However, the guiding principle of the session identification heuristic is to assume the simplest solution in the absence of any additional information. Also, because proxy-level caching can be effectively defeated by dynamic content or disabled with negative expiration dates for static content, the heuristic does not attempt to deal with the effect of proxy-level caching on sessions.

The session identification heuristic is summarized in Table 5.1. Line 7 checks to see if the session has timed out or if the referring file is not present in any of the open session histories. If so, a new session is opened in lines 8 and 9. Since the log has already been sorted by IP address/agent, all of the open session histories are potential candidates for the page file access being processed. Line 11 calls the function *Distance* shown in Table 5.2. Given a list of histories and a page file, f, the Distance function finds the history that most recently accessed f. Recency is defined as the number of page files between the end of the history and f (line 9 of Table 5.2 checks for this). In the case of ties, the time between the last file access and the last access of f in the histories is compared (shown in line 13). Finally, if the times are equivalent, by default, the access is assigned to the history with the lower index. A

Table 5.1: Session Identification Heuristic.

```
Heuristic Identify
1. Let $H_i = \{f_1, \ldots, f_n\}$ denote a time 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
6.     for each $l_j$ do
7.         if $((t_j - t_{j-1}) > \mathrm{T}) \bigvee r_j \ni \{H_0, \ldots, H_m\}$ then
8.             Increment i
9.             Add $l_j$ to $H_i$
10.        else
11.            assign = Distance($H$,$r_j$)
12.            Add $l_j$ to $H_{\mathrm{assign}}$
13. end;
```

random assignment could also be used to break ties. The index of the history that f should be assigned to is returned by the Distance function, and line 12 of the session identification heuristic uses the returned index to add the page file to the appropriate history.

Applying the session identification heuristic to the Figure 5.2 log, lines 4,5,6, and 8 would be identified as a separate session (designated as session 3) due to the different agent. Figure 5.4 shows the remaining file accesses, which are all for the IP/Agent combination of "123.456.78.9/Mozilla/4.2 (Win98,I)." As shown, the accesses are assigned to session 1 until time t_2 (From lines 9 through 11 of the example log), where a request is made for `index.html` and the referrer is null. This is not part of the current active session, so a new session is opened. The requests for `1.html` and `A.html` at time t_2 could be a part of either open session, since they both contain `index.html`. The calls to *Distance* result in both requests being assigned to session 2. The only remaining request that could belong to either session is at time t_4, where

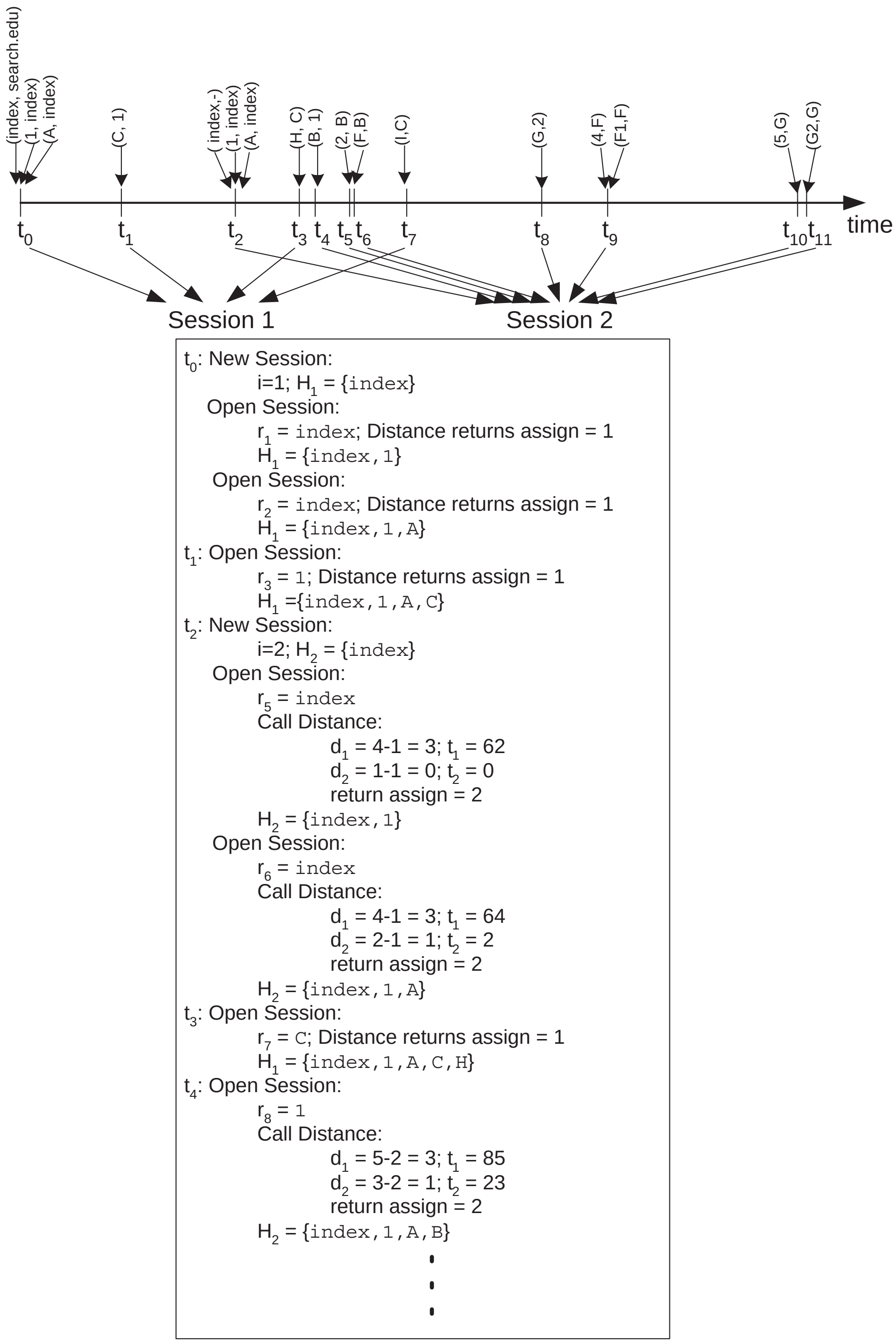
Table 5.2: Distance Function.

```
Function Distance($H, f$)
Let $H_i$ denote a time ordered session history.
Let $f$ denote a page file.
set min = $\infty$
for each $H_i \in H$ do
    if $f \in H_i$
        $d_i = H_i.size() - H_i.index(f)$
        $t_i = H_i.t_n - H_i.t_f$
        if ($d_i <$ min)
            assign = $i$
            min = $d_i$
        else if ($d_i =$ min)
            if ($t_i < t_{\text{assign}}$)
                assign = $i$
return assign
end;
```

B.html is requested from 1.html (Line 13 of the log). Again, 1.html is closer to the end of the session 2 history, causing the heuristic to assign the file access to session 2. Note that this is not guaranteed to be correct if client-side caching is present. The user for session 2 could have left the site after viewing the home page, and the session 1 user could have used the "back" button to get to the home page before requesting page B.html. The remaining requests at times t_5 through t_{11} can only have come from one of the two open sessions. The full results of applying the heuristic to the Figure 5.2 example are shown in Figure 5.5.

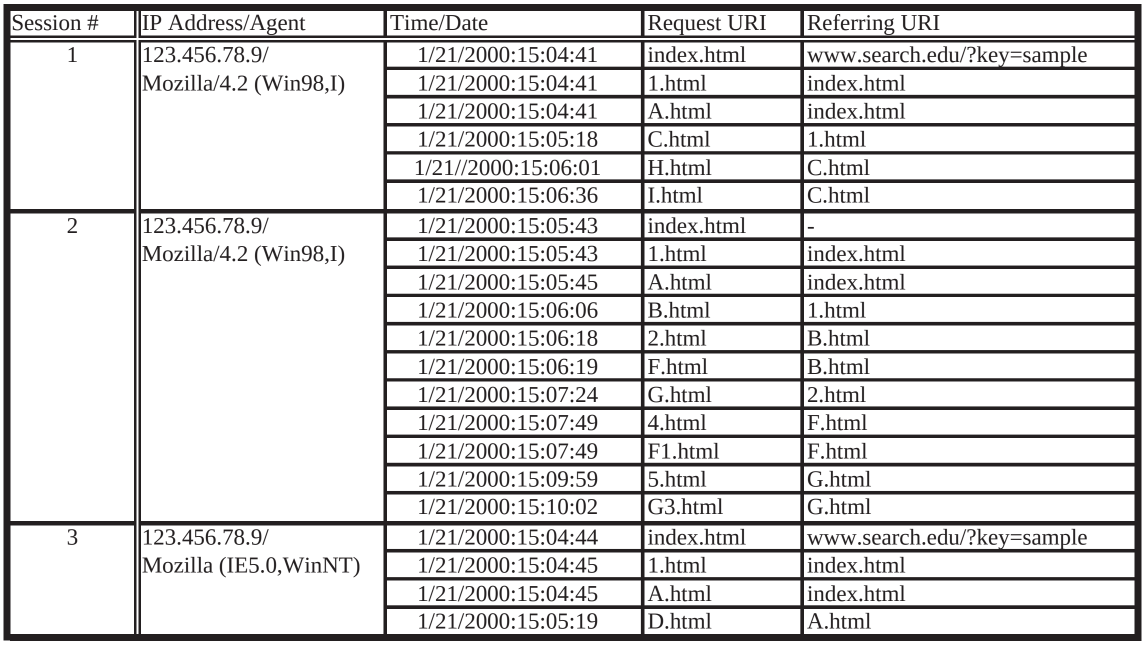
5.1.3 Page View Identification

The page view identification step determines which page file requests are part of the same page view and what content was served. This is necessary to provide meaningful results in the pattern analysis phase. If this step is not performed, the discovered



```
$t_0$: New Session:
        i=1; $H_1$ = {index}
    Open Session:
        $r_1$ = index; Distance returns assign = 1
        $H_1$ = {index,1}
    Open Session:
        $r_2$ = index; Distance returns assign = 1
        $H_1$ = {index,1,A}
$t_1$: Open Session:
        $r_3$ = 1; Distance returns assign = 1
        $H_1$ ={index,1,A,C}
$t_2$: New Session:
        i=2; $H_2$ = {index}
    Open Session:
        $r_5$ = index
        Call Distance:
                $d_1$ = 4-1 = 3; $t_1$ = 62
                $d_2$ = 1-1 = 0; $t_2$ = 0
                return assign = 2
        $H_2$ = {index,1}
    Open Session:
        $r_6$ = index
        Call Distance:
                $d_1$ = 4-1 = 3; $t_1$ = 64
                $d_2$ = 2-1 = 1; $t_2$ = 2
                return assign = 2
        $H_2$ = {index,1,A}
$t_3$: Open Session:
        $r_7$ = C; Distance returns assign = 1
        $H_1$ = {index,1,A,C,H}
$t_4$: Open Session:
        $r_8$ = 1
        Call Distance:
                $d_1$ = 5-2 = 3; $t_1$ = 85
                $d_2$ = 3-2 = 1; $t_2$ = 23
                return assign = 2
        $H_2$ = {index,1,A,B}
                        .
                        .
                        .
```

Figure 5.4: File Accesses for “123.456.78.9/Mozilla/4.2 (Win98,I)”



Session #	IP Address/Agent	Time/Date	Request URI	Referring URI
1	123.456.78.9/ Mozilla/4.2 (Win98,I)	1/21/2000:15:04:41	index.html	www.search.edu/?key=sample
		1/21/2000:15:04:41	1.html	index.html
		1/21/2000:15:04:41	A.html	index.html
		1/21/2000:15:05:18	C.html	1.html
		1/21//2000:15:06:01	H.html	C.html
		1/21/2000:15:06:36	I.html	C.html
2	123.456.78.9/ Mozilla/4.2 (Win98,I)	1/21/2000:15:05:43	index.html	-
		1/21/2000:15:05:43	1.html	index.html
		1/21/2000:15:05:45	A.html	index.html
		1/21/2000:15:06:06	B.html	1.html
		1/21/2000:15:06:18	2.html	B.html
		1/21/2000:15:06:19	F.html	B.html
		1/21/2000:15:07:24	G.html	2.html
		1/21/2000:15:07:49	4.html	F.html
		1/21/2000:15:07:49	F1.html	F.html
		1/21/2000:15:09:59	5.html	G.html
		1/21/2000:15:10:02	G3.html	G.html
3	123.456.78.9/ Mozilla (IE5.0,WinNT)	1/21/2000:15:04:44	index.html	www.search.edu/?key=sample
		1/21/2000:15:04:45	1.html	index.html
		1/21/2000:15:04:45	A.html	index.html
		1/21/2000:15:05:19	D.html	A.html

Figure 5.5: Results of Session Identification Heuristic.

patterns can be dominated by page files that make up a single popular page view. Page view identification relies heavily on the results of the structure and content preprocessing for the site. While the exact content served as a result of each user action is often available from the request field in the Web server logs, it is sometimes necessary to have access to the content server information as well. Since content servers can maintain state variables for each active session, the information necessary to determine exactly what content is served by a user request is not always available in the URI.

The first part this step is to identify the content served by each page file request in a session. The results of the content preprocessing are used to map each parsed URI (or combination of URI and content server information) to a particular piece of content (or content that is considered equivalent for reporting and analysis purposes).

The second part of page view identification is determining which page files make up each page view. Each session must start with a *seed* page view - i.e. an initial page file or set of page files from which all subsequent page views will be derived. In the vast majority of cases, the seed page view is made up of a single file, or starts with a single file that defines the frame structure and immediately causes additional page files to be requested. It is very rare for an unrelated site to link to more than one page file of a different site in a single hypertext link. However, it is possible, and for these cases all of the page files contributing to the seed page view would have to be explicitly entered into the algorithm. Once the seed page view is specified for a session, each subsequent request is checked against the site structure to determine which frame (or frames) is being replaced. The "new" page view consists of the requested file plus all of the page files that are remaining in the browser display from the previous page view. Each distinct page view discovered in a log is given an identifier. The page views are not given identifiers in advance, since for a large complex site the number of possible page views is vast, and many of them may never be accessed in a given set of usage data.

The page view identification step can find errors or anomalies in a Web site by identifying sessions with incomplete page views. As mentioned in the discussion about data cleaning, if the graphics files are not stripped out, a session with complete page views *except* the graphics files can indicate an automated agent or spider[4]. If one page file of a particular page view is frequently missing, it may indicate an error, or a particularly slow-loading view that users are clicking on "stop" before the page view is completely loaded.

The algorithm for converting page file accesses into page views is shown in Table 5.3. The initial page view is set in line 6. Line 8 checks to see if there are already

[4] This can also mean a user has turned off his or her graphics in order to speed up the browsing process.

Table 5.3: Page View Identification Algorithm.

```
Algorithm PageView
Let $O(f_1, \ldots, f_n)$ denote an open list of page files.
Let $T(t_1, \ldots, t_n)$ denote the times associated with $O$
Let $C_v(f_1, \ldots, f_m)$ denote the current view.
Let $r_v$ denote the view referring file.
Let $f_i$ and $r_i$ denote the requested and referring files.
Set initial page view, $k = 0$
for each log entry, $L_i$, do
    if $O \equiv \emptyset$ then
        set $r_v = r_i$; $t_k^e = t_i$
        if $r_v \ni C_v$ then
            $C_v = \text{Complete}(S, r_v)$
    if $r_i = r_v$ then
        add $f_i$ to $O$; $t_i$ to $T$
        if $(l = \text{FindLink}(O, r_v))) \neq \text{NULL}$ then
            $C_v = \text{ReplaceView}(O, l)$
            increment $k$
            set $v_k = C_v$; $h_k = r_v$; $c =$false
            set $t_k^f = \min(T)$; $t_k^l = \max(T)$
            set $O = \emptyset$
    else
        Incomplete view, run error handling
set $t_k^e =$null
end;
```

files in the open list, O. Since a link may replace multiple page files at once, an "open list" of files must be maintained until a matching link is found. An empty open list indicates that a page view was just formed with the last log entry, and the current log entry is the start of a new page view. The referring file for the view and the end time for the previous page view are set in line 9. If the view referring file is not part of the previous page view, a cached page access must have occurred, and the function *Complete* is call in line 11. The minimum requirements of the *Complete* function are to return a page view that contains the referring file in order for the *PageView* algorithm to continue. Whether or not it actually adds the cached views to the session is optional. For the current discussion, it will be assumed that the *Complete* function simply returns the most recent page view from session S that contains the file, r_v. In the next section, a version of the *Complete* function that adds cached page views to the session will be presented.

Continuing with the *PageView* algorithm, line 12 checks to see if the current referring file is the correct one for the view being built, and if so, adds the requested page file and the time to the appropriate list in line 13. If the referring file changes while a new view is being built, an error handling routine is called at line 21. The actions of the error handling routine can vary. The entire session can be thrown out as corrupt, or the current contents of each frame location can be declared to be a page view (even if it is not valid by the site structure) and the algorithm can continue. This type of error can only occur when a link results in more than one page file requests. Any standard hypertext link will result in a single page file request and a new page view after a single file is added to the open list. The *FindLink* function in line 14 checks to see if the contents of the open list are equivalent to any of the links for the referring file. If a matching link is found, the current view is set in line 15 with a call to *ReplaceView*. The *ReplaceView* function takes the current view,

Table 5.4: FindLink and ReplaceView Functions.

```
Let $F(f_1, \ldots, f_n)$ denote a list of page files.
Let $r$ denote a single page file.
Let $L(p, c_1, \ldots, c_m)$ denote a global list of links.

Function FindLink$(F, r)$
for each $L$ where $p = r$, do
    if $L_i(r, c) = F$
        return $i$
end;

Function ReplaceView$(F, l)$
for each $c \in L_l$, do
    set $F(f_i) = c_i$
return $F$
end;
```

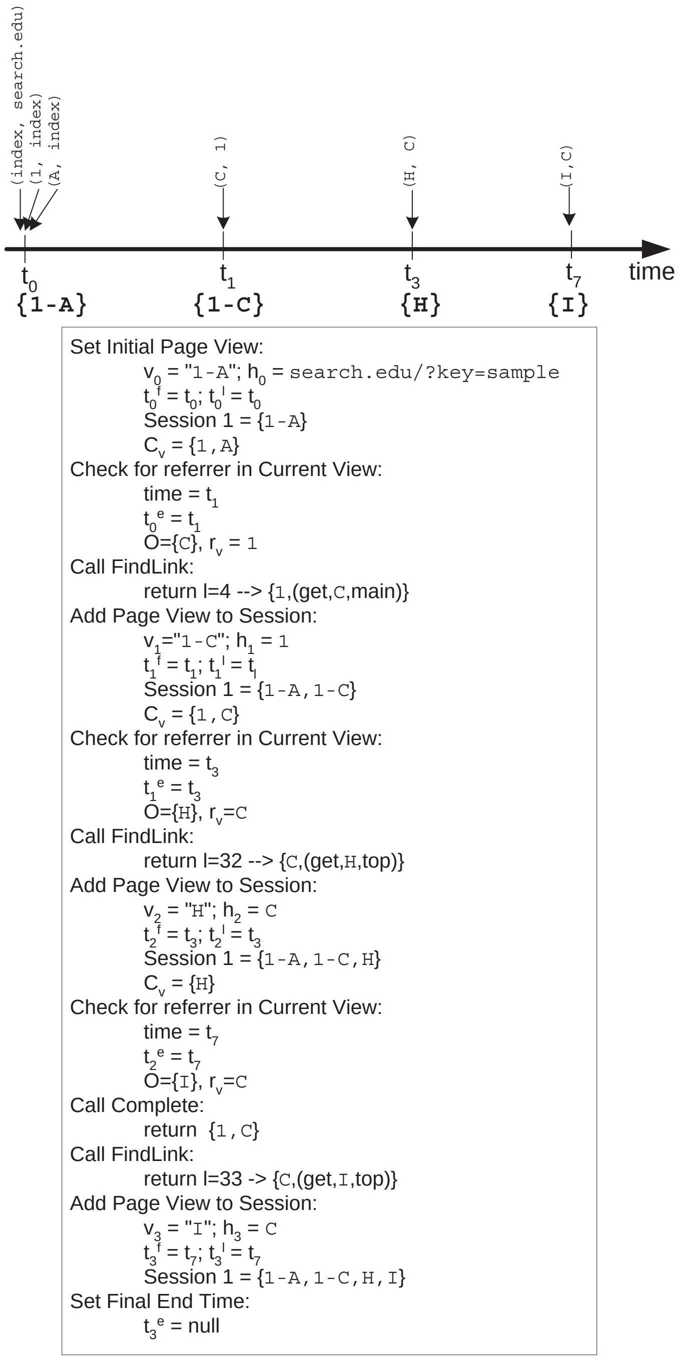
C_v, and replaces the appropriate frame locations with the page files for link l. The *FindLink* and *ReplaceView* functions are shown in Table 5.4. Lines 16 through 18 add the new page view and parameters to the session . As shown in line 19, after each match and creation of a new page view, the open list, O, is cleared in order to start building a new page view. Finally, line 22 sets the last time to null since there are no subsequent file requests for the session.

For the case of the example site, each URI directly identifies a distinct piece of content, making the content identification step trivial. For the example, the identifiers will consist of the names of the page files comprising the page view with the file extension stripped off. The `index.html` page file is used solely to set up the frame schema - in this case a *left* frame and a *main* frame. Therefore, all of the sessions for this site should start with requests for `index.html, 1.html`, and `A.html`, which leads to a seed page view of `1-A`, e.g. "the home page." Each of the three sessions in the example start with the home page view, as expected. Figure 5.6 shows how the algorithm would be applied to session 1. After the initial page view of "1-A" is set, the request for C.html at time t_1 is read. The referring file, 1.html, is part of the current page view, so *FindLink* is called, and the fourth link from equation 4.4 is returned. This is used in a call to *ReplaceLink* to add page view 1-C to the session. Page view H is similarly added at time t_3. However, at time t_7, the referring file, C.html, is not part of the current page view, H. Therefore, *Complete* is called, which returns "1-C" as the last page view containing C.html. The final page view, I, is then added to the session. Figure 5.7 shows the results of applying the page view algorithm to the entire example log.

5.1.4 Path Completion

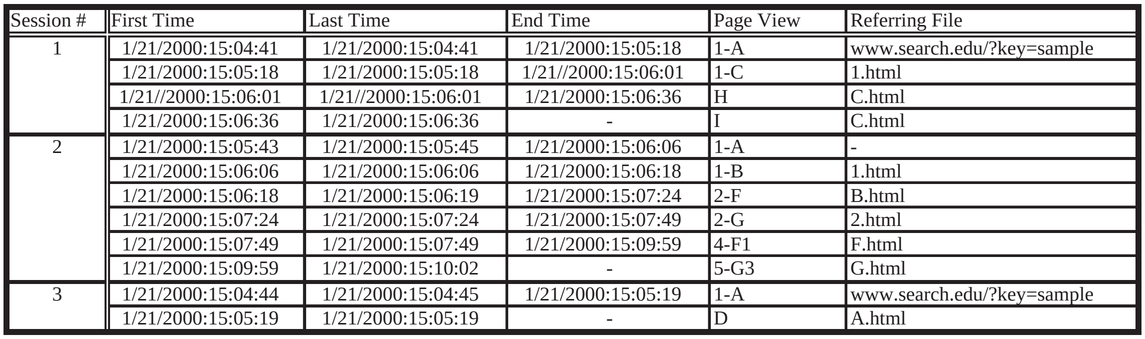
The final major step for preprocessing usage data is that of inferring cached page references. As discussed previously, the only verifiable method of tracking cached page views is to monitor usage from the client side with either a software agent or modified browser. However, in the absence of a client-side tracking mechanism, the referrer field for each request can be used to detect some of the instances when cached pages have been viewed. As shown in the *PageView* algorithm example, if the referring page file for a page view is not part of the previous page view, the user must have accessed a cached page. Similar to identifying sessions with only the IP address, agent, and site structure, there is no provably correct algorithm for determining what cached pages were viewed, only heuristics.

There are three common methods that browsers provide to users for accessing a cached page view. The most common method (based on the results of GVU surveys [34]) is the use of the *back* button provided at the top of the popular browsers. The browser maintains a history stack that has page views pushed on and popped off throughout a user session. The second most common method of cached page access



```
Set Initial Page View:
    $v_0$ = "1-A"; $h_0$ = search.edu/?key=sample
    $t_0^f = t_0$; $t_0^l = t_0$
    Session 1 = {1-A}
    $C_v$ = {1,A}
Check for referrer in Current View:
    time = $t_1$
    $t_0^e = t_1$
    O={C}, $r_v$ = 1
Call FindLink:
    return l=4 --> {1,(get,C,main)}
Add Page View to Session:
    $v_1$="1-C"; $h_1$ = 1
    $t_1^f = t_1$; $t_1^l = t_l$
    Session 1 = {1-A,1-C}
    $C_v$ = {1,C}
Check for referrer in Current View:
    time = $t_3$
    $t_1^e = t_3$
    O={H}, $r_v$=C
Call FindLink:
    return l=32 --> {C,(get,H,top)}
Add Page View to Session:
    $v_2$ = "H"; $h_2$ = C
    $t_2^f = t_3$; $t_2^l = t_3$
    Session 1 = {1-A,1-C,H}
    $C_v$ = {H}
Check for referrer in Current View:
    time = $t_7$
    $t_2^e = t_7$
    O={I}, $r_v$=C
Call Complete:
    return  {1,C}
Call FindLink:
    return l=33 -> {C,(get,I,top)}
Add Page View to Session:
    $v_3$ = "I"; $h_3$ = C
    $t_3^f = t_7$; $t_3^l = t_7$
    Session 1 = {1-A,1-C,H,I}
Set Final End Time:
    $t_3^e$ = null
```

Figure 5.6: Session 1 Page Views



Session #	First Time	Last Time	End Time	Page View	Referring File
1	1/21/2000:15:04:41	1/21/2000:15:04:41	1/21/2000:15:05:18	1-A	www.search.edu/?key=sample
	1/21/2000:15:05:18	1/21/2000:15:05:18	1/21//2000:15:06:01	1-C	1.html
	1/21//2000:15:06:01	1/21//2000:15:06:01	1/21/2000:15:06:36	H	C.html
	1/21/2000:15:06:36	1/21/2000:15:06:36	-	I	C.html
2	1/21/2000:15:05:43	1/21/2000:15:05:45	1/21/2000:15:06:06	1-A	-
	1/21/2000:15:06:06	1/21/2000:15:06:06	1/21/2000:15:06:18	1-B	1.html
	1/21/2000:15:06:18	1/21/2000:15:06:19	1/21/2000:15:07:24	2-F	B.html
	1/21/2000:15:07:24	1/21/2000:15:07:24	1/21/2000:15:07:49	2-G	2.html
	1/21/2000:15:07:49	1/21/2000:15:07:49	1/21/2000:15:09:59	4-F1	F.html
	1/21/2000:15:09:59	1/21/2000:15:10:02	-	5-G3	G.html
3	1/21/2000:15:04:44	1/21/2000:15:04:45	1/21/2000:15:05:19	1-A	www.search.edu/?key=sample
	1/21/2000:15:05:19	1/21/2000:15:05:19	-	D	A.html

Figure 5.7: Results of Page View Identification Algorithm.

is a click on a hypertext link that has already been traversed in the current session. The link may or may not be in the history stack, depending on the previous path taken through the Web site. The third method is to directly access a page from the recent *history* list provided by the browser. Since it is impossible to determine which method was used to reference a cached page view, the path completion heuristic assumes that the most popular method, the back button, is used by default. If the page view is not in the back stack, the full session history is searched for the page view. For the session history search, it is assumed that the shortest available path was used to reach the page. Because the session history contains all of the page views that are available from the direct history list access method, the direct access method will never be used by the heuristic. While the conclusion of the heuristic is not guaranteed to be correct, it will always be consistent given a specific situation.

The final task of the path completion heuristic is to assign a time to the inferred page views. In the absence of any additional information, the heuristic assigns a constant view time for each of the cached page views. The assumption is that the user is revisiting the pages simply for navigation purposes.

The path completion heuristic is shown in Table 5.5. It is a function called from the *PageView* algorithm when the view referring file is not in the current page view.

Lines 6 through 9 build the stack that would be associated with the back button for the session, S. Line 10 sets the location of the "next" page view in the session. Lines 12 through 16 loop through the back stack looking for the referring file, r_v. If the referring file is found, the portion of the back stack popped off is added to the session. If r_v is not found, the full session history is searched in lines 18 through 22. Finally, lines 24 through 26 set the times for the added page views based on a fixed time interval, T. The page view that contains r_v is returned by the function to *PageView*.

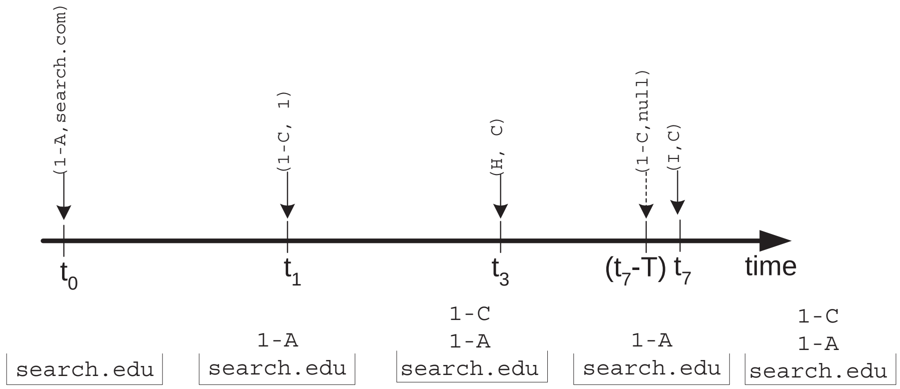
Figure 5.8 shows how the heuristic would be run on session 1. A back stack is built from the session history and then checked for r_v, which in this case is equal to `C`. `C` is immediately found in the page view on the top of the stack, `1-C`. The times for the inferred page view and the new end time for page view `H` are set and `1-C` is returned to the *PageView* algorithm. Session 2 is more complicated and the portion from time $t_1 0$ on is shown in Figure 5.9. The back stack does not include the full history since the previous use of the back button earlier in the session popped `2-G` and one instance of `2-F` off the stack. The referring file, `G`, is not found in the back stack and the full history must be searched. Page view `2-G` is added to the session at index 7. The back stack is then re-checked, and `2-F` is inserted at index 7 (pushing `2-G` to index 8) in order to complete the path from `4-F1` to `2-G`. The appropriate times are updated, and `2-G` is returned to *PageView*. The results of running the heuristic on the full example are shown in Figure 5.10.

5.2 Episode Identification

Episode identification is an optional preprocessing step that can be performed after the required preprocessing steps. An episode is defined by the W3C as a *semantically meaningful* subset of a user session. Since only server sessions are available in this case, episodes are more narrowly defined as subsets of server sessions. Any num-

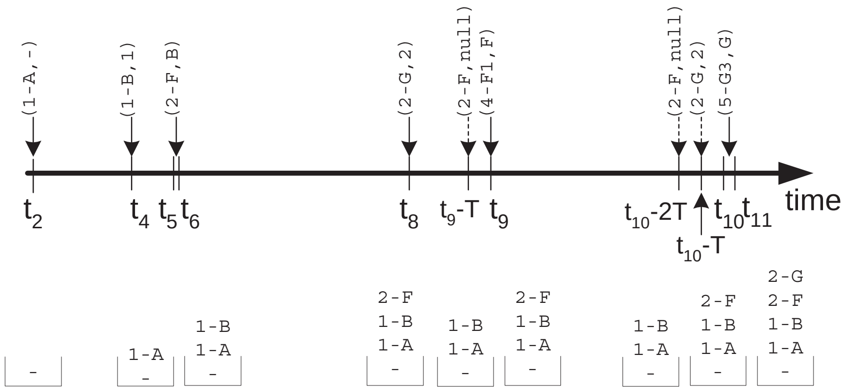
Table 5.5: Path Completion Heuristic.

```
Function Complete($S, r_v$)
1. Let $S$ denote a session of page views, $V_i$
2. Let $r_v$ denote the referring page file
3. Let $stk$ denote a stack
4. Let $T$ denote a fixed time interval
5. for each $V \in S$ do
6.     if $h_i \neq$ null then
7.         $stk.push(v_{i-1})$
8.     else
9.         $stk.pop()$
10. set n = $S.size() + 1$
11. set notfound = true
12. while $((v_s = stk.pop()) \neq null)$ AND notfound
13.     add $v_s$ to $S_{temp}$ with $c$=true
14.     if $r_v \in v_s$ then
15.         set notfound = false
16.         insert $S_{temp}$ into $S$ at $n$
17. if (notfound) then
18.     for $j = n - 2$ down to 1 do
19.         if $r_v \in v_j$ then
20.             insert $V_j$ into $S$ with $c$=true at $n$
21.             set $r_v = h_j$
22.             goto 11
23. set $newsize = S.size()$
24. for $k = newsize$ down to $n$ do
25.     set $t^e_k = t^f_{k+1}$
26.     set $t^l_k = t^e_k - T; t^f_k = t^l_k$
27. set $t^e_{n-1} = t^f_n$
28. return $v_{newsize}$
29. end;
```



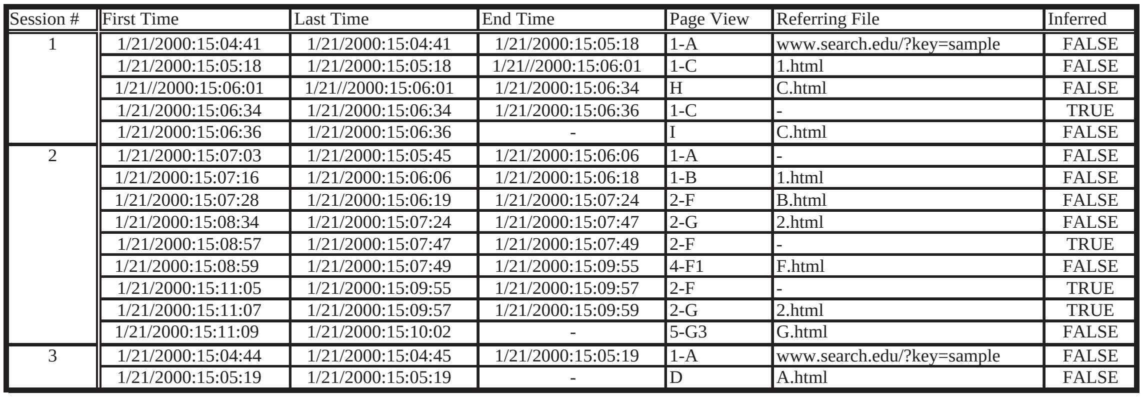
```
S = {1-A,1-C,H}, $r_v$ = C
Build Stack:
          stk = {search.edu,1-A,1-C}
          n = 4
Check Stack:
          $S_{temp}$ = {1-C}
          S = {1-A,1-C,H,1-C}
Update Times:
          newsize = 4
          $t^e_4 = t_7$; $t^l_4 = t_7 - T$; $t^f_4 = t^l_4$
          $t^e_3 = t^f_4$
return 1-C
```

Figure 5.8: Path Completion for Session 1.



```
S = {1-A,1-B,2-F,2-G,2-F,4-F1}, $r_v$ = G
Build Stack:
        stk = {1-A,1-B,2-F}
        n = 7
Check Stack:
        $S_{temp}$ = {}
        G not found in stack
Check History
        at j=4, G is found in 2-G
        S = {1-A,1-B,2-F,2-G,2-F,4-F1,2-G}
        $r_v$ = 2
Check Stack:
        stk = {1-A,1-B,2-F}
        $S_{temp}$ = {2-F}
        S ={1-A,1-B,2-F,2-G,2-F,4-F1,2-F,2-G}
Update Times:
        newsize = 8
        $t^e_8 = t_{10}$; $t^l_8 = t_{10}$- T; $t^f_8 = t^l_8$
        $t^e_7 = t^f_8$; $t^l_7 = t^f_8$- T; $t^f_7 = t^l_7$
        $t^e_6 = t^f_7$
return 2-G
```

Figure 5.9: Path Completion for Session 2.



Session #	First Time	Last Time	End Time	Page View	Referring File	Inferred
1	1/21/2000:15:04:41	1/21/2000:15:04:41	1/21/2000:15:05:18	1-A	www.search.edu/?key=sample	FALSE
	1/21/2000:15:05:18	1/21/2000:15:05:18	1/21//2000:15:06:01	1-C	1.html	FALSE
	1/21//2000:15:06:01	1/21//2000:15:06:01	1/21/2000:15:06:34	H	C.html	FALSE
	1/21/2000:15:06:34	1/21/2000:15:06:34	1/21/2000:15:06:36	1-C	-	TRUE
	1/21/2000:15:06:36	1/21/2000:15:06:36	-	I	C.html	FALSE
2	1/21/2000:15:07:03	1/21/2000:15:05:45	1/21/2000:15:06:06	1-A	-	FALSE
	1/21/2000:15:07:16	1/21/2000:15:06:06	1/21/2000:15:06:18	1-B	1.html	FALSE
	1/21/2000:15:07:28	1/21/2000:15:06:19	1/21/2000:15:07:24	2-F	B.html	FALSE
	1/21/2000:15:08:34	1/21/2000:15:07:24	1/21/2000:15:07:47	2-G	2.html	FALSE
	1/21/2000:15:08:57	1/21/2000:15:07:47	1/21/2000:15:07:49	2-F	-	TRUE
	1/21/2000:15:08:59	1/21/2000:15:07:49	1/21/2000:15:09:55	4-F1	F.html	FALSE
	1/21/2000:15:11:05	1/21/2000:15:09:55	1/21/2000:15:09:57	2-F	-	TRUE
	1/21/2000:15:11:07	1/21/2000:15:09:57	1/21/2000:15:09:59	2-G	2.html	TRUE
	1/21/2000:15:11:09	1/21/2000:15:10:02	-	5-G3	G.html	FALSE
3	1/21/2000:15:04:44	1/21/2000:15:04:45	1/21/2000:15:05:19	1-A	www.search.edu/?key=sample	FALSE
	1/21/2000:15:05:19	1/21/2000:15:05:19	-	D	A.html	FALSE

Figure 5.10: Results of Path Completion Heuristic.

ber of definitions for episodes can be manually created for a Web site based on the content. For example, episodes of only the product pages viewed on an e-commerce site, or sports page views from a news site can be created. Another popular episode definition is the subset of page views that deal with the "shopping cart checkout" process[5]. However, if some assumptions are made about the way users browse a Web site, semantically meaningful episodes can be identified without any knowledge of the specific content of a site. Three methods based on such assumptions, *page type*, *reference length* and *maximal forward reference*, make an attempt to identify semantically meaningful episodes based on the page usage type.

5.2.1 General Episode Model

Using the concept of auxiliary and media page references, there are two ways to define episodes, as shown in Fig. 5.11, where auxiliary and media page references are labeled with an A or M respectively. The first would be to define an episode as all of the auxiliary references up to and including each media reference for a given user.

[5] E-commerce sites often use separate definitive tracking mechanisms available through commercial content servers such as [6] for tracking critical events such as this.

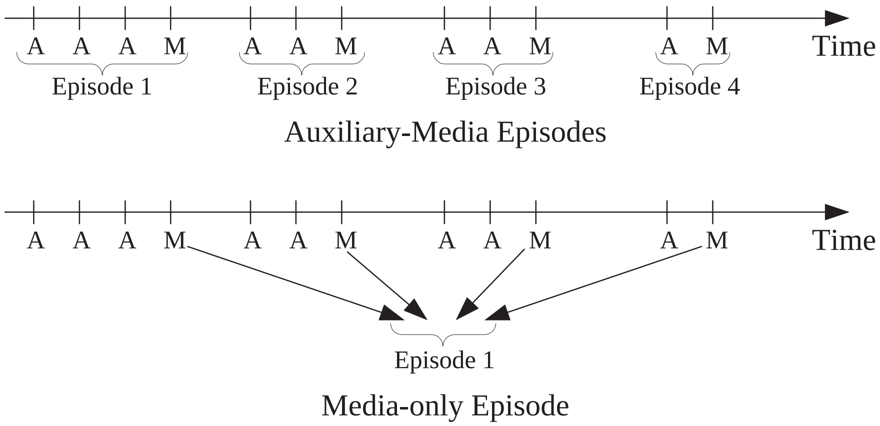
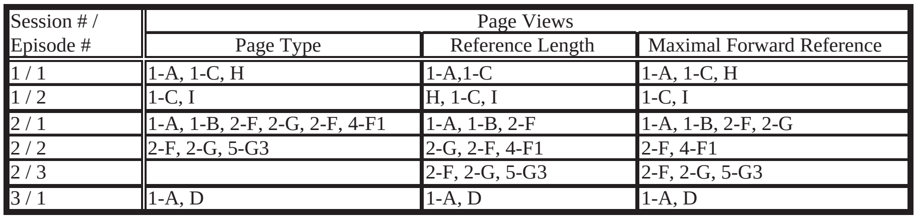


Figure 5.11: Auxiliary-Media and Media-Only Episode Types

Mining these *auxiliary-media* episodes would essentially give the common traversal paths through the web site to a given media page. The second method would be to define an episode as all of the media references for a given user. Mining these *media-only* episodes would give associations between the media pages of a site, without any information as to the path taken between the pages. Media pages can be identified by examining the content, structure, or usage of a site. The *page type* method classifies pages based on the content. The *reference length* and *maximal forward reference* methods rely on usage data to identify the usage type of a page. While none of the methods relies on the structure of a site, as will be shown in Chapter 8, the *maximal forward reference* method is heavily dependent on the structure of a Web site, even though its classification is based on usage patterns.

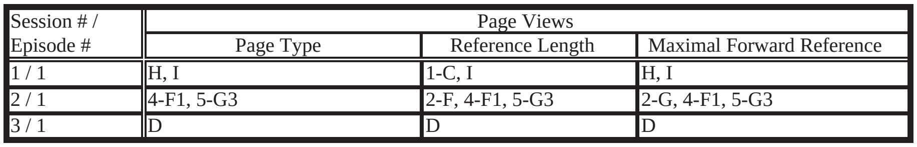
5.2.2 Page Type Method

The page type episode identification method relies on the page views being pre-classified for usage type. As discussed in Chapter 4, The usage type for a page can be labeled by hand, or a classification algorithm can be used based on the content of the



Session # /	Page Views		
Episode #	Page Type	Reference Length	Maximal Forward Reference
1 / 1	1-A, 1-C, H	1-A,1-C	1-A, 1-C, H
1 / 2	1-C, I	H, 1-C, I	1-C, I
2 / 1	1-A, 1-B, 2-F, 2-G, 2-F, 4-F1	1-A, 1-B, 2-F	1-A, 1-B, 2-F, 2-G
2 / 2	2-F, 2-G, 5-G3	2-G, 2-F, 4-F1	2-F, 4-F1
2 / 3		2-F, 2-G, 5-G3	2-F, 2-G, 5-G3
3 / 1	1-A, D	1-A, D	1-A, D

Figure 5.12: Auxiliary-media Episodes



Session # /	Page Views		
Episode #	Page Type	Reference Length	Maximal Forward Reference
1 / 1	H, I	1-C, I	H, I
2 / 1	4-F1, 5-G3	2-F, 4-F1, 5-G3	2-G, 4-F1, 5-G3
3 / 1	D	D	D

Figure 5.13: Media-only Episodes

pages to automatically label each page. With this method, the label for each page is constant across all users. This creates a problem for the page views that are a mix of media, and one of the auxiliary usage types such as navigation. For the purposes of episode identification, each page view must be declared as media, or auxiliary.

For the example site and log in Figures 4.2 and 5.2, the page views that contain a pure media page in the *main* frame will be classified as a *media* page view and all other combinations (navigation or mixed) will be classified as *auxiliary* page views. This leads to the *auxiliary-media* episodes shown in Figure 5.12 and the *media-only* episodes shown in Figure 5.13.

5.2.3 Reference Length Method

The reference length episode identification method is based on the assumption that the amount of time a user spends on a page correlates to whether the page should be classified as a auxiliary or media page for that user. Figure 5.14 shows a histogram of the lengths of page file references between 0 and 600 seconds for a server log from the Global Reach Internet Productions (GRIP) Web site [2].

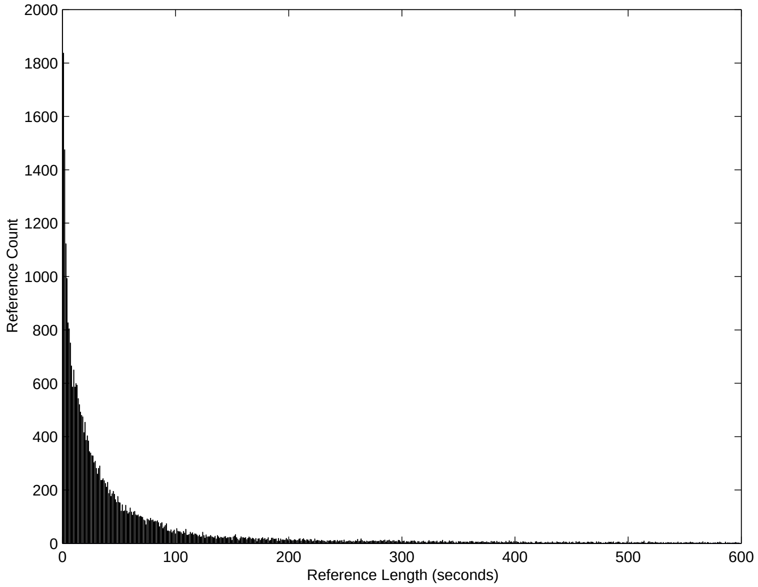


Figure 5.14: Web Page File Reference Lengths (seconds)

Qualitative analysis of several other server logs reveals that like Fig. 5.14, the shape of the histogram has a large exponential component. However, when page views are taken into account, instead of page files, the shape of the curve is slightly different, as shown in Figure 5.15. This histogram was generated from an e-commerce site with three frames per page view. The difference between the two curves can be accounted for by the removal of the reference lengths that belong to frame components that are requested immediately before another component for the same page view. This can be seen in the example log for all of the page file requests for file `1.html`. The second page view component, file `A.html`, is always requested immediately after `1.html`, giving the appearance of a very short view length. Once `1.html` and A.html are rolled into the same page view, the short reference lengths are absorbed by the page view reference lengths.

All of the curves can be reasonably fit to a heavy tailed gamma distribution.

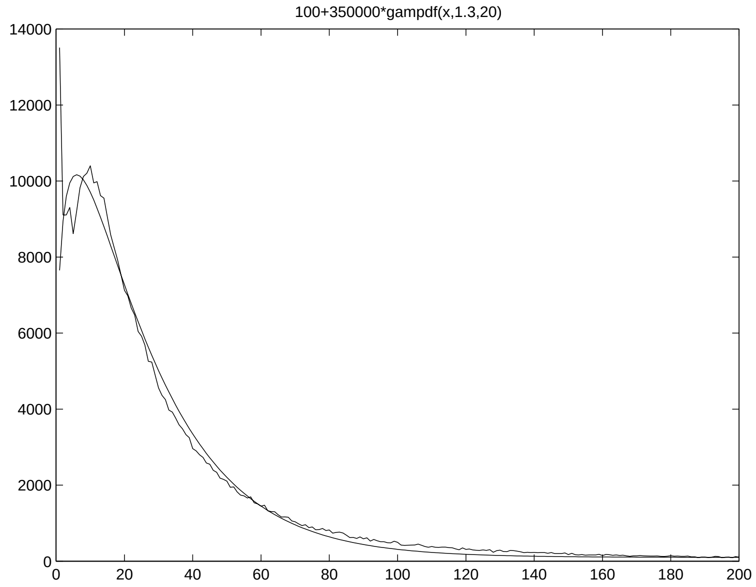


Figure 5.15: Web Page View Reference Lengths (seconds)

The difficulty comes in determining the shape parameters. As shown in Figure 5.15, shape parameters of 1.3 and 20, with an offset of 100 to create the heavy tail, fits this particular Web site quite well. However, the parameters are not constant across all Web sites. Since the reference length method ultimately relies on estimates, a pure exponential distribution will be used to model the view length curves. Although an exponential distribution does not fit as well as a gamma distribution with the appropriate shape parameters, it can be applied uniformly without any knowledge of a particular Web site.

It is expected that the variance of the times spent on the auxiliary pages is small, and the auxiliary references make up the lower end of the curve. The length of media references is expected to have a wide variance and would make up the upper tail that extends out to the longest reference. If an assumption is made about the percentage of auxiliary references in a log, a reference length can be calculated that estimates the cutoff between auxiliary and media references. Specifically, given a percent of auxiliary references, the reference length method uses a maximum likelihood estimate to calculate the time length `t` as shown in (5.4).

$$t = \tfrac{-ln(1-\gamma)}{\lambda}$$

$$\text{where } \gamma = \text{ \% of auxiliary references,}$$

$$\lambda = \text{ reciprocal of observed mean reference length.} \qquad (5.4)$$

The definition of (5.4) comes from integrating the formula for an exponential distribution, from γ to zero. The maximum likelihood estimate for the exponential distribution is the observed mean. The time length `t` could be calculated exactly by sorting all of the reference lengths from a log and then selecting the reference length that is located in the $\gamma\times$`log_size` position. However, this increases the complexity of the algorithm from linear to O(nlogn) while not necessarily increasing the accuracy of the calculation since the value of γ is only an estimate. Although, as mentioned, the exponential distribution does not fit the histograms of server log data exactly, it provides a reasonable estimate of the cutoff reference length. It would be interesting to examine the cost-benefit tradeoffs for fitting gamma distributions to a particular site and using the more accurate cutoff times.

The reference length approach makes the assumption that all of the last references are media references, and ignores them while calculating the cutoff time. This assumption can introduce errors if a specific auxiliary page is commonly used as the exit point for a Web site. While interruptions such as a phone call or lunch break can result in the erroneous classification of a auxiliary reference as a media reference, it is unlikely that the error will occur on a regular basis for the same page. A reasonable minimum support threshold during the application of a data mining algorithm is expected to weed out these errors.

Once the cutoff time is calculated, either auxiliary-media episodes or media-only episodes can be identified. It is important to note that unlike the *page type* method, results generated with the *reference length* method only apply when those pages are used as media references. For example, an association rule, $\mathtt{A} \Longrightarrow \mathtt{B}$, is normally taken to mean that when `A` is in a set, so is `B`. However, if this rule has been generated with the reference length method, it has a more specific meaning, namely that `A` implies `B` only when both `A` and `B` are used as media references. This property allows data mining on media-only episodes to produce rules that might be missed by including all of the page references in a log. If users that treat page `A` as an auxiliary page do not generally go on to page `B`, inclusion of the auxiliary references into the data mining process would reduce the confidence of the rule $\mathtt{A} \Longrightarrow \mathtt{B}$, possibly to the point where it is not reported. Depending on the goals of the analysis, this can be seen as an advantage or a disadvantage. The key is that auxiliary-media episodes can be used when this property is undesirable.

The one parameter that the reference length approach requires is an estimation of the overall percentage of references that are auxiliary. The estimation of the percentage of auxiliary references can be based on the structure and media of the site or experience of the data analyst with other server logs. The results presented in Chapter 8 show that the approach is fairly robust and a wide range of auxiliary percentages will yield reasonable sets of association rules.

Again, based on the example log, two sets of episodes can be identified with the reference length method. An estimate for the auxiliary reference percentage can be taken from the results of the page type method, which in this case leads to a γ of 68%. The cutoff value, t is then calculated to be 35.2 seconds. Due to the small number of entries in the example, the cutoff calculated by the reference length method is not statistically valid, and varies significantly with the choice of γ. A γ of 0.6 gives a t

of 28.3 seconds and a γ of 0.75 gives a t of 42.9 seconds. As shown in Figures 5.12 and 5.13, the episodes are slightly different than the ones identified with the page type method. However, these episodes are based on how the users are *actually* using the site, as opposed to how the designers *expect* the site to be used.

5.2.4 Maximal Forward Reference Method

The maximal forward reference episode identification approach is based on the work presented in [35]. Instead of time spent on a page, each episode is defined to be the set of pages in the path from the first page in a user session up to the page before a backward reference is made. A forward reference is defined to be a page not already in the set of pages for the current episode. Similarly, a backward reference is defined to be a page that is already contained in the set of pages for the current episode. A new episode is started when the next forward reference is made. The underlying model for this approach is that the maximal forward reference pages are the media pages, and the pages leading up to each maximal forward reference are the auxiliary pages. Like the reference length approach, two sets of episodes, namely auxiliary-media or media-only, can be formed. The maximal forward reference approach has an advantage over the reference length in that it does not require an input parameter that is based on an assumption about the characteristics of a particular set of data.

The results of running the maximal forward reference method are also shown in Figures 5.12 and 5.13. The resulting episodes are slightly different from both the page type and reference length methods. The single episode for user 2 is the same in all cases, but the episodes for the first and third users get divided up differently. As will be shown in Chapter 8, as the connectivity of a Web site increases, the maximal forward reference method tends to break down since a backward reference is less likely.

Chapter 6

Pattern Discovery

Pattern discovery draws upon methods and algorithms developed from several fields such as statistics, machine learning and pattern recognition. As discussed in Chapter 2, a wide variety of pattern discovery techniques have been applied to Web usage data. The first part of this chapter is an extension of Chapter 2, and gives a brief overview of some of the more popular methods that have been used to discover patterns for Web Usage Mining. In addition, the specific details for the frequent itemset discovery, clustering, and classification algorithms used in this thesis are given.

6.1 Overview

Once the preprocessing tasks described in Chapters 4 and 5 have been performed, the number of pattern discovery techniques that can be applied to the usage data is almost unlimited. The methods range in complexity from relatively simple techniques such as *statistical analysis* to more computationally expensive methods such as *hidden markov modeling*. In practice, several techniques are usually applied to a set of usage data in order to form a well rounded picture of how a Web site is being used.

6.1.1 Statistical Analysis

Statistical techniques are the most common method to extract knowledge about visitors to a Web site [3, 5, 8, 10]. By analyzing the session file, one can perform different kinds of descriptive statistical analysis (frequency, mean, median, etc.) on variables such as page views, viewing time and length of a navigation path. Many Web traffic analysis tools produce a periodic report containing statistical information such as the most frequently accessed pages, average view time of a page or average length of a path through a site. This report may include limited low-level error analysis such as detecting unauthorized entry points or finding the most common invalid URI. Despite lacking in the depth of its analysis, this type of knowledge can be potentially useful for improving system performance, enhancing system security, facilitating site modification, and providing support for marketing decisions.

6.1.2 Frequent Itemsets and Association Rules

Frequent Itemset discovery can be used to relate pages that are most often referenced together in a single server session. Examples of frequent itemsets are as follows:

- The Home Page and Shopping Cart Page are accessed together in 20% of the sessions.
- The Donkey Kong Video Game and Stainless Steel Flatware Set product pages are accessed together in 1.2% of the sessions.

Any set of `n` frequent items can further broken into `n` separate *association rules*, where a directionality is added to the rule. A frequent itemset of pages `A` and `B` leads to two association rules of $\mathtt{A} \Rightarrow \mathtt{B}$ and $\mathtt{B} \Rightarrow \mathtt{A}$. The first frequent itemset example listed above now becomes:

- When the Shopping Cart Page is accessed in a session, the Home Page is also accessed 95% of the time.
- When the Home Page is accessed in a session, the Shopping Cart Page is also accessed 20% of the time.

In the context of Web Usage Mining, frequent itemsets and association rules refer to sets of pages that are accessed together with a support value exceeding some specified threshold. These pages may or may not be directly connected to one another via hyperlinks. For example, rule discovery using the Apriori algorithm [20] may reveal a correlation between users who visited a page containing electronic products to those who access a page about sporting equipment. This is useful for identifying cross-promotional opportunities. Aside from being applicable for business and marketing applications, the presence or absence of such rules can help Web designers to restructure their Web site. The rules may also serve as a heuristic for prefetching documents in order to reduce user-perceived latency when loading a page from a remote site.

6.1.3 Clustering

Clustering is a technique to group together a set of items having similar characteristics [63, 77]. In the Web Usage domain, the two most common types of clusters that are discovered are user clusters and page clusters. Clustering of users tends to establish groups of users exhibiting similar browsing patterns, for example:

- Users in cluster "2" for a news site access articles related to sports and finance.

Such knowledge has been used to provide personalized Web content to the users, or infer user demographics in order to perform market segmentation in E-commerce applications[1]. On the other hand, clustering of pages can discover groups of pages having related content:

- The Donkey Kong Video Game, Pokemon Video Game, and Video Game Caddy product pages belong to the same usage cluster.

[1] In practice, demographic information is obtained directly from user registrations, or purchased from a third party.

This information is useful for search engines and Web assistance providers. In both applications, permanent or dynamic HTML pages can be created that suggest related hyperlinks to the user according to the user's query or past history of information needs.

6.1.4 Classification

Classification is the task of mapping a data item into one of several predefined classes [51]. In the Web domain, one is often interested in developing a profile of users belonging to a particular class or category. This requires extraction and selection of features that best describe the properties of a given class or category. Classification can be done by using supervised inductive learning algorithms such as decision tree classifiers [89], naive Bayesian classifiers [73], k-nearest neighbor classifiers [52], or Support Vector Machines [29, 107]. For example, classification of usage data coupled with registration data may lead to the discovery of patterns such as:

- 30% of users who placed an online order from `/Product/Music` are in the 18-25 age group and live on the West Coast.
- The Donkey Kong Video Game, Pokemon Video Game, and Video Game Caddy product pages are all part of the Video Games product group.

6.1.5 Sequential Patterns

The technique of sequential pattern discovery [70, 105] attempts to find inter-session patterns such that the presence of a set of items is followed by another item in a time-ordered set of sessions or episodes. For example:

- The Video Game Caddy page view is accessed after the Donkey Kong Video Game page view 50% of the time.

By using this approach, Web marketers can predict future visit patterns which will be helpful in placing advertisements aimed at certain user groups. Other types of temporal analysis that can be performed on sequential patterns includes trend analysis [21] or change point detection [54]. Trend analysis can be used to detect changes in the usage patterns of a site over time, and change point detection identifies when specific changes take place. For example:

- Page views for the Donkey Kong Video Game have been decreasing over the last two quarters.
- The Donkey Kong Video Game page views increased from January through March, were steady until October, then began to drop.

6.1.6 Dependency Modeling

Dependency modeling is another useful pattern discovery task in Web Mining. The goal here is to develop a model capable of representing significant dependencies among the various variables in the Web domain. As an example, one may be interested to build a model representing the different stages a visitor undergoes while shopping in an online store based on the actions chosen (ie. from a casual visitor to a serious potential buyer). There are several probabilistic learning techniques that can be employed to model the browsing behavior of users. Such techniques include Hidden Markov Models [26] and Bayesian Belief Networks [80]. Modeling of Web usage patterns will not only provide a theoretical framework for analyzing the behavior of users but is potentially useful for predicting future Web resource consumption. Such information may help develop strategies to increase the sales of products offered by the Web site or improve the navigational convenience of users.

6.2 Algorithms

The usage patterns that will be presented in Chapter 8, as well structure and content preprocessing discussed in Chapter 4 make use of several pattern discovery algorithms.

While the details of preparing text for clustering and classification were discussed in Chapter 4, there are several options involving page views and page files when preparing for pattern discovery.

For a multi-framed site, patterns can be discovered based on page views, the individual page files, or the set of page files for a particular frame location. Page view based patterns prevent the problem of discovering patterns that simply represent the different components of the same page view. For example, the “Home Page” view may be made up of two page files, `HomeLeft.html` and `HomeBody.html`. Performing pattern discovery with page view identifiers such as “Home Page” prevents cluttering the results with associations between `HomeLeft.html` and `HomeBody.html`. However, in the case where a page file in one frame may be associated with several page files in another frame, page view pattern discovery has two potential problems. Because the occurrences of a page file may be spread across several page views, each page view with the page file may not appear enough times to meet a minimum support criteria, whereas the individual page file may have sufficient support. Second, patterns can be discovered that only contain page views with the same page file. For example, consider a product page file that can be viewed along with ten different left navigation page files. This means that there are ten page views that differ only by the left navigation frame. The product page may appear in 10% of the sessions, but when split among the ten page views, may never appear in more than 1% of the sessions. Even if all of the page views meet the minimum support, it is likely that a pattern relating all ten of the page views will be discovered. This is not much better than discovering patterns of page view components.

A third option is to perform pattern discovery on page files from a single frame, such as the “main” or ”body” frame. This prevents the discovery of page view components in addition to preventing the dilution of a page file among multiple page views. The problem with patterns discovered for a single frame location comes during the pattern analysis phase. When using a technique such as pattern filtering based on site structure, patterns that do not contain all of the page view components can be missing the links that tie the page files together. Consider a set of product pages that are all listed on the same left navigation frame. Taken individually, none of the product pages are linked to each other, but when the entire page views are considered, they are all doubly linked. The results presented in Chapter 8 will show that when filtering based on the site structure is performed, patterns based on all of the page files should be discovered. When no filtering or content filtering is used, single frame patterns provide the best results. For single frame sites, the page files and page views are equivalent, preventing any of the problems discussed above.

6.2.1 Frequent Itemsets

All of the information contained in the server sessions or episodes is not necessary for frequent itemset discovery. The order and number of occurrences of a page view or page file in a session is not required. Therefore, sessions must be stripped down to a list of unique session ID/page pairs. A minimum support cutoff must be identified in order to limit the number of discovered patterns and the computation time. The support of an itemset is the fraction of the total sessions that the set appears in together. Support, S, is defined as follows for a set of n items, where D is the data set and i is an item:

$$S = \frac{count(\{i_1, \ldots, i_n\} \in D)}{count(D)} \tag{6.1}$$

Once the frequent itemsets are discovered, *interest* can be used to objectively rank the sets. Interest [31] is defined as the *support* of a frequent itemset divided by the probability of all of the items appearing together in a set if the items are randomly and independently distributed:

$$I = \frac{S(i_1, \ldots, i_n)}{\prod_{j=1}^{n} S(i_j)} \tag{6.2}$$

An interest measure that is greater than one indicates that the items in the itemset appear together more often than what would be expected through a random distribution. Items that have very high support will often appear in frequent itemsets simply because they randomly appear together in sessions. However, these itemsets tend to have a low interest measure. If the itemsets are broken out into association rules, the confidence of each rule can be calculated. The confidence of a rule is the fraction of sessions where the subsequent is present if the antecedent is also present. Typically, association rules are limited to single item antecedents (otherwise, the number of rules for an `n` item itemset can be much greater than `n`). Confidence is defined as follows for a rule $i_a \Rightarrow \{i_{s1}, \ldots, i_{sn}\}$:

$$C = \frac{S(i_a, i_{s1}, \ldots, i_{sn})}{S(i_a)} \tag{6.3}$$

The support, confidence, and interest measures will be used for the generation of experimental results in Chapter 8.

6.2.2 Clustering

The clustering results obtained in Chapter 8 were obtained using *hypergraph clustering* of frequent itemsets. The hypergraph clustering method of [55] takes a set of association rules and declares the items in the rules to be vertices, and the rules themselves to be hyperedges. Since association rules have a directionality associated with each rule, the algorithm combines all rules with the same set of items, and uses an average of the confidence of the individual rules as the weight for a hyperedge.

Clusters can be quickly found by using a hypergraph partitioning algorithm such as hMETIS [61].

The hypergraph clustering algorithm described in [55] was originally adapted in several ways in the TopCAT (Top Clusters and Associations from Text) system [38] to fit the unstructured text domain. However, the TopCAT modifications have been incorporated into the WebSIFT system and have been used to successfully discover clusters of Web pages and users. Because TopCAT discovers frequent itemsets instead of association rules, the rules do not have any directionality and therefore do not need to be combined prior to being used in a hypergraph. The interest of each itemset used for the weight of each edge. Since interest tends to increase dramatically as the number of items in a frequent itemset increases, the log of the interest is used in the clustering algorithm to prevent the larger itemsets from completely dominating the process.

It was found that the stopping criteria presented in [55] only works for domains that form very highly connected hypergraphs. The [55] algorithm continues to recursively partition a hypergraph until the weight of the edges cut compared to the weight of the edges left in either partition falls below a set ratio (referred to as *fitness*). This criteria has two fundamental problems:

- It will never divide a loosely connected hypergraph into the appropriate number of clusters. This is because it stops *as soon as* if finds a partition that meets the fitness criteria.
- It can inappropriately partition a group of items that should be left together. If the initial hypergraph is a group of items that logically belong to a single cluster, the algorithm will go ahead and partition the items anyway.

In order to solve these problems, a user-defined *cutoff* parameter is used which represents the maximum allowable cut-weight ratio (the weight of the cut edges divided by the weight of the uncut edges in a given partition). The cut-weight ratio is defined as follows. Let P be a partition with a set of m edges e, and c the set of n edges cut in the previous split of the hypergraph:

$$cw(P) = \frac{\Sigma_{i=1}^{n} weight(c_i)}{\Sigma_{j=1}^{m} weight(e_j)} \tag{6.4}$$

A hyperedge remains in a partition if two or more vertices from the original edge are in the partition. For example, a cut-weight ratio of 0.5 means that the weight of the cut edges is half of the weight of the remaining edges. The algorithm assumes that natural clusters will be highly connected by edges. Therefore, a low cut-weight ratio indicates that hMETIS made what should be a natural split between the vertices in the hypergraph. A high cut-weight ratio indicates that the hypergraph was a natural cluster of items and should not have been split. Once the stopping criteria has been reached, vertices can be “added back in” to clusters if they are contained in an edge that “overlaps” to a significant degree with the vertices in the cluster. The minimum amount of overlap required is defined by the user. This allows items to appear in multiple clusters. Finally, the *connectivity* of each item in the cluster is checked. Connectivity measures the strength of the item’s connection to the other items in the cluster, and is defined as follows, where x is number of edges containing v, and y is the total number of edges in the cluster:

$$cn(v) = \frac{\sum_{i=1}^{x} weight(e_i)}{\sum_{j=1}^{y} weight(e_j)} \tag{6.5}$$

Items with a low connectivity are removed from the cluster. It should be noted that like the clustering methods of [82], every item does not necessarily end up in a cluster. Items that are not represented in a frequent itemset and items that do not meet the connectivity requirement do not appear in any cluster. This usually results in very “clean” clusters since items that are not particularly related to anything else are not forced to appear in a cluster. However, if clusters of all of the items in a data set are desired, a different clustering method, such as k-means, should be used.

6.2.3 Classification

For classification problems with a high number of dimensions such as Web usage data or text, Support Vector Machines can be used. The Support Vector Machine (SVM) is a learning method introduced by Vapnik based on his statistical learning theory [107] and Structural Risk Minimization principle. SVMs use the Vapnik-Chervonenkis (VC) dimension of a problem to characterize its complexity, which can be independent of the dimensionality of the problem. When using SVMs for classification, the basic idea is to find the optimal separating hyperplane between the positive and negative examples. The optimal hyperplane is defined as the one giving the maximum margin between the training examples that are closest to the hyperplane. The group of examples (vectors) that lie closest to the separating hyperplane are referred to as support vectors. Once the separating hyperplane is found, new examples can be classified simply by checking which side of the hyperplane they fall on. A simple linearly separable example is shown in Figure 6.1, where there are only two dimensions. The dotted lines represent the limits of linear planes that would correctly classify the training examples. The bold line labeled with an "**h**" represents the plane that provides the maximum separation between the two classes. The three support vectors (shaded training examples) in the figure can be used to completely define **h**. SVMs are referred to as *universal learners*, meaning that various types of representations such as polynomials, radial basis functions, or neural networks can be used.

The Support Vector Machine combines four concepts, as described in detail in [36]:

- Use of the Structural Risk Minimization (SRM) inductive principle.
- Mapping of input samples onto a high-dimensional space through the use of a set of predefined nonlinear basis functions.

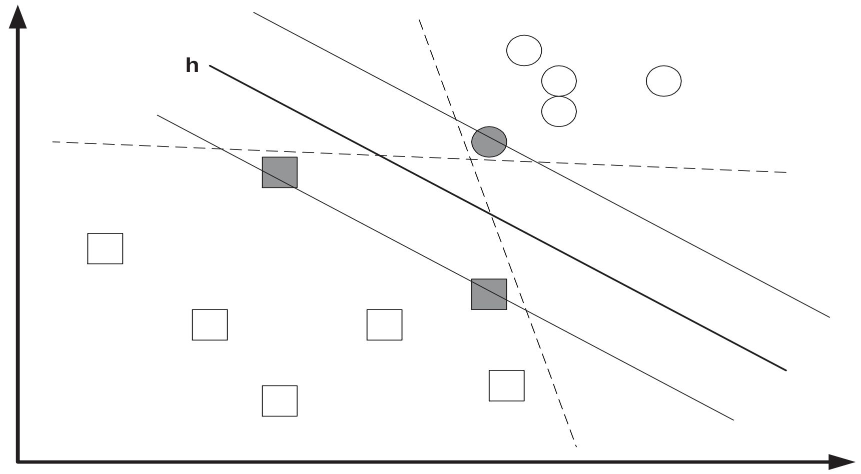


Figure 6.1: Support Vector Machine Classification Example

- Use of linear functions with constraints on complexity to approximate the input samples in the high-dimensional space.
- Use of the duality theory of optimization to make estimates of model parameters in a high-dimensional feature space computationally tractable.

For classification, the optimal separating hyperplane, **h** is defined by the values of x where the decision function $D(x)$ is equal to zero. Given a set of training data (x_i, y_i) from an input space $\mathbf{x}$, the decision function is defined as:

$$D(x) = \sum_{i=1}^{n} \alpha_i^* y_i H(x_i, x) \tag{6.6}$$

The α_i^*'s are the maximal solution to the following quadratic optimization problem:

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j H(x_i, x_j) \tag{6.7}$$

Subject to the constraints:

$$\sum_{i=1}^{n} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq \frac{C}{n}, \; i = 1, \ldots, n \tag{6.8}$$

where H is the inner product kernel and C is the regularization parameter. For polynomials of degree q, the inner product kernel is equal to:

$$H(x, x') = [(x \cdot x') + 1]^q \tag{6.9}$$

For radial basis functions of the form:

$$f(x) = \sin(\sum_{i=1}^{n} \alpha_i \exp\{-\frac{|x - x_i|^2}{\sigma^2}\}) \tag{6.10}$$

where σ is equal to width, the inner product kernel is equal to:

$$H(x, x') = \exp\{-\frac{|x - x'|^2}{\sigma^2}\} \tag{6.11}$$

Because of the separation of model complexity and dimensionality, it has been shown [60, 40] that SVMs will perform well relative to other learning methods in domains that have inherently high dimensionality. A more complete discussion of SVMs can be found in [36].

Chapter 7

Pattern Analysis

The ultimate goal of any data mining effort is to provide the analyst with insight into a particular domain. This insight can come from directly inspecting the entire set of discovered patterns, or by performing another round of aggregation, filtering or analysis on the discovered patterns. Pattern analysis is the step that is designed to convert discovered rules, patterns, and statistics into *knowledge* or insight involving the Web site being analyzed. The major difficulty in pattern analysis lies in the attempt to precisely define the term *knowledge*. The transformation from information to understanding is virtually impossible to quantify since a large component relies on the human being (or group of humans) that is performing the analysis. A simple query mechanism may suffice for a given data set and analyst, but under other circumstances, sophisticated visualization tools such as a starfield display [57] may be required. Therefore, from a system design perspective, the pattern analysis phase of any data mining task is really one of providing tools to facilitate the transformation of information into knowledge.

Just about any tool or filter that can be applied to the output of data mining algorithms can be classified as a pattern analysis tool. The most common tool is a query mechanism, such as SQL for relational databases, that allows an analyst to manually filter, sort, and combine various pieces of data to suit his or her needs. On-line Analytical Processing (OLAP) tools take queries one step further by precomputing several levels of data aggregation in advance. This allows the analyst to *roll-up* and *drill-down* data in real-time. Visualization of results through multi-colored graphs and charts is another popular way of providing pattern analysis capabilities.

With Web Usage Mining, as with many data mining domains, one of the biggest challenges in pattern analysis is separating the *interesting* results from those that are not particularly useful. Since pattern discovery algorithms are generally set up to find the strongest or most common patterns in a particular data set, the most highly ranked results are often the most obvious. As briefly mentioned in the Introduction and Chapter 2, the concept of interestingness generally involves patterns that were previously unknown to the analyst. In order to ensure that all patterns of potential interest are discovered, the thresholds of many pattern discovery algorithms must be set low enough such that hundreds or thousands of patterns are output. While the number of items to analyze can been reduced by several orders of magnitude (a server log for a large e-commerce site can contain millions of page views per day), a list of several thousand patterns is only marginally more useful than the raw data. This is especially true when the ranking of the results has no bearing on its interestingness. The rest of this chapter focuses on methods and tools that can be used to identify interesting patterns.

7.1 Measures of Interestingness

The goal of the analysis drives the choice of what interestingness measures will be appropriate. A marketing analyst may want to examine the most popular patterns to gain insight into what is driving sales on an e-commerce site. However, an analyst trying to detect improper accesses to a Web site may be looking for certain types of patterns, or patterns that don't meet the criteria for "normal." The objective measures are often tied to a particular type of pattern or even a specific algorithm, and often highlight the patterns that are *least* interesting to an analyst. Support and confidence identify the rules that are the most common and have the highest correlation strength. While these measures are critical for limiting the number of discovered patterns, they do not necessarily help the analyst to identify unusual patterns. Because these objective measures can not take any domain knowledge into account, a common problem with discovered patterns is that they simply corroborate the obvious. In order to identify potentially interesting patterns for a particular domain, a pattern-form independent, single pattern, subjective interestingness framework is needed. The final dimension that must be specified for the interestingness framework, according to [56], is the representation. In order to choose a representation, the next section defines the properties that a framework for identifying subjective interestingness should have in order to handle the types of data encountered in Web Usage Mining.

7.1.1 Desirable Properties for an Interestingness Framework

In order to create a list of desirable properties for an interestingness measure, the possible types of data (evidence) and the goals of the framework need to be defined. There are two major types of evidence that need to be handled:

- Frequency data - The mined usage patterns.
- Subjective data - Encoding of structure or content, plus any additional analyst beliefs.

The subjective evidence can be further broken down into probabilistic, fuzzy, and qualitative categories. An example of probabilistic subjective evidence would be an analysts belief that pages `A` and `B` will be used together 70% of the time. Beliefs about the usage type of a page that has some navigational and media qualities naturally maps to a fuzzy set definition, with a probability assigned to each potential usage type. Finally, evidence from humans often comes in a qualitative format, such as "I believe pages A and B will be used together *most of the time*." Given a set of default beliefs derived from the structure, content, and domain experts, plus the evidence from discovered patterns, the goals of the framework are to:

- Identify mined patterns that *conflict* with an existing belief.
- Identify mined patterns that reduce the level of ignorance about a belief.
- Identify changes in the evidence for a belief over time.

In addition, the framework should update the existing belief set based on the new evidence when *soft beliefs* are involved. [101] divides beliefs into two categories - *hard* and *soft*. A hard belief is a constraint that should not change, even with the introduction of conflicting evidence. If the evidence conflicts with a hard belief, then there must have been an error in the collection of the evidence. On the other hand, a soft belief is one that an analyst is willing to revise in the face of new and possibly conflicting evidence. The strength of a soft belief is referred to as the *degree* of the belief. Therefore, in addition to a representation, an interestingness framework must have a mechanism for updating the belief set in the face of new evidence. The task of comparing and combining evidence from multiple sources has been studied extensively under the heading of *reasoning with uncertainty* or *evidential reasoning*. The research efforts can be very broadly classified into *quantitative* and *qualitative* systems. Qualitative systems such as non-monotonic logics and Truth Maintenance systems [49] can not easily handle the large amounts of frequency data that is generated by data mining and will not be considered further. With the goals and types of evidence listed above, a framework should have the following properties:

1. Rankable interestingness measure.
2. Update functions for incorporating new evidence.
3. Explicit measure of ignorance.

4. Explicit evidence for and against a belief.

5. Ability to handle frequency based evidence.

6. Ability to handle subjective evidence.

7. Ability to handle fuzzy sets.

The choice of a representation greatly affects the choice of mechanism for evidence comparison and aggregation, however, they are not completely dependent. [101] briefly lists five different methods for handling degrees of soft beliefs with the foundation and mechanism for each method treated as a single dimension. As with any complex domain, the number of dimensions that can be used to classify the various quantitative uncertain reasoning systems is highly variable - Bonissone [27] uses 14, Quinlan [88] uses 4, and others, such as Shafer [98] and Pearl [80] tend to focus on a single defining dimension. In addition, the dimensions used are not necessarily orthogonal and the definitions are not always consistent from one author to the next. Bonissone [27] divides his 14 dimensions into three *layers* for defining a framework - a *representation* layer, *inference*(i.e. mechanism) layer, and *control* layer. Once a representation is chosen, the control layer can choose the appropriate inference mechanism for the task at hand. This general architecture is very flexible, and will be adopted for the Web Usage Mining interestingness framework, which will be referred to as the *information filter*.

For quantitative systems, representations are generally either single-valued (e.g. probability) or two-valued (e.g. belief pairs). Bayesian probability inference is by far the overwhelming choice for single-valued representations. For two-valued representations, either probability bounds or Dempster-Shafer [96] inference are the most common. The choice of the inference mechanism affects how different pieces of evidence are combined and aggregated, and how conflicts are resolved. In order to

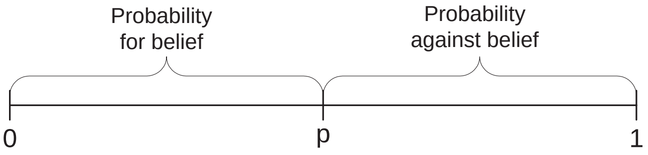


Figure 7.1: Bayesian Concepts

discuss the relative merits of the various representations and inference mechanisms, a discussion of the basic properties of each is warranted.

7.1.2 Bayesian Inference

The main tenet of any Bayesian system is that a probability between the values of zero and one can be assigned to any belief. By definition, the probability against a belief, $\mathcal{B}$, is equal to the probability not assigned in favor of the belief:

$$P(\mathcal{B}) = x, \quad P(\neg\mathcal{B}) = 1 - x, \quad 0 \leq x \leq 1 \tag{7.1}$$

When new evidence, e is obtained about a belief, the probability is updated using Bayes rule:

$$P(\mathcal{B}|e) = \frac{P(e|\mathcal{B})P(\mathcal{B})}{P(e)} \tag{7.2}$$

The new probability is referred to as being *conditioned* by the evidence, e. In order make use of Bayesian methods, both prior probabilities for all beliefs in addition to the probability of the occurrence of a particular piece of evidence are required. When nothing is known in advance about a particular belief, a default value such as 0.5 is assigned. Figure 7.1 shows the concepts involved in Bayesian inference.

7.1.3 Probability-Bound Inference

The idea of probability bounds is to allow for a range of probabilities to be considered simultaneously. Often, the prior probability for a particular belief is not known, and instead of picking a single value such as 0.5, a two-valued representation such as probability bounds allows an explicit statement of ignorance. For a belief $\mathcal{B}$, evidence collected for or against $\mathcal{B}$ can be used to form the *probability bounds*, $[l, u]$, where:

$$\begin{aligned} l &= \text{lower bound on the probability of support for } \mathcal{B} && (7.3) \\ u &= \text{upper bound on the probability of support for } \mathcal{B} && (7.4) \end{aligned}$$

The values of l and u must satisfy the constraints:

$$l + (1 - u) \leq 1, \quad l \geq 0, \quad u \geq 0 \qquad (7.5)$$

In this case, a lower and upper bound of 0 and 1 respectively would indicate that the probability for a certain belief could be anywhere in the full range of possible probabilities. There have been several methods proposed for combining different sets of probability bounds obtained from different sets of evidence about the same belief. Bonissone [27] defines five different triangular norms (T-norms) as follows for combining two sets of evidence:

$$\begin{aligned} T_1(a,b) &= \max(0, a + b - 1) && (7.6) \\ T_{1.5}(a,b) &= \begin{cases} (a^{0.5} + b^{0.5} - 1)^2 & \text{if } (a^{0.5} + b^{0.5}) \geq 1 \\ 0 \text{ otherwise} \end{cases} && (7.7) \\ T_2(a,b) &= ab && (7.8) \\ T_{2.5}(a,b) &= (a^{-1} + b^{-1} - 1)^{-1} && (7.9) \\ T_3(a,b) &= \min(a,b) && (7.10) \end{aligned}$$

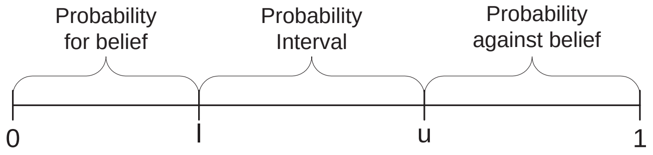


Figure 7.2: Probability-Bound Concepts

For two sets of evidence, $[l_1, u_1]$ and $[l_2, u_2]$, the combined evidence is equal to the following:

$$[T_i(l_1, l_2), S_i(u_1, u_2)] \tag{7.11}$$

where S_i is the conorm defined as:

$$S_i(a, b) = 1 - T_i(1 - a, 1 - b) \tag{7.12}$$

The different norms range from conservative combination techniques to liberal combination techniques as i increases from 1 to 3. In addition to Bonissone's norms and conorms, Baldwin has identified a few other combination techniques [25]. The liberal T_3 norm is equivalent to Zadeh's intersection rule [111], and is also used by Baldwin for combining groups of evidence from the same source prior to reasoning. The basic concepts involved with probability-bound inference are shown in Figure 7.2.

7.1.4 Dempster-Shafer Inference

Dempster-Shafer (DS) methods also use a two-valued measure to describe the degree of belief for or against a given statement.For a belief $\mathcal{B}$, evidence collected for or against $\mathcal{B}$ can be used to form an *support pair*, $[s_n, s_p]$, where:

$$s_n = \text{necessary support of } \mathcal{B} \tag{7.13}$$

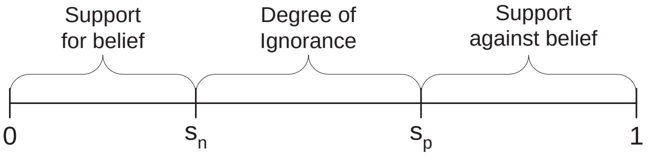


Figure 7.3: Dempster-Shafer Concepts

$$s_p = \text{possible support of } \mathcal{B} \tag{7.14}$$

$$(1 - s_p) = \text{necessary support of } \neg\mathcal{B} \tag{7.15}$$

$$(1 - s_n) = \text{possible support of } \neg\mathcal{B} \tag{7.16}$$

$$(s_p - s_n) = \text{uncertainty of } \mathcal{B} \tag{7.17}$$

The values of s_n and s_p must satisfy the constraints:

$$s_n + (1 - s_p) \leq 1 \tag{7.18}$$

$$s_n \geq 0, s_p \geq 0 \tag{7.19}$$

Figure 7.3 shows the concepts that map to each region of a belief scale, given s_n and s_p. As an example, assume that evidence has been collected about the belief B, that Web pages `A` and `C` are related. If all of the evidence is in support of B, the DS pair is $[1, 1]$. On the other extreme, if all of the evidence is against B, the DS pair is $[0, 0]$. If the data leads to a 25% degree of belief that B is true, and a 40% degree of belief that B is false, then $[0.25, 0.6]$ would represent the appropriate DS pair. This says that the degree of uncertainty about B is 35%. Finally, if there is no evidence pertaining to B, the DS pair is $[0, 1]$, giving an uncertainty of 100%. Independent of the type of the source for generating an DS pair, pairs can be combined per Dempster's rule of combination [96] to obtain a single DS pair per belief. The basic rule is as follows:

If $\mathcal{B}$:$[s_{1n}, s_{1p}]$ AND $\mathcal{B}$:$[s_{2n}, s_{2p}]$ are two independent DS pairs from different sources about belief $\mathcal{B}$, then conclude $\mathcal{B}$:$[s_n, s_p]$, where

$$k = 1 - s_{1n}(1 - s_{2p}) - s_{2n}(1 - s_{1p}) \tag{7.20}$$

$$s_n = [s_{1n}s_{2n} + s_{1n}(s_{2p} - s_{2n}) + s_{2n}(s_{1p} - s_{1n})]/k \tag{7.21}$$

$$\begin{aligned}1 - s_p = [&(1 - s_{1p})(1 - s_{2p}) + (s_{1p} - s_{1n})(1 - s_{2p}) + \\ &(s_{2p} - s_{2n})(1 - s_{1p})]/k\end{aligned} \tag{7.22}$$

In order for the use of Dempster's combination rule to be valid, the sources of evidence must be independent of each other, as stated in equations above, and the *frame of discernment* must be *exclusive* and *exhaustive.* The frame of discernment refers to the domain of conclusions on which reasoning is taking place. In this case, the frame of discernment is simply whether or not a particular belief is or is not true. Therefore, the frame of discernment is both exclusive (The choices do not overlap), and exhaustive (There are no other choices). The question of independent evidence sources is not quite as clear cut. It could be argued that the usage of a site is dependent on the structure to some degree (a user can not follow links that are not there). However, the nature of this dependence is difficult to define quantitatively. Therefore, similar to naive Bayesian methods [91], the evidence sources will be assumed to be independent.

7.1.5 Statistical Inference

Statistical inference is the use of hypothesis testing with the mean and variance of values in order to conclude if two sets of values are different. It is listed in [101] as a possible choice for the mechanism for an interestingness framework. It is not used for reasoning with uncertainty because subjective evidence rarely comes from enough sources to make the calculation of means and variances meaningful. In addition, when the goal is to infer new facts in a knowledge base and propagate changes, an attempt to carry along means and variances would create a very complex system. However, since the goals of interestingness framework include identifying conflicting evidence and changes in the discovered patterns, not to infer new facts, statistical inference should not be automatically ruled out. A statistical inference framework would use a single-value probability representation, the same as defined for Bayesian inference and shown in Figure 7.1. Interestingness can be determined by defining a null hypothesis that the existing value is equal to the new combined value, and checking for a violation of the hypothesis. Sources of evidence are combined by standard averaging techniques, so the existing and combined values are actually observed means, $\hat{\mu}$, with an associated standard deviation, $\hat{\sigma}$. The amount of change is defined as follows:

$$H_0 \quad : \quad \mu_o = \mu_c \tag{7.23}$$

$$H_A \quad : \quad \mu_o \neq \mu_c \tag{7.24}$$

$$z \quad = \quad \frac{|\hat{\mu}_o - \hat{\mu}_c|}{\sqrt{\frac{\hat{\sigma}_o^2}{N_o} + \frac{\hat{\sigma}_c^2}{N_c}}} \tag{7.25}$$

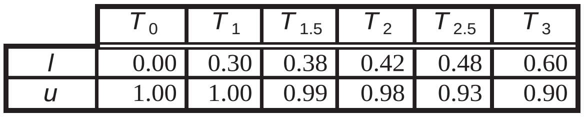
Where the o subscript denotes the original(existing) value and the c subscript denotes the combined value. N indicates the number of observations that contribute to each observed mean. The determination of the appropriate N can be difficult. Evidence based on discovered patterns is typically the result of aggregating hundreds or thousands of sessions. However, subjective evidence is usually based on a single source. As mentioned above, this is one reason statistical inference has not been applied to general reasoning with uncertainty frameworks.

7.1.6 Comparison of Inference Mechanisms

Given the properties listed in Section 7.1.1 and the discussion of the inference mechanisms above, the question is which of the four mechanisms makes the most sense based the seven desired properties of the interestingness framework. Any of the mechanisms fulfills requirements 1, 2, and 5 through 7, however, the Bayesian and Statistical methods do not allow for an explicit measure of ignorance or explicit evidence for and against a belief (since the probabilities for and against always have to sum to 1, one type of evidence always defines the other). This leaves the two-valued representations. A lot of controversy has surrounded the use of Dempster-Shafer inference. However, as pointed out by Dempster [48], Shafer [97], and Pearl [80], this is often due to the misinterpretation of DS pairs as probability-bounds. To illustrate the differences in the inference mechanisms, take as an example two sets of evidence. e_1 and e_2, concerning a particular belief, B. Assume that the subjective quantification of the evidence leads to initial DS pairs and probability bounds that are equivalent:

$$\begin{aligned} e_1 &\rightarrow B_1 : [0.6, 0.8] \\ e_2 &\rightarrow B_2 : [0.7, 0.9] \end{aligned}$$

The combined belief pairs that would be calculated using the various T-norms are given in Figure 7.4. As shown, T_0, the most conservative norm, calculates a final pair of complete ignorance given the slight conflict between the two sets of evidence. This type of combination is appropriate for domains such as medical diagnosis, where the consequences of an improper inference can be severe. For the determination of interestingness, the more liberal T_3 norm makes more sense. This leads to a belief pair of $B[0.6, 0.9]$, meaning that the "true" probability of B occurring is somewhere between 0.6 and 0.9. Even the T_3 norm will always result in increasing or maintaining



	T_0	T_1	$T_{1.5}$	T_2	$T_{2.5}$	T_3
l	0.00	0.30	0.38	0.42	0.48	0.60
u	1.00	1.00	0.99	0.98	0.93	0.90

Figure 7.4: T-Norms for Combination Example

the degree of ignorance. This is somewhat counter-intuitive, since one would expect the existence of more evidence in support of a belief to shrink the bounds around the true value, similar to when the number of observations is increased when testing a statistical hypothesis.

For DS inference, the use of equations 7.20 through 7.22 results in a belief pair of $B : [0.85, 0.9]$. Note that the DS pair is not implying that the probability of B occurring is between 0.85 and 0.9. It is making a weaker ontological commitment about the belief, and should be interpreted as the support for belief B being true is 0.85, and support against belief B being true is 0.1. It does not mean that B is expected to occur between 85% and 95% of the time. A breakdown of the calculations for Dempster's combination rule is shown in Figure 7.5. Note that each belief pair contains three choices, support of B or e_n, against B or $1 - e_p$, and unknown $(e_p - e_n)$. The new values are created by summing up the intersecting values for each choice and normalizing them by the amount of conflict, k (equation 7.20). The amount of conflict is essentially the sum of the intersections that lead to the null set. For Dempster-Shafer, the final assignment of belief to the null set is required to be zero.

As shown in the example, the results of combining evidence with probability-bounds inference is quite different than the results of DS inference. As more pieces of evidence that agree with the first sets of evidence are collected about a particular belief, the probability bounds will stay roughly the same. However, for multiple sets of evidence in support of a belief, the combined belief pair will approach [1,1] for DS inference (and conversely, multiple sets of evidence against a belief will result in

		e_{2n}	$1 - e_{2p}$	$e_{2p} - e_{2n}$
		0.7	0.1	0.2
e_{1n}	0.6	0.42	0.06	0.12
$1-e_{1p}$	0.2	0.14	0.02	0.04
$e_{1p} - e_{1n}$	0.2	0.14	0.02	0.04

k = 1-(0.06 + 0.14) = 0.8

e_n = (0.42 + 0.12 + 0.14)/0.8 = 0.85

1- e_p = (0.02 + 0.04 + 0.02)/0.8 = 0.1

Figure 7.5: Dempster-Shafer Combination Example

a belief pair that approaches [0,0]). Ultimately, the choice of inference mechanism relies on the semantics and type of behavior that make the most sense for a particular domain. As described in the next section, because it is difficult to predict even a range of usage probabilities based on the structure and content of a Web site, the allowance for weaker ontological commitments with the DS inference mechanism makes it the logical choice for the information filter.

7.2 Information Filter

The *information filter* is broken into two major parts - an *objective filter* and a *subjective filter*. The objective filter handles changes in the numeric measures that are associated with the various pattern discovery methods and relates to the third goal of the interestingness framework - identifying changes in the discovered patterns over time. The subjective filter handles beliefs about the usage of pages in a Web site and relates to all three goals of the interestingness framework.

7.2.1 Subjective Filter

For Web Usage Mining, the assumption is that content and structure data can be used as surrogates for the Web site designer's domain knowledge. Links between

Table 7.1: Information Filter Examples

Mined Knowledge	Domain Knowledge Source	Interesting Belief Example
General Usage Statistics	Site Structure	The head page is not the most common entry point for users
General Usage Statistics	Site Content	A page that is designed to provide content is being used as a navigation page
Frequent Itemsets	Site Structure	A set of pages is frequently accessed together, but not directly linked
Usage Clusters	Site Content	A usage cluster contains pages from multiple content categories

pages provide evidence in support of those pages being related. The stronger the topological connection is between a set of pages, the higher the value of s_n is set for the evidence pair. Evidence pairs based on the site content can also be automatically generated by looking at content similarity, and assigning values of s_n and s_p based on the calculated "distance" between pages. Table 7.1 gives some examples of the types of interesting beliefs that can be identified in the Web Usage Mining domain using the subjective filter.

Notice that domain knowledge in the examples is fairly fuzzy, and no specific values are given for the beliefs. This is because while it is fairly easy to identify beliefs such as, "pages `A` and `B` will be used together often," it is much more difficult to reliably estimate the exact parameters of a frequent itemset, cluster of pages, or some other usage pattern in advance. Especially since the number and values of the parameters vary greatly with the pattern discovery method. The Web site structure and content will indicate which pages are *likely* to be used together (or not used together), but not necessarily how often, or to what numerical certainty. This is why the DS inference mechanism has been chosen for the subjective filter. The issue of estimating parameter values (or even deciding which of several available parameters should be estimated at all) is avoided by attaching the weaker semantics to the belief pairs. In addition, DS inference can handle both hard and soft beliefs in the same framework. This is because a belief that is declared to be 100% true ([1,1]) or 100% false ([0,0]) can not be changed by the introduction of new conflicting evidence.

Now that all four dimensions of the subjective filter have been defined, an actual measure of interestingness is needed that meets the three goals of the framework. Comparing the existing belief pair to the combined belief pair after the introduction of new evidence fulfills the second and third goals. A reduction in ignorance or a change in the degree of belief will be identified by changes in the evidence pair. A formal definition of evidence pair change, δ can be defined as follows[1]:

$$\delta = \sqrt{(s_n^o - s_n^c)^2 + (s_p^o - s_p^c)^2} \tag{7.26}$$

In the definition above, the c superscript designates a combined value and o designates an original value. This measure picks up changes in the belief, but not necessarily conflict. Another criticism of Dempster's combination rule has been that the conflict between two sets of evidence is "ignored" since the final belief pair is normalized by the amount of conflict. However, because the DS inference is only being used to update belief pairs, not to infer new facts in a knowledge base, this isn't necessarily a problem. The amount of conflict between an existing belief and a new set of evidence can be added to the amount of change in the belief pair to get a measure of interestingness that meets all three of the goals set in Section 7.1.1. The subjective interestingness, $\mathcal{I}_s$ of a belief after the introduction of new evidence is defined as follows, where k is as defined in equation 7.20:

[1] While this definition uses the familiar L2-norm, other norms could be substituted as appropriate.

Table 7.2: Combination and Interestingness of Boundary Cases

#	New	Existing	Combined	δ	$1-k$	$\mathcal{I}_s$
1	[0.1,0.1]	[0.9,0.9]	[0.5,0.5]	0.82	0.57	1.39
2	[0.9,0.9]	[0.1,0.1]	[0.5,0.5]	0.82	0.57	1.39
3	[0.1,0.1]	[0.1,0.9]	[0.1,0.1]	0.10	0.80	0.90
4	[0.9,0.9]	[0.1,0.9]	[0.9,0.9]	0.10	0.80	0.90
5	[0.1,0.1]	[0.1,0.1]	[0.01,0.01]	0.17	0.13	0.30
6	[0.9,0.9]	[0.9,0.9]	[0.99,0.99]	0.17	0.13	0.30
7	[0.1,0.9]	[0.1,0.9]	[0.17,0.83]	0.02	0.10	0.12
8	[0.1,0.9]	[0.1,0.1]	[0.1,0.1]	0.1	0.00	0.10
9	[0.1,0.9]	[0.9,0.9]	[0.9,0.9]	0.1	0.00	0.10

$$\mathcal{I}_s = \delta + (1-k) \tag{7.27}$$

In the simplest case, all evidence is either 100% for a belief, 100% against a belief, or there is no evidence about a belief. However, as mentioned earlier, belief pairs of [1,1] or [0,0] are special cases because they can not be revised by the introduction of new evidence. While this is a desirable quality in the case of hard beliefs, it is not a good idea for soft beliefs. Therefore, initial soft beliefs will bounded by [0.1,0.1] and [0.9,0.9] to allow for revision in the face of new evidence (This does not prevent combined values from exceeding the bounds, as shown in Table 7.2). There are nine different "boundary" cases that can occur when combining evidence from domain and usage sources for soft beliefs. These are shown in Table 7.2 along with amount of subjective interestingness, $\mathcal{I}_s$.

The first two boundary conditions are for fully conflicting evidence, which gives the highest measure of interestingness. The next two conditions are where there is no existing knowledge about a belief (full ignorance) and the new evidence is strongly positive or negative. Boundary conditions five and six show evidence that is in complete agreement, and the last three conditions are where the new evidence has nothing to say about the belief. Notice that both the change and conflict terms are necessary to create an interestingness measure that meets the three goals and makes intuitive sense. The change term (δ) ends up ranking the conditions with evidence in complete agreement (conditions 5 and 6) as more interesting than the conditions where new strong evidence is replacing ignorance (conditions 3 and 4). However, use of the conflict measure alone would rank the strong new evidence conditions over the conditions of complete conflict (conditions 1 and 2). Another property of the DS semantics and combination rule is that the evidence gets appropriately weighted. As discussed, multiple sets of evidence in favor of a belief will result in a belief pair approaching [1,1] (demonstrated in condition 6 of Table 7.2). The closer an existing belief pair is to the upper or lower bounds, the less a single set of conflicting set of evidence will change the combined pair. This prevents one anomalous set of evidence from distorting the overall belief. If the anomalous evidence becomes the norm, the belief pair will start to be pulled more and more towards the new "corrected" values.

7.2.2 Objective Filter

The *objective filter* is activated for subsequent sets of evidence pertaining to discovered usage patterns. While the subjective filter picks up interesting additions, deletions, and large scale changes to the discovered patterns, the objective filter is designed to monitor changes in the actual parameters that are part of any pattern discovery algorithm. For example, consider an association rule relating pages `A` and `B` with some support, S and confidence, C. Even if the association rule is regularly discovered each time the usage patterns are analyzed, significant changes to the support or confidence should also be considered interesting. A measure like support may not factor into the calculation of an evidence pair supporting the belief that pages `A` and `C` are related through usage. However, a significant drop or increase in the support is certainly of interest. Changes in the parameters associated with a pattern discovery method are best tracked with statistical inference. Maintenance of a mean and variance for each parameter is relatively easy and allows for testing of a null hypothesis that the existing mean and the new value are equivalent. In [101] it is pointed out that hypothesis testing can not handle the situation where it is *expected* that a value is going to change, only the case where the expectation is for a constant mean. However, if a change is expected by the analyst for some reason, this can always be formulated as a subjective belief, and handled in the subjective filter instead. The natural interestingness measure for the objective filter is the significance (α) of the hypothesis test defined in equations 7.23 through 7.25.

7.2.3 Thresholds

The information filter allows two thresholds to be set, one for each of the interestingness measures. Discovered patterns that have either a subjective or objective interestingness that exceeds one of the thresholds are flagged as potentially interesting. Figure 7.6 shows the detailed architecture of the information filter. The representation layer contains both DS belief pairs and parameter values from the usage patterns. The control layer very simply checks if the usage pattern has been previously seen, in order to determine if the objective filter is appropriate. Finally, the inference layer calculates the interestingness of the discovered patterns and performs the evidence combination in order to use the updated belief pairs and parameter values for the next set of evidence. The task of quantifying the various kinds of evidence into belief pairs will be discussed in the next section.

7.3 Evidence Quantification

In order to make use of the subjective information filter described in the previous section, the different sources of evidence need to be transformed into belief pairs. By the nature of *subjective* evidence, there is no one correct method for quantifying the evidence. All of the quantification methods given in this section ultimately rely on domain knowledge about the usage of Web sites. For belief pairs created by the analyst that are not a direct result of structure or content evidence, the quantification is whatever the analyst feels comfortable with. The task of converting fuzzy concepts, such as *unlikely* or *extremely likely*, into numerical ratings has been used successfully in a number of research efforts [28, 90]. However, the main goal of the information filter is to use the existing data sources to automatically generate belief pairs in order to provide standardized measures for similar situations and avoid the labor associated with manual quantification. The remainder of this section defines methods that can be used for quantifying evidence from usage, structure, or content data into beliefs that various sets of pages will be related through usage.

7.3.1 Usage Evidence Quantification

The appropriate method for quantifying usage evidence is completely dependent on the type of pattern discovered and the algorithm used for the discovery. As mentioned previously, the number and types of parameters associated with pattern discovery methods is vast. However, since the beliefs are in the form of what sets of pages will be related through usage, any discovered usage pattern is a direct measure of the true relation.

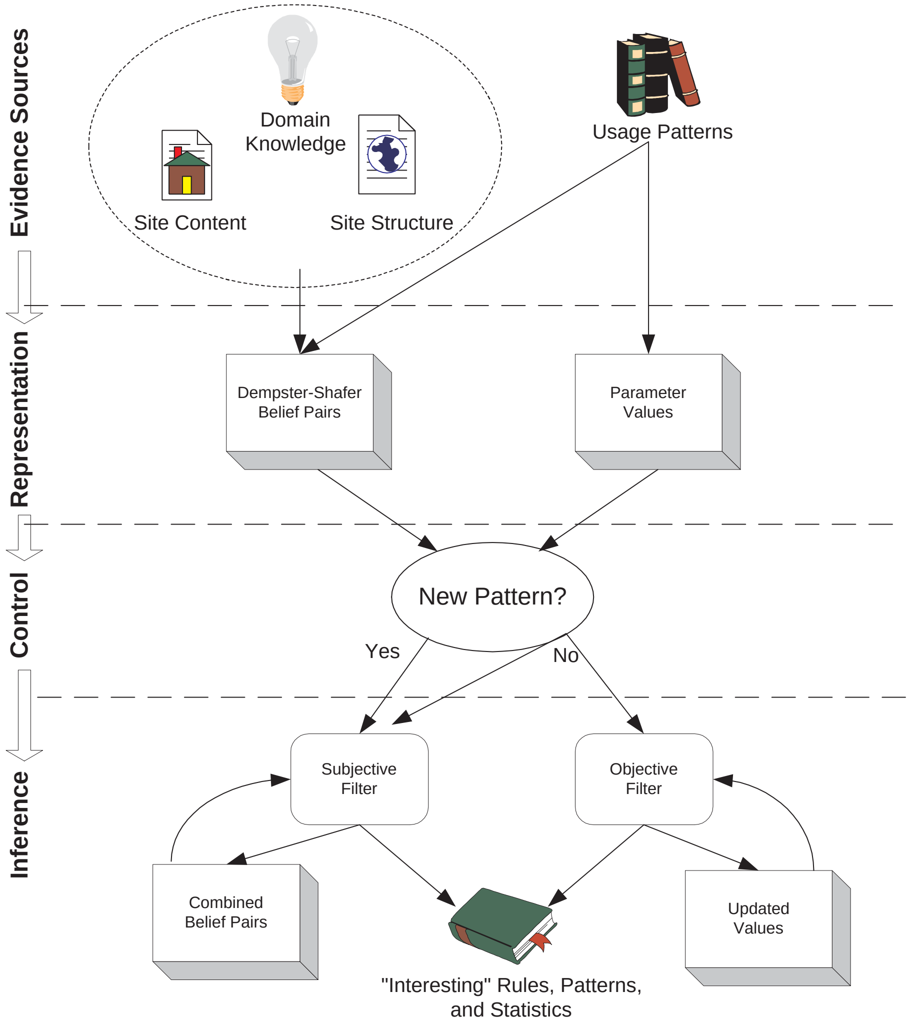


Figure 7.6: Detailed Information Filter Architecture

Frequent Itemset Quantification

As described in Chapter 6, frequent itemsets are one of the simplest forms of patterns that can be discovered. Therefore, the set of beliefs that evidence from frequent itemsets is applicable to will also be simple, and will not include any restrictions on the order or time of the page accesses. The beliefs for a set of pages, $\mathcal{P} = \{v_1, \ldots, v_n\}$ and a set of sessions, $\mathcal{S}$, will be of the form:

$$\begin{aligned} &B^1[s_n, s_p] \text{ where} \\ &s_n = \text{ evidence for } \{\mathcal{S} : v_1 \in \mathcal{S}\} = \{\mathcal{S} : v_2 \in \mathcal{S}\} \ldots = \{\mathcal{S} : v_n \in \mathcal{S}\} \\ &1 - s_p = \text{ evidence for } \{\mathcal{S} : v_1 \in \mathcal{S}\} \neq \{\mathcal{S} : v_2 \in \mathcal{S}\} \ldots \neq \{\mathcal{S} : v_n \in \mathcal{S}\} \end{aligned} \tag{7.28}$$

This type of belief will be referred to as a *type 1* belief. The only parameter that is regularly associated with frequent itemsets is the *support* of the itemset. However, *association rules*, which take frequent itemsets one step further by assigning a conditional direction to the itemsets, also have a measure of *confidence*. The confidence of an association rule naturally maps to a degree of belief that a set of pages are related through usage. Given an `n` item frequent itemset, `n` different association rules can be generated, all with potentially different levels of confidence. To create a single evidence pair for a frequent itemset, the maximum confidence from the corresponding set of association rules can be used:

$$B^1_u[s_n, s_p] = [C_{max}, C_{max}] \tag{7.29}$$

where C_{max} is taken from the set of confidence levels, $\{C_1, \ldots, C_n\}$, associated with an `n` item frequent itemset. While an evidence pair can be generated for each individual association rule, this causes a number of practical problems, the largest being the combinatorial explosion of the number of rules. The second problem has to do with very frequently occurring pages. Users do not generally browse an entire Web site. In fact, the mean number of pages requested during a visit is around ten [86]. Therefore, the confidence for any association rule with a popular page view as the antecedent is expected to be low. This does not necessarily mean that the pages are unrelated, just that each visitor is not viewing every page on the site. However, low confidence association rules for pages that are expected to be related will get labeled as interesting. A low confidence cutoff could be used during pattern discovery to limit the discovered rules to those with high confidence, but this will lead to rules having unexpectedly low confidence from being excluded as well. The use of the maximum confidence for a set of association rules with the same pages, i.e. the frequent itemsets, somewhat mitigates the affect of user attrition on the usage evidence. Also, sets with unexpectedly low confidence will remain in the evidence set and given a high degree of subjective interestingness along with the rules with unexpectedly high confidence.

Usage Cluster Quantification

Unlike frequent itemsets, usage clusters do not necessarily represent actions that occur within single sessions. A set of pages from a usage cluster may not appear in any one session. Also, the available parameters for creating evidence pairs depends on the algorithm used to create the clusters. There are no universal parameters like support and confidence to help construct an evidence pair. Figure 7.7 shows the four different cases that can be encountered. The first two cases are where the usage cluster is either equal to or a subset of the domain grouping[2]. The third case is when a single usage cluster spans multiple domain sets, and the fourth case is the opposite, where a single domain set spans multiple usage clusters. All of the cases except the

[2] The domain page grouping can be created by clustering or classifying pages based on the content or structure. The pages could also be manually grouped by a domain expert.

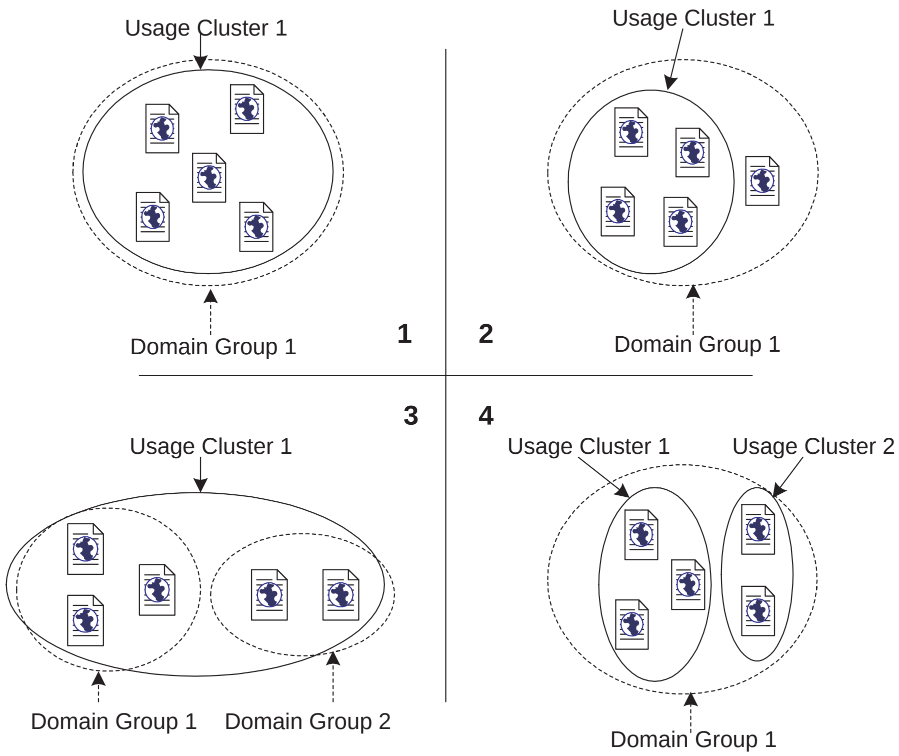


Figure 7.7: Domain vs. Usage Clusters

first could be considered interesting. Case two is somewhat interesting for what the usage cluster does not contain. Case three is interesting because multiple domain clusters are related through usage, and case four shows a case where there is a finer granularity to the page classification than indicated by the domain evidence.

The beliefs that relate usage cluster evidence are that pages related by content or structure will be represented by a single usage cluster. If a usage cluster maps to several sets of pages related by content/structure, or several usage clusters map to the same content/structure cluster or category, this would be unexpected and therefore interesting. These beliefs, referred to as *type 2* beliefs, will be of the following form for a set of usage clusters, $\mathcal{U}$ and domain sets $\mathcal{D}$:

$$
\begin{aligned}
&B^2[s_n, s_p] \text{ where} \\
&s_n = \text{ evidence for } \{\mathcal{U}_i \bigcap \mathcal{D}_j\} = \begin{cases} \mathcal{D}_j & \text{for } i = j \\ \emptyset & \text{for } i \neq j \end{cases} \\
&1 - s_p = \text{ evidence for } \{\mathcal{U}_i \bigcap \mathcal{D}_j\} \neq \begin{cases} \mathcal{D}_j & \text{for } i = j \\ \emptyset & \text{for } i \neq j \end{cases}
\end{aligned}
\tag{7.30}
$$

Notice that the type 2 beliefs are fundamentally different from the type 1 beliefs in that the support values are based on evidence from more than one source. For the type 1 beliefs, the support values are based on usage evidence (or as will be seen in the next section, structure evidence) alone. Here, it is the intersection of usage evidence and domain evidence that dictates the support values. Therefore, the quantification of evidence from usage clusters will be tied to the domain evidence that is being used for comparison. Section 7.3.3 will present a specific method for comparing usage clusters with a content hierarchy.

7.3.2 Structure Evidence Quantification

There are two kinds of structure that must be handled for a general Web site - intra page-view structure and inter page-view structure. As discussed in Chapter 3, it may be desirable to discover patterns based on the individual page files, and then filter out the patterns that represent nothing but a single page view. This can be accomplished within the information filter framework by assigning a hard belief pair of [1,1] to any set of page files that make up a single page view.

The inter page-view structure is equivalent to a directed graph. A natural measure of the degree of belief that a set of pages will be used together is the connectivity (or lack of connectivity) of a set of page views. A directed graph is defined by a set of nodes, V, and a set of directed edges, E, represented by ordered pairs (u, v), where u is the source and v is the destination of the edge. A directed graph is weakly connected if every pair of nodes is joined by a *semipath.* A semipath is defined as a series of edges that lead from the first node to the second node, ignoring the direction of the edges in the path. A directed graph is connected if there is a directed path from at least one to the other of every pair of nodes. Finally, a directed graph is *strongly connected* if every two nodes are mutually reachable through a directed path.

The edge-connectivity of a graph, $\mathcal{K}$ is defined as the minimum number of edges that must be removed in order to result in a disconnected graph. In this case, the definition of weakly connected will be used, so the maximum connectivity of a set of n nodes is $2(n-1)$, and the minimum connectivity is 0 (disconnected). The higher the connectivity, the greater the degree of belief that the pages will related through usage. On the other hand, the degree of belief against a set of pages being related can be defined by the disconnectivity of a graph. For a set of n nodes, the minimum number of edges that must be present to have a weakly connected graph is $(n-1)$. Therefore, the disconnectivity, $\mathcal{K}'$, is bounded by $(n-1)$ (no edges at all), and 0 (the graph is connected). All of the definitions just listed can be derived from Menger's theorem [72], which roughly states that the minimum number of nodes separating two nonadjacent nodes s and t equals the maximum number of disjoint s-t paths. In order to meet the restrictions of equations 7.18 and 7.19, the connectivity and disconnectivity of a set of nodes must be normalized by their maximum possible values. This gives and evidence pair for a set of page views as follows:

$$B_s^1[s_n, s_p] = [\frac{\mathcal{K}}{\mathcal{K}_{max}}, 1 - \frac{\mathcal{K}'}{\mathcal{K}'_{max}}] \tag{7.31}$$

This results in a fully strongly connected set of nodes being assigned an evidence pair of $[1,1]$ and a fully disconnected set of nodes being assigned an evidence pair of $[0,0]$ (which would replaced by $[0.9,0.9]$ and $[0.1,0.1]$ respectively, as described in Section 7.2). Figure 7.8 shows the structure evidence pairs that would be calculated for a number of simple cases and Figure 7.9 shows a few examples of the subjective interestingness that would be calculated when comparing frequent itemsets with site structure.

7.3.3 Content Evidence Quantification

Content evidence can come from clustering, categorization, or a manually defined hierarchy. Content information is best used as evidence for the type 2 beliefs, that sets of pages will have equivalent usage and domain clusters. The following quantification method can be used for a content hierarchy, which is very common for e-commerce Web sites. It is assumed that the hierarchy is a tree, with the root node containing the entire site. The tree does not have to be balanced, and a given leaf (e.g. product) can have more than one parent (e.g. category). For any group of pages from the content hierarchy, there exists a *local root* that is an ancestor for all of the pages in the group. Assuming that the leaf nodes are labeled as level 0, and the site root node is level n, the *minimum local root* is defined as the local root with the minimum height for a set of pages, $\mathcal{P}$ and the set of local roots, r:

$$\begin{aligned} &hr_{min} = min(height(r_1), \ldots, height(r_m)), \\ &\text{where, } \forall v \Rightarrow r_i \in ancestor(v) \end{aligned} \tag{7.32}$$

If $hr_{min} = 1$, for a set of pages, then the pages are all members of the same low-level content category. As hr_{min} increases, the pages in the set represent more different categories. The reciprocal of hr_{min} is a natural quantification for content evidence supporting a type 2 belief. Because the tree is not required to be balanced, the height of the minimum local root may vary for different pages in the set. Therefore, the reciprocal of the average height of the local root, normalized to range between zero and one, will be used for evidence in support of type 2 beliefs.

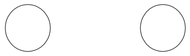
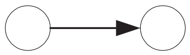
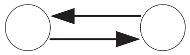
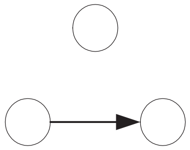
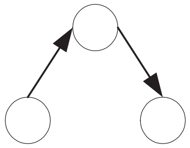
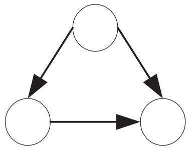
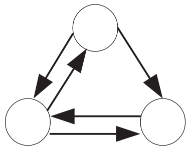
	K	K’	$[s_n,s_p]$
	0	1	[0.1,0.1]
	1	0	[0.5,0.9]
	2	0	[0.9,0.9]
	0	1	[0.1,0.5]
	1	0	[0.25,0.9]
	2	0	[0.5,0.9]
	3	0	[0.75,0.9]

Figure 7.8: Sample Evidence Pairs for Simple Graphs

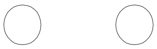
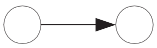
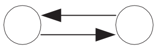
	Usage Evidence (Max Confidence)		
Structure Evidence	10%	50%	90%
	0.30	0.50	1.39
	1.27	0.59	0.58
	1.39	0.50	0.30

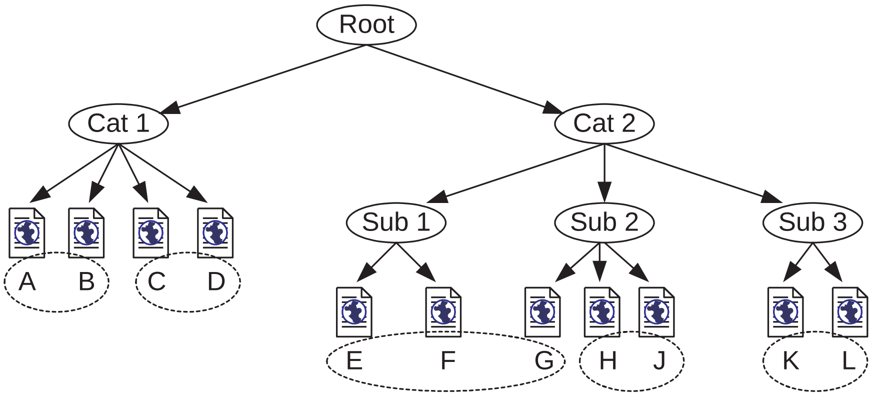
Figure 7.9: Structure vs. Usage Subjective Interestingness

A high local root can be considered as evidence against a set of pages being represented by a single usage cluster. In the extreme, if the Web site root is the minimum local root for a set of pages, this is the strongest evidence against that set of pages belonging to the same usage cluster. The maximum height of the tree has a large effect on the strength of the evidence. A distance of two between the site root and local root is not particularly strong negative evidence if the maximum height of the tree is three, but very strong if the maximum height is twenty. Therefore, the evidence against a set of pages being represented by a single cluster will be taken as the distance to the site root divided by the maximum tree height. This results in an evidence pair as follows:

$$B_c^2[s_n, s_p] = [\frac{\frac{1}{avg(hr_{min})} - \frac{1}{h_{max}}}{1 - \frac{1}{h_{max}}}, \frac{(h_{max} - avg(hr_{max}))}{h_{max}}] \tag{7.33}$$

The evidence pair defined above handles cases one and three from Figure 7.7, but not cases two and four. Both cases two and four will result in an evidence pair of $[1, 1]$ from equation 7.33 because the usage clusters are subsets of a single content category. The fourth case can be handled with the reciprocal of the number of distinct sets with the same local root, which will be referred to as the *multi-set factor*. In an extreme case, there could be a usage cluster associated with each individual item in a content category. This means the multi-set factor can range from the reciprocal of the maximum number of items to one. The factor can be normalized to range from zero to 1, similar to the method used in equation 7.33. However, when the minimum local root is high in the content tree, it is not particularly unusual to find more than one cluster with the same local root. Therefore, the multi-set factor will only be applied when $hr_{min} = 1$. The subset case from Figure 7.7 is difficult to handle in an automated manner. While a cluster that contains all but one item of a content category could be considered interesting, the subjective interestingness of a cluster containing half of the items, or two items from a category is not quite as clear. In practice, especially with clustering methods that do not necessarily place all of the items into clusters, it is very common to have clusters that are subsets of content categories. Therefore, a *subset factor* will not be incorporated into the content evidence pair.

Figure 7.10 shows an example content hierarchy, usage clusters, and the content and combined belief pairs that would be calculated as a result. The usage clusters are simply assigned an evidence pair of $[0.9, 0.9]$. While this doesn't take into account the strength of the cluster, it is parameter independent. The highest subjective interestingness is assigned to the cluster that spans multiple categories, $\{E, F, G\}$. Next, the clusters that have the same local root, $\{A, B\}$ and $\{C, D\}$, are ranked as slightly less interesting. Finally, clusters $\{H, J\}$ and $\{K, L\}$ are ranked the lowest in terms of subjective interestingness.



Page Set	A,B	C,D	E,F,G	H,J	K,L
hr_{min}	1	1	2	1	1
h_{max}	2	2	3	3	3
max pages	4	4	5	3	2
set count	2	2	1	1	1
multi-set factor	0.50	0.50	1.00	1.00	1.00
s_{1n}	0.50	0.50	0.25	0.90	0.90
s_{1p}	0.50	0.50	0.50	0.90	0.90
s2n	0.90	0.90	0.90	0.90	0.90
s2p	0.90	0.90	0.90	0.90	0.90
s_n	0.90	0.90	0.86	0.99	0.99
s_p	0.90	0.90	0.86	0.99	0.99
I_{subj}	1.07	1.07	1.18	0.30	0.30

Figure 7.10: Content Hierarchy Example

Chapter 8

Preprocessing Experiments

In order to properly test the information filter discussed in Chapter 7, the raw click-stream data collected from server logs must be cleaned and preprocessed into server sessions. This chapter discusses the results from several experiments run to test the effectiveness of the preprocessing heuristics and episode identification methods described in Chapter 5. The results from experiments conducted to test the information filter will be presented in the next chapter.

8.1 Data Sets

Data from several Web sites has been collected. The size and dates of the server logs for each data set are shown in Table 8.1. The Global Reach site hosts Web sites for several small businesses. The log for the Global Reach site, www.global-reach.com, is from 1997 and prior to large-scale e-commerce on the Web. No products were being sold directly through the Global Reach sites at the time. The Global Reach data set consists only of the Web server log, without any additional information about the structure or content of the site.

The Computer Science Department site is designed to provide information about people, research projects, facilities and classes. The dataset includes the content (e.g. text) and structure for the site. The frame structure for the Computer Science site is highly variable and was not taken into account during the preprocessing.

Table 8.1: Data Used for Evaluation

Site Name	Dates	Sessions	Page File Requests	Page View Requests	Site Structure
Global Reach	3/97	9718	51,697,	-	No
Computer Science Dept.	2/10/99 to 2/17/99	10,609	43,158	-	Yes
Small E-commerce	9/99 to 11/1999	7079	63,623	63,623	No
Large E-commerce	12/16/99	46,405	663,844	463,450	Yes
Large E-commerce	12/19/99	50,581	608,823	405,006	Yes

The Small and Large E-commerce sites both sell products directly through the Web. The Small E-commerce site sells a small set of closely related products, and the Large E-commerce site is a general retailer for a variety of merchandise. The Small E-commerce site is a single framed site, which means the page files and page views are equivalent. The Large E-commerce site has three frames for every page view - a left navigation frame, top navigation frame, and a body frame. The majority of the experiments reported on in this chapter and the next will be carried out on the data from the Large E-commerce site.

A simplified map of the Large E-commerce site is shown in Figure 8.1. The map is not complete, but shows the core set of pages and links that are used most often. The page files outlined in bold are the key pages in the site related to the display and sale of products. There are two main entry points, one for members and one for anonymous users. The site includes 571 different categories and 5757 different products leading to between 40,000 and 50,000 distinct page views each day.

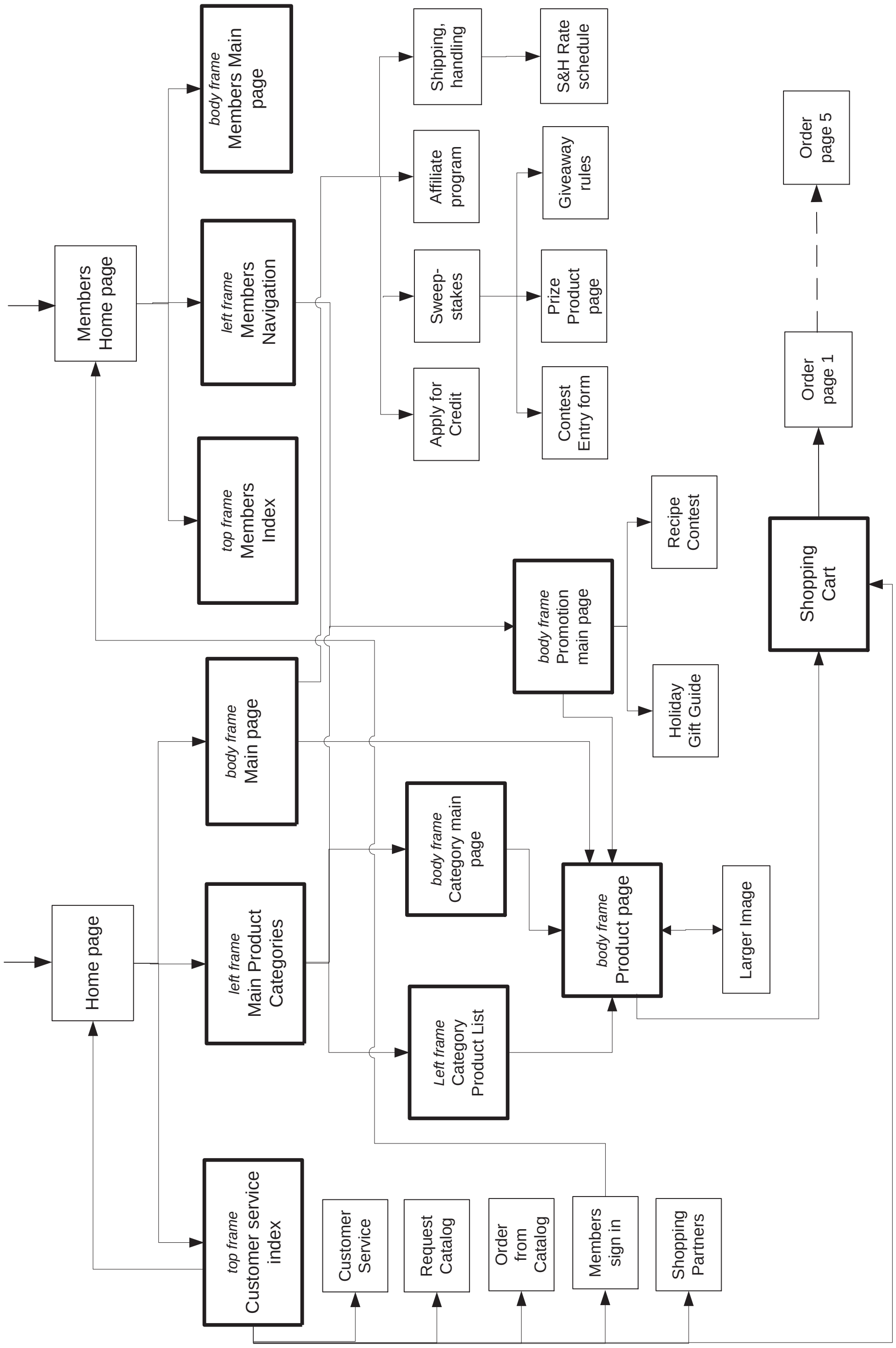


Figure 8.1: Large E-Commerce Site Map

8.2 Session Identification

The test hypothesis for the session identification heuristic described in Chapter 5 is that sessions identified with the heuristic will be identical to sessions identified through a positive identification method such as cookies or embedded session IDs. For the data sets where only the IP address and agent are available for identifying sessions, there is no practical method for testing this hypothesis. However, for logs with an embedded session ID, a direct comparison can be made.

Methods

The input for these experiments was output of the data cleaning task for usage preprocessing. The schema for the input and output for these experiments is shown in Figure 8.2. For both the Large and Small E-commerce logs, a *clean_log* table was created by parsing an ECLF log, removing all graphics requests, and removing all log entries that did not contain an embedded session ID. This included removing a number of hidden POST requests, since the embedded session ID was not recorded in the Server log. Each distinct URI for both the request and referrer fields was given a numeric alias. The CGI values that contained the embedded session IDs were not included when assigning numeric aliases. The session identification heuristic described in Chapter 5 was implemented in Java and run by connecting to the database tables through a JDBC connection. The accuracy of the heuristic was calculated by comparing the sessionid column of the *session_log* table to the id_agent_id column.

Results

The results of running the experiment on the Small E-commerce log and one of the Large E-commerce logs are shown in Table 8.2.

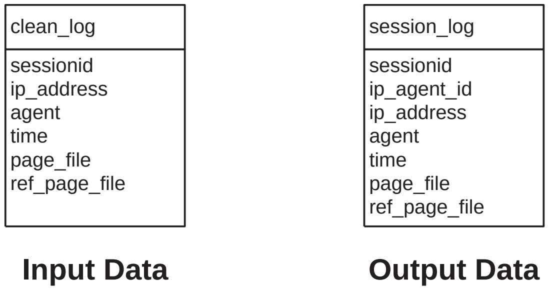


Figure 8.2: Schema for Session Identification Experiments

Table 8.2: Comparison of IP/Agent Session Heuristic to Embedded Session IDs.

Data Set	Large E-commerce 1	Small E-commerce
Embedded Sessions	46,405	7,062
IP/Agent Sessions	47,705	6,406
Embedded Sessions w/ Multiple IP/Agent	8,053	384
IP/Agent w/ Multiple Embedded Sessions	7164	1471
Correct IP/Agent Sessions	26,977	2,796
Percent Correct	58.13%	39.59%

Discussion

As shown in Table 8.2, the performance of the heuristic for the e-commerce sites is very poor. Even though the total number of sessions is fairly close, the competing effects of rotating IP addresses and ISP proxy servers results in a low overall accuracy. Another problem factoring into the error for the e-commerce sites is that the secure and non-secure proxy servers for many ISPs have different IP addresses. In any case, the test hypothesis is not confirmed. It is clear that for the state of the Web today, the Chapter 5 session identification heuristic is not adequate. Unless a better session identification heuristic is developed, a positive session identification method will be required in order to get accurate results from Web Usage Mining.

8.3 Path Completion

The test hypothesis for the path completion heuristic described in Chapter 5 is that the heuristic will identify all of the page references missing from a server log due to client-side caching. An additional hypothesis is that dynamic sites are not subject to a significant amount of client-side caching.

8.3.1 Static Web Site

A small portion of the Computer Science Web site, the WebSIFT research project pages, was used to test the hypothesis that the path completion heuristic will correctly identify all of the page references missing from a server log. The WebSIFT project pages are a set of completely static single frame Web pages with a fairly simple inter-page structure. In most cases, there is only one possible path between a set of pages.

Methods

The WebSIFT project pages were instrumented with a Java applet that was written to act as a client-side data collection mechanism, similar to the experiments described in [99]. Each time one of the instrumented pages was requested by a client browser, data about the request was sent back to a server side Java application for logging. The Java applet was capable of detecting cached page references that occurred when the forward or back buttons were used. The existence of the applet prevented repeated requests for a page view through a hypertext link from being cached. The schema for the client log, along with the schema for the server log input and output for this experiment are shown in Figure 8.3. Minimal preprocessing was necessary to create the *client_log* table since the format of the log could be controlled through the Java application. Only URI normalization was performed to ensure that the page view identifier strings would match the identifiers from the Web server log. The server input for this experiment was the output of the session identification task for usage preprocessing. The *session_log* table was created by parsing an ECLF log, removing all graphics requests, and running the session identification heuristic. The *session_log* table was also filtered to remove all requests that did not contain the string "`cooley/websift`," since only the WebSIFT project pages were instrumented with the Java applet.

The page view identification algorithm described in Chapter 5 was run on the *session_log* table. However, since the page views were single framed and static, the page view identification algorithm simply called the path completion heuristic when the referring URI did not match the previous requested URI. A two second constant was used as the access length for the inferred page views. The comparison of the *client_log* and *complete_log* tables was performed manually.

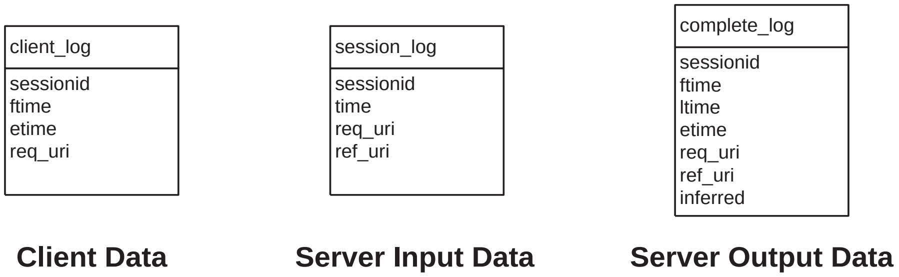


Figure 8.3: Schema for Static Path Completion Experiment

Results

Unfortunately, due to low amounts of traffic and the lack of robustness of the server-side component for the Java applet, very little data was collected for analysis, 293 page views over 24 sessions. The comparison showed that about 25% of the page references for the WebSIFT project site were cached due to the use of the back button. The path completion heuristic correctly identified 95% of the missing references.

Discussion

The path completion heuristic performed very well on the static site, confirming the test hypothesis. The only cached page views that were missed occurred at the end of a session, with no subsequent requests. However, because of the simplicity of the site and the low volume of traffic, these results are not necessarily generalizable to other Web sites.

8.3.2 Dynamic Web Site

Even though it was not possible to collect client-side data for comparison, the path completion algorithm was also run on a portion of the data from the Large E-commerce site.

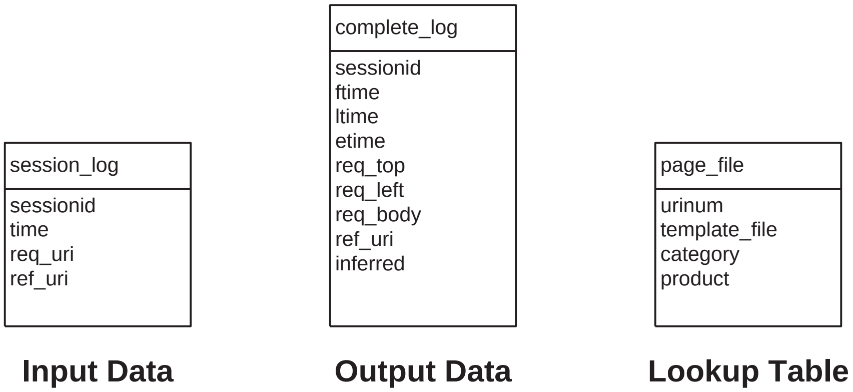


Figure 8.4: Schema for Dynamic Path Completion Experiment

Methods

The input and output schema for this experiment are shown in Figure 8.4. The *req_uri*, *ref_uri*, *req_top*, *req_left*, and *req_body* fields in the *session_log* and *complete_log* tables are all numeric identifiers that correspond to an entry in the *page_file* table. For the Large E-commerce site, the *template_file*, *category*, and *product* CGI values serve as the identifier for the page file content. The *session_log* table was created by parsing an ECLF log, removing all graphics requests, removing all log entries that did not contain an embedded session ID, and setting the sessionid field equal to the sessionid CGI value for the request. This included removing a number of hidden POST requests, since the embedded session ID was not recorded in the Server log. The *template_file*, *category*, and *product* CGI values for both the request and referrer were replaced with the appropriate numeric identifier from the *page_file* table.

The page view identification algorithm described in Chapter 5 was run on the *session_log* table. All of the page views for the Large E-commerce site have three frames, so each page view identified had a top, left, and body page file. The path completion heuristic was called from the page view identification algorithm when the

Table 8.3: Path Completion for Large E-commerce Web Site.

Sessions	15,145
Logged Page Views	41,059
Inferred Page Views	844
Total Page Views	41,903
Sessions with $\geq$ 1 inferred View	429

referring page file was not equal to one of the three page files from the previous page view. The error handling routine for the page view identification algorithm simply discarded sessions where a previous page view could not be found containing the referring page file.

Results

The results of the experiment are summarized in Table 8.3. 15,145 of 30,000 sessions were successfully processed by the heuristic. 2.8% of the sessions had at least one detectable cached reference. The average number of cached references was roughly two page views for the sessions that had at least one cached page view. 15 sessions had greater than four cached page views.

Discussion

As shown, the amount of caching appears to be far less for the fully dynamic Web site than the static portion of the Computer Science site tested in the previous section. Because there was no client-side logging mechanism in place, there is no way to verify the results. The high number of rejected sessions was mainly a result of removing the hidden POST requests from the log during preprocessing. The hidden POST requests often contain the referring page file for a subsequent request. Without access to the hidden POST data, there is no reasonable way to infer these missing page references. For the sessions that were successfully completed, the results confirm the test hypothesis, and indicate that cached page references can be ignored for fully dynamic Web sites without having a large effect on the analysis results.

8.4 Episode Identification

The reference length method of episode identification presented in Chapter 5 relies on the assumption that pages treated as media pages will have longer view times than auxiliary pages. The first question that needs to be answered is how often this holds true. The second question is that assuming media pages do have longer reference times, how do the episodes identified with this method compare to episodes identified by another method, such as maximal forward reference.

8.4.1 Page Usage Type

The test hypothesis for this experiment is that the average reference length for media page views is longer than the average reference length for navigation page views. Determining the intended usage type of a page view is a difficult problem in general due to the fact that most page views are a combination of links, text, and graphics. The most accurate method is to label the pages by hand, or apply specific knowledge about a Web site. The Large E-commerce site is set up in such a way that the "actual" page usage type is fairly easy to determine. The page files for the "body" frame are almost exclusively media pages. Even though a list of links is always present in the top and left frames, page views that are a result of a change in the body frame can be declared to be media views. Similarly, page views that are the result in changes to the left frame can be declared to be auxiliary views.

Methods

The input for this experiment is the output of the page view identification task for usage preprocessing, shown in Figure 8.4 as table *complete_log*. The data cleaning and session identification tasks were performed by parsing an ECLF log, removing all graphics requests, removing all log entries that did not contain an embedded session ID, and setting the sessionid field equal to the sessionid CGI value for the request. This included removing a number of hidden POST requests, since the embedded session ID was not recorded in the Server log. The *template_file*, *category*, and *product* CGI values for both the request and referrer were replaced with the appropriate numeric identifier from the *page_file* table. The page view identification algorithm described in Chapter 5 was run on the *session_log* table. All of the page views for the Large E-commerce site have three frames, so each page view identified had a top, left, and body page file. The path completion heuristic was not called.

Page views that were a result of a change to the *req_body* field were declared to be media page views, and page views without a change to the *req_body* field were declared to be auxiliary page views. The reference length for each page view was calculated as the difference between the last file request time (*ltime*) and the view end time (*etime*). The last page view for each session was not included in the calculations since there was no end time for the page view.

Results

The average reference length for media page views for the Large E-commerce site was 83.4 seconds, and the average reference length for the auxiliary page views was 40.6 seconds. Figures 8.5 and 8.6 show the reference lengths for a typical media page view and a typical navigation page view from the Large E-commerce site.

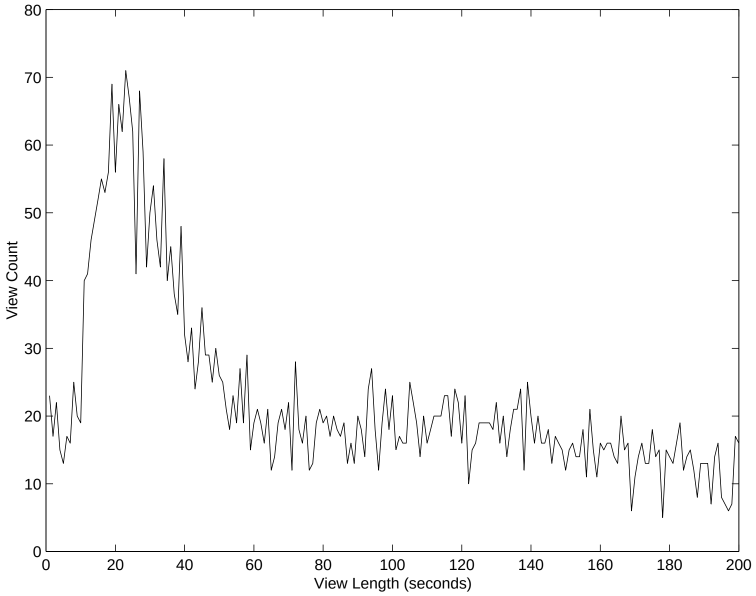


Figure 8.5: Typical Media Page Reference Lengths

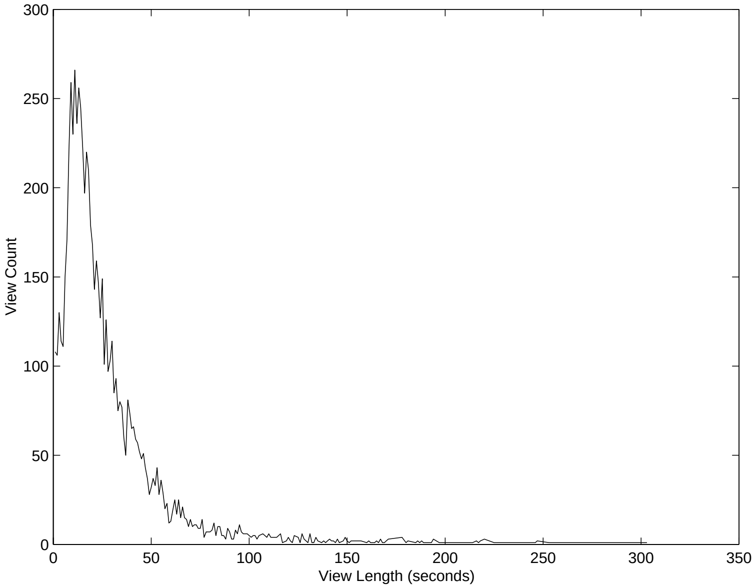


Figure 8.6: Typical Navigation Page Reference Lengths

Discussion

The average media page reference length was much longer than the auxiliary page reference length, confirming the test hypothesis. In addition, the reference lengths for the navigation page are heavily skewed towards short view times and the media page view has a distribution with a higher percentage of long view times. However, there are still a significant number of “short” reference lengths for the media page, and “long” reference lengths for the navigation page. This means that the reference length episode identification method will end up misclassifying a large number of page views. However, as discussed in Chapter 5, the reference length method is designed to identify the page views that are *used* as a media page, not the page views that have been statically *classified* as a media page. Media pages with an unusually short average reference length and navigation pages with an unusually long average reference length may indicate a Web site design problem.

8.4.2 Synthetic Data

In order to compare the performance of the episode identification methods, a server log with known patterns was needed. As was shown in Chapter 5, the three different approaches result in different sets of episodes, even for a simple example. The test hypothesis for this experiment is that the reference length episode identification method will perform better than other episode identification methods for usage data that has a heavy-tailed gamma distribution of reference lengths. Server logs with generated data were created for the purpose of comparing the approaches. Besides a known distribution, the advantage of using generated data for testing the episode identification methods is that the actual percentage of auxiliary references and interesting association rules are known in advance. The obvious disadvantage is that it is, after all, only manufactured data and should not be used as the only tool to evaluate the episode identification methods.

Methods

The data generator takes a text file with a description of a Web site as a directed tree or graph and some embedded association rules. The embedded association rules become the "interesting" rules that are checked for during experiments with different episode identification approaches. The interesting rules in this case are associations that are not born out by the site structure, which would be identified by the site filter. For each "user", the next log entry is one of three choices, a forward reference, backward reference, or exit. The probability of each choice is taken from the input file, and a random number generator is used to make the decision. If the page reference is going to be a media reference, the time is calculated using a normal distribution and a mean value for the time spent on a page taken from the input file. The times for auxiliary hits are calculated using a gamma distribution. Figure 8.7 shows a histogram of the reference lengths for a generated data set, which is very similar to the histogram of page file access times for the server log data shown in Fig. 5.14.

Three different types of Web sites were modeled for evaluation, a sparsely connected graph, a densely connected graph, and a graph with a medium amount of connectivity. The sparse graph had an average incoming order of one for the nodes. The medium and dense graphs used the same nodes as the sparse graph, but had more edges added for an average node degree of four and eight, respectively.

Episodes were identified using the reference length and maximal forward reference episode identification methods discussed in Chapter 5. In addition, a "straw-man" heuristic referred to as the *time window* method was used to identify episodes. The time window episode identification method partitions a server session into time intervals no larger than a specified parameter. The approach does not try to identify

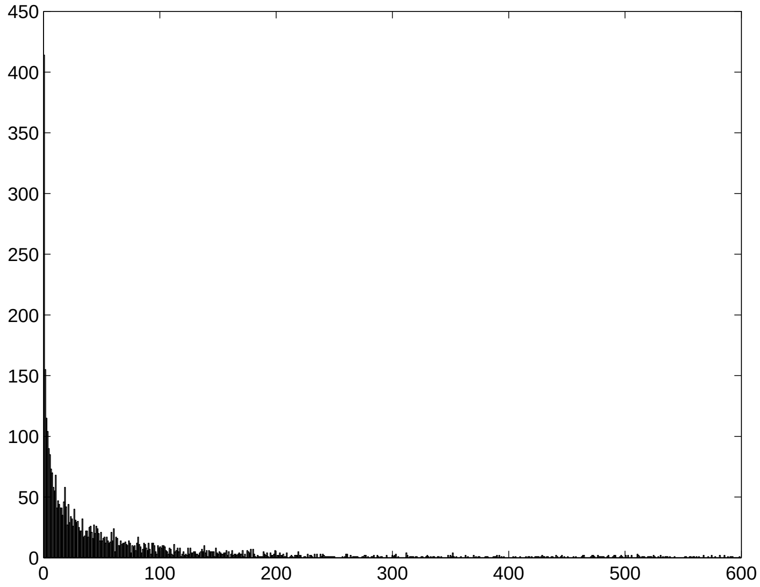


Figure 8.7: Histogram of Generated Data Reference Lengths (seconds)

episodes based on media or auxiliary references, but instead assumes that meaningful episodes have an overall average length associated with them. For a sufficiently large specified time window, each episode will contain an entire server session. Since the time window approach is not based on the media/auxiliary page model, it is not possible to create two separate sets of episodes. The last reference of each episode does not correspond to a media reference, the way it does for the auxiliary-media episodes of the reference length and maximal forward reference methods. The *usage type* method from Chapter 5 was not included in these tests since it is guaranteed to be correct given the method of test data generation. Finally, the Apriori algorithm was run on each set of episodes to discover association rules.

Table 8.4: Number Discovered/Total Possible Interesting Rules.

Approach	Parameter	Sparse	Medium	Dense
Time Window	10 min.	0/4	0/3	0/3
	20 min.	2/4	2/3	1/3
	30 min.	2/4	2/3	2/3
Reference Length	50%	4/4	3/3	3/3
	65%	4/4	3/3	3/3
	80%	4/4	3/3	3/3
M. F. R.		4/4	2/3	1/3

Table 8.5: Ratio of Reported Confidence to Actual Confidence

Approach	Parameter	Sparse	Medium	Dense
Time Window	10 min.	null	null	null
	20 min.	0.82	0.87	0.87
	30 min.	0.98	0.90	0.88
Reference Length	50%	0.99	0.95	0.96
	65%	1.0	0.99	0.96
	80%	0.97	0.99	0.96
M. F. R.		0.79	0.47	0.44

Results

Table 8.4 shows the number of rules discovered for each combination of episode identification method and Web site density. Table 8.5 shows the average ratio of reported confidence to actual confidence for the interesting rules discovered.

Discussion

As shown, the reference length approach performed the best, even though it uses a pure exponential distribution to estimate the cutoff time, while the synthetic data was created with a combination of normal and gamma distributions. The maximal forward reference approach performed well for sparse data, but as the connectivity of the graph increased, its performance degraded. This is because as more forward paths become available, a media reference is less likely to be the "maximal forward reference." For dense graphs, auxiliary-media episodes would probably give better results with the maximal forward reference approach. As expected, the performance of the time window approach was relatively poor, but as the time window increased, so did the performance.

The differences between the reference length and maximal forward reference approaches stand out in Table 8.5. The reported confidence of rules discovered by the reference length approach were consistently close to the actual values. Note that even though the synthetic data had an actual auxiliary page ratio of 70%, inputs of 50% and 80% produced reasonable results. The reported confidence for the rules discovered by the maximal forward reference approach was significantly lower than the actual confidence, and similar to the results of Table 8.4, it degraded as the connectivity of the graph increased. These results confirm the hypothesis that the reference length episode identification method performs best for data with a heavy-tailed gamma distribution.

8.4.3 Web Server Data

The test hypothesis for this experiment is that association rules discovered from reference length episodes will contain subjectively interesting rules not found from maximal forward reference association rules. In addition, the maximal forward reference association rules will not contain any subjectively interesting rules not found with the reference length method.

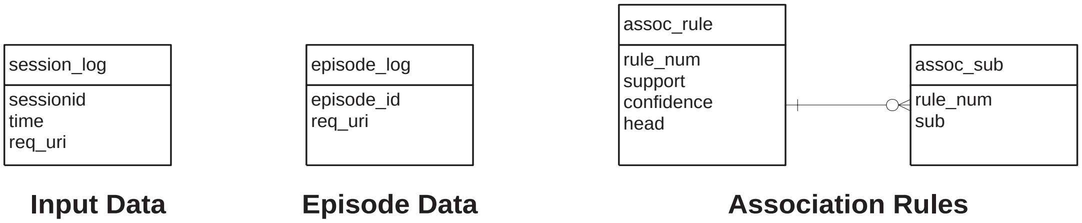


Figure 8.8: Schema for Episode Identification Experiment

Methods

The input for this experiment was the output of the session identification task of the usage preprocessing step. The *session_log* table was created by parsing a CLF log, removing all graphics requests, and running the session identification heuristic. The episode identification methods described in Chapter 5 were run on the *session_log* table to create four versions of the *episode_log* table. Media-only and Auxiliary-Media episodes were identified using the reference length and maximal forward reference methods. An auxiliary percentage parameter of 65% was used for the reference length method. The Apriori algorithm was then run on each of the *episode_log* tables with thresholds of 0.1% support and 20% confidence. The discovered rules were manually inspected for subjective interestingness.

Results

1150 association rules were discovered for the reference length episodes and 398 for the maximal forward reference episodes. Table 8.6 shows some examples of the discovered association rules.

Table 8.6: Examples of Association Rules from Global Reach

Approach Used	Conf.(%)	Supp.(%)	Association Rules
Reference Length (media-only)	61.54	0.18	`/mti/clinres.htm /mti/new.htm` ⇒ `/mti/prodinfo.htm`
Reference Length (media-only)	100.00	0.15	`/mti/pubs.htm /mti/prodinfo.htm /mti/Q&A.htm` ⇒ `/mti/clinres.htm`
Reference Length (media-only)	26.09	0.14	`/cyprus-online/dailynews.htm` ⇒ `/mti/Q&A.htm`
Maximal Forward Reference (media-only)	52.17	0.14	`/cyprus-online/Magazines.htm /cyprus-online/Radio.htm` ⇒ `/cyprus-online/News.htm`
Maximal Forward Reference (aux-media)	73.50	1.32	`/mti/clinres.htm /mti/new.htm` ⇒ `/mti/prodinfo.htm`

Discussion

The first two rules shown in Table 8.6 are straight-forward association rules that could have been predicted by looking at the structure of the web site. However, the third rule shows an unexpected association between a page of the *cyprus-online* site and a page from the *MTI* site. Approximately one fourth of the users visiting `/cyprus-online/dailynews.htm` also chose to visit `/mti/Q&A.htm` during the same session. However, since the *cyprus-online* page no longer exists on the Global Reach server, it is not clear if the association is the result of an advertisement, a link to the *MTI* site, or some other factor. The fourth rule listed in Table 8.6 is one of the 150 rules that the maximal forward reference approach discovered that was not discovered by the reference length approach. While the reference length approach discovered many rules involving the *MTI* web site, the maximal forward reference approach discovered relatively few rules involving the *MTI* site. An inspection of the *MTI* site revealed that the site is a fully connected graph. Consistent with the results of the previous experiment, the maximal forward reference approach does not perform well under these conditions. The association rule algorithm was run with auxiliary-media episodes created from the maximal forward reference approach to confirm the theory that the rules missed by the media-only episodes would be discovered. The last rule listed in Table 8.6 is the same as the first rule listed, and shows that the auxiliary-media episodes from the maximal forward reference approach can discover rules in a highly connected graph. However, at thresholds of 0.1% support and 20% confidence, approximately 25,000 other rules were also discovered with this approach.

Chapter 9

Pattern Analysis Experiments

The predominant theme that has been repeated throughout this thesis is that the majority of the patterns discovered through Web Usage Mining are uninteresting, and that a method of identifying *subjectively interesting* patterns is needed. This chapter first tests the statement that discovered patterns with the highest objective parameters are often uninteresting. The information filter described in Chapter 7 is then used to rank discovered patterns based on subjective interestingness. The Large E-commerce and Computer Science data sets described in the previous chapter are used for all of the experiments presented in this chapter.

9.1 Frequent Itemsets

The test hypothesis for this experiment is that use of page view identifiers or single frames from a multi-framed site will result in fewer non-trivial discovered patterns than the use of all of the page files, as discussed in Chapter 6. A non-trivial pattern is one that contains items from more than one page view. A server log from the Large E-commerce site is used for this experiment. In addition, the hypothesis that itemsets with the highest objective measures are subjectively uninteresting will also be tested.

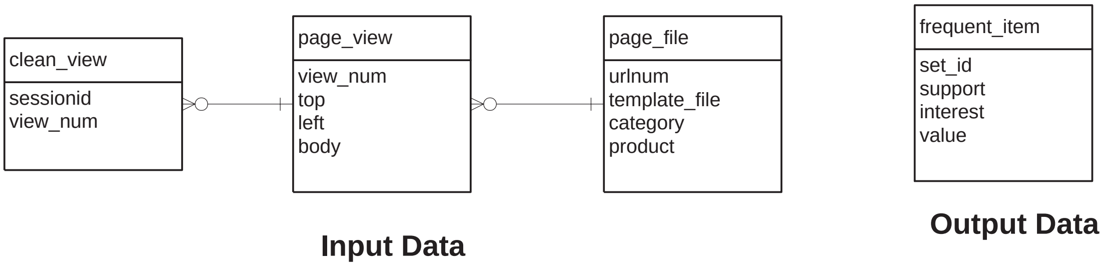


Figure 9.1: Schema for Frequent Itemsets

Methods

The input for this experiment, shown in Figure 9.1, is the output of the page view identification task for usage preprocessing. The data cleaning and session identification tasks were performed by parsing an ECLF log, removing all graphics requests, removing all log entries that did not contain an embedded session ID, and setting the sessionid field equal to the sessionid CGI value for the request. This included removing a number of hidden POST requests, since the embedded session ID was not recorded in the Server log. The *template_file*, *category*, and *product* CGI values for both the request and referrer were replaced with the appropriate numeric identifier from the *page_file* table. The page view identification algorithm described in Chapter 5 was then run on the *session_log* table. All of the page views for the Large E-commerce site have three frames, so each page view identified has a top, left, and body page file. The path completion heuristic was not called since the Apriori algorithm does not require the time or number of occurrences of an item within a session.

As discussed in Chapter 6, patterns can be based on the page view identifiers, page files from a single frame, or page files from multiple frames. For page view patterns, the *clean_view* data are the input to the pattern discovery algorithms. For

Table 9.1: Large E-commerce Site Frequent Itemsets.

Pattern Type	Total Number of Patterns	Number of Non-subset Patterns	Number of Distinct Page Files
Page View	5691	110	69
Left Frame	442	49	35
Body Frame	1173	89	43
Both Frames	76,655	121	78

single and multi-frame patterns, the *clean_view* data are joined with the *page_view* data in order to access the individual components of the page views. The final table, *page_file* is used to translate the identifiers back into human understandable labels once the patterns have been discovered.

A 1% support threshold was used to generate frequent itemsets for the Large E-commerce site with the Apriori algorithm for the page views, left frame, body frame, and both the left and body frame. The top frame was left out of the experiments since there were only two distinct page files for the Large E-commerce top frame location, and the values of the left and body frames uniquely determined the value of the top frame.

Results

The number of discovered patterns for each input type is shown in Table 9.1. Since any $\mathtt{n}$ item frequent itemset has $\sum_{x=1}^{n-2} \binom{n}{n-x}$ frequent subsets, the number of sets that are not subsets of any other frequent itemset are also shown.

Tables 9.2, 9.3, 9.4, and 9.5 show the top ten discovered rules ordered by support with an interest of at least one standard deviation above the average interest for all discovered non-subset itemsets[1].

[1] Less than ten rules are displayed for the cases where ten rules with sufficient interest were not

Table 9.2: Page View Frequent Itemsets.

Page View	Support	Interest
Home Page, Sweepstakes Home Sweepstakes Entry	4.29	21.7
Home Page, Main Domestics	2.27	19.6
Home Page, Main Jewelry	2.00	27.9
Home Page, Main Apparel	1.91	24.5
Home Page, Main Home Accents	1.90	22.3
Home Page, Main Home Accents, Furniture	1.39	22.4
Home Page, Main Kids, Video Games, Nintendo	1.31	576.2
Holiday Sweepstakes, Holiday Sweepstakes Entry	1.17	65.2
Home Page, Main Sports	1.13	39.2
Order 1, Order 2, Order 3 Order 4, Order 5	1.10	3.23×10^6

Table 9.3: Left Frame Frequent Itemsets.

Left Frame	Support	Interest
Main Categories, Home Accents, Furniture	1.75	21.1
Main Categories, Jewelry, Gold	1.20	26.3
Main Categories, Sporting Goods, Recreation	1.16	37.0
Main Categories, Apparel, Coats	1.09	23.0
Main Categories, Apparel, LeftOrder1, LeftOrder2, LeftOrder3	1.05	250.6

Table 9.4: Body Frame Frequent Itemsets.

Body Frame	Support	Interest
Main Body, Sweepstakes Entry, Sweepstakes Home	4.48	20.6
Main Body, Catalog Order, Order1 Order2, Order3, Order4, Order5	2.21	62591
Main Body, Domestics Body, Home Accents Body	1.93	7.6
Main Body, Holiday Sweepstakes, Holiday Sweepstakes Entry	1.76	43.2
Main Body, Domestics Body, Housewares Body	1.72	6.8
Main Body, Home Accents Body, Housewares Body	1.64	7.3
Main Body, Shipping Chart, Order1 Order2, Order3	1.37	561.8
Main Body, Shopping Cart, Order1, Order2 Order3, Order4, Order5	1.34	94687

Table 9.5: Both Frame Frequent Itemsets.

Frame	Support	Interest
Home Page, Holiday Sweepstakes, Holiday Sweepstakes Entry	1.75	43.5
Home Page, Members Home, Member Signup, Member Sign In, Order Status	1.74	284.8
Home Page, Kids Main, Video Games, Nintendo	1.66	5313.9
Home Page, Shopping Cart, Order1, Order2 Order3, Order4, Order5	1.33	1.9×10^{10}
Home Page, Main Electronics, Kids, Video Games, Video Games	1.24	6719.3
Home Page, Main Jewelry, Gold	1.20	703.1
Home Page, Kids, Video Games, Handheld	1.18	5305.2
Home Page, Member Sign In, Member Sign Up Order1, Order2, Order3	1.16	2795421
Home Page, Sporting Goods, Recreation	1.16	1387.8
Home Page, Main Apparel, Coats	1.09	535.6

Discussion

Manual inspection shows that none of the discovered rules are particularly interesting from a subjective point of view. Most of the frequent itemsets confirm that the Home Page and one of the high level categories are related through usage. The order stream also appears in a few of the rules. The use of page view identifiers ends up “hiding” a number of patterns due to the fragmentation of page files among several page views, confirming the test hypothesis. Even though the number of non-subset patterns is relatively similar for the page view and two frame itemsets, the two frame itemsets contain more items on average.

9.2 Page File Frequent Itemsets versus Structure

The test hypothesis for these experiments is that Web site structure can be used rank frequent itemsets based on subjective interestingness. Two preliminary experiments were run on the static Computer Science data, and an experiment using the full subjective information filter was run on the Large E-commerce data.

9.2.1 Unconnected Components in Frequent Itemsets

To test the feasibility of filtering discovered rules based on site structure, a simple test was run on frequent itemsets discovered from the Computer Science data. The hypothesis is that frequent itemsets that represent unconnected components are subjectively interesting.

present in the results.

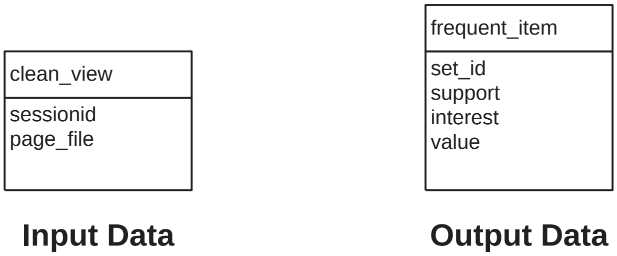


Figure 9.2: Schema for Preliminary Experiments

Methods

The input for this experiment was the output of the session identification task for usage preprocessing. Data cleaning and session identification were performed by parsing an ECLF log, removing all graphics requests, and running the session identification heuristic. The *clean_view* table shown in Figure 9.2 was then created by selecting the distinct page files referenced in each session. Since the Computer Science site is predominantly static, the URI served as the content identifier for each page file.

The Apriori algorithm was run to discover frequent itemsets with a support threshold of 0.1%. Page view component identification was performed manually on the discovered frequent itemsets. Any trivial frequent itemsets (itemsets that contain nothing but components of the same page view) were removed. All frequent itemsets that were a subset of a larger frequent itemset were also filtered out prior to checking for the connectivity of the components. Each frequent itemset was declared to be connected if the edge connectivity of the set of items was at least one, as defined in Chapter 7. Any itemset with an edge connectivity of zero was declared to be potentially interesting.

Table 9.6: Unconnected Frequent Itemsets

#	Mined Support(%)	Related Pages
1	0.10	/Research/, /tech_reports/
2	0.10	/employment/, /newsletter/
3	0.10	/faculty/, /newsletter/
4	0.10	/icra99/ICRA99-Index.htm, /icra99/Notice.html, /icra99/TechnProgram.htm, /icra99/advprogram2.htm
5	0.10	/new/, /sem-coll/
6	0.10	/reg-info/98-99_schedule.html, /reg-info/ss1-99.html, /reg-info/ss2-99.html
7	0.11	/Research/Agassiz/, /faculty/
8	0.11	/icra99/Notice.html, /icra99/best.html
9	0.11	/icra99/Proceeding-Order.htm, /icra99/Registration.htm
10	0.22	/grad-info/, /grad-info/97-98-grad-handbook.html
11	0.25	/grad-info/, /grad-info/96-97-grad-handbook.html

Results

693 frequent itemsets were discovered, with a maximum set size of six pages. There were 178 unique pages represented in all of the rules. 11 frequent itemsets were identified as potentially interesting, as shown in Table 9.6.

Discussion

Of the frequent itemsets shown in Table 9.6, the two including the graduate handbook (numbers 10 and 11) are of note because these pages are out-of-date. A page with the 1998-99 graduate handbook exists, and the links from the /grad-info/ page to the older handbooks have been removed. However, since the pages were not actually removed from the site and other pages in the site reference them (or users have old bookmarks), the older handbooks are still accessed. The support of these itemsets are 0.25% and 0.22% respectively. Had the support threshold been set higher to limit the total number of itemsets discovered, the rules would have been missed.

9.2.2 Connected Components without Frequent Itemsets

This experiment tests the hypothesis that connected components with high individual support and no discovered frequent itemset are also subjectively interesting.

Methods

The input for this experiment was the output of the session identification task for usage preprocessing. Data cleaning and session identification were performed by parsing an ECLF log, removing all graphics requests, and running the session identification heuristic. The *clean_view* table shown in Figure 9.2 was then created by selecting the distinct page files referenced in each session. Since the Computer Science site is predominantly static, the URI served as the content identifier for each page file.

The Apriori algorithm was run to discover frequent itemsets with a support threshold of 0.1%. Page view component identification was performed manually on the discovered frequent itemsets. Any trivial frequent itemsets (itemsets that contain nothing but components of the same page view) were removed. All of the connected pairs of pages that had sufficient individual support were compared with the two-item frequent itemsets. Pairs of pages that did not have a corresponding frequent itemset were declared to be interesting.

Results

693 frequent itemsets were discovered, with a maximum set size of six pages. There were 178 unique pages represented in all of the rules. 10 missing page pairs were identified as potentially interesting, as shown in Table 9.7.

Table 9.7: Connected Items without a Frequent Itemset

#	Web Pages
1	/Research/Agassiz/agassiz_pubs.html,/Research/Agassiz/agassiz_people.html
2	/Research/GIMME/tclprop.html, /Research/GIMME/Nsync.html
3	/Research/airvl/minirob.html, /Research/airvl/loon.html
4	/Research/mmdbms/home.shtml, /Research/mmdbms/group.html
5	/newsletter/kumar.html, /newsletter/facop.html
6	/newsletter/letter.html, /newsletter/facop.html
7	/newsletter/letter.html, /newsletter/kumar.html
8	/newsletter/newfac.html, /newsletter/facop.html
9	/newsletter/newfac.html, /newsletter/kumar.html
10	/newsletter/newfac.htm, /newsletter/letter.html

Discussion

In Table 9.7, the fourth pair of pages is of note because the first page functions solely as an entry page to a particular research group's pages. However, the link from the first page is flashing and located fairly low on the page. This suggests a design problem since not all of the visitors from the first page are visiting the second. These results confirm the hypothesis that pages used together less than would be expected from the site structure can be subjectively interesting.

9.2.3 Information Filter

The test hypothesis for this experiment is that frequent itemsets can be ranked according to subjective interestingness using the information filter described in Chapter 7 and the structure of a Web site.

Methods

The input for this experiment, shown in Figure 9.1, is the output of the page view identification task for usage preprocessing. The data cleaning and session identification tasks were performed by parsing an ECLF log, removing all graphics requests, removing all log entries that did not contain an embedded session ID, and setting the sessionid field equal to the sessionid CGI value for the request. This included removing a number of hidden POST requests, since the embedded session ID was not recorded in the Server log. The *template_file*, *category*, and *product* CGI values for both the request and referrer were replaced with the appropriate numeric identifier from the *page_file* table. The page view identification algorithm described in Chapter 5 was then run on the *session_log* table. All of the page views for the Large E-commerce site have three frames, so each page view identified had a top, left, and body page file. The path completion heuristic was not called since the Apriori algorithm does not require the time or number of occurrences of an item within a session.

A 0.5% support threshold was used to generate frequent itemsets with the Apriori algorithm for both the left and body frame of the Large E-commerce site data. The top frame was left out of the experiments since there were only two distinct page files for the Large E-commerce top frame location, and the values of the left and body frames uniquely determined the value of the top frame. The non-trivial, non-subset itemsets were input into the subjective information filter and compared against the structure of the Web site. The subjective interestingness of each pattern was calculated as described in Chapter 7 using a Java program accessing the data through a JDBC connection.

Table 9.8: Frequent Itemsets with High Subjective Interestingness.

Page View	Support	Maximum Confidence	Interest	Subjective Interest
Home Page, Main Holiday, Holiday Sweepstakes	0.55	10.2	10	1.26
Home Page, Main Holiday, Sweepstakes	0.56	11.5	2	1.23
Home Page, Main Housewares, Main Holiday	0.50	13.0	50	1.19
Home Page, Main Holiday, Member Order Status	0.51	9.35	21	1.18
Home Page, Main Domestics, Main Housewares, Main Electronics	0.50	12.44	25484	1.14

Results

155,951 frequent itemsets were discovered by the experiment, of which 209 were non-trivial and non-subset itemsets. The five itemsets with the highest subjective interestingness are shown in Table 9.8 and the five itemsets with the lowest subjective interestingness are shown in Table 9.9.

Discussion

All but one of the frequent itemsets represented connected pages, resulting in most of the patterns being assigned a low subjective interestingness measure. However, the one unconnected frequent itemset was declared to be the least interesting of all the

Table 9.9: Frequent Itemsets with Low Subjective Interestingness.

Page View	Support	Maximum Confidence	Interest	Subjective Interest
Home Page, Main Electronics, Computers, Peripherals	0.59	95.4	11281	0.58
Home Page, Main Home Accents, Wall Decor, Misc.	0.56	96.8	100547	0.58
Home Page, Main Kids, Video Games, Handheld	0.57	98.6	31025	0.58
Home Page, Main Kids, Video Games, Nintendo	0.87	98.8	31102	0.58
Main Holiday, Main Domestics, Main Home Accents	0.50	13.0	41	0.32

discovered patterns. This is because the maximum confidence for the itemset was quite low, meaning that more often than not, the pages were not accessed together as expected due to the lack of connectivity. The rules with the highest subjective interestingness are all rules that have much lower confidence than would be expected due to the connectivity of the pages. The first two rules indicate that there is not very much cross-traffic between the sweepstakes pages and the Home Page. It turns out that there is a direct site entry point into the sweepstakes pages. A possible explanation for the low confidence is that users are coming to the site to register for the sweepstakes, and then leaving without browsing the site. This could be confirmed by performing sequential pattern analysis on the Home Page and sweepstakes pages. The last three rules are also potentially interesting due to a lack of cross-traffic between the page views. It is not immediately clear why this is the case for the third and fourth patterns, but a possible explanation for the fifth pattern is that users that browse Housewares and Domestics are not interested in Electronics (or vice-versa). The first four itemsets in Table 9.9 are all levels of a single category hierarchy. The last itemset is the one unconnected itemset already mentioned. Note that four of the five subjectively interesting patterns have a very low objective interest measure, and they all have low support and confidence. The Table 9.9 patterns have just the opposite characteristics - the confidence and objective interest measures are quite high.

9.3 Product Episode Clusters versus Content

The experiments in this section were designed to test the ability of the information filter to rank clusters according to subjective interestingness based on the content of a Web site.

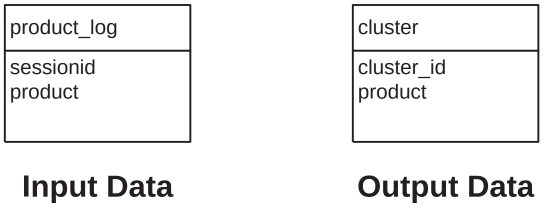


Figure 9.3: Schema for Product Cluster Experiments

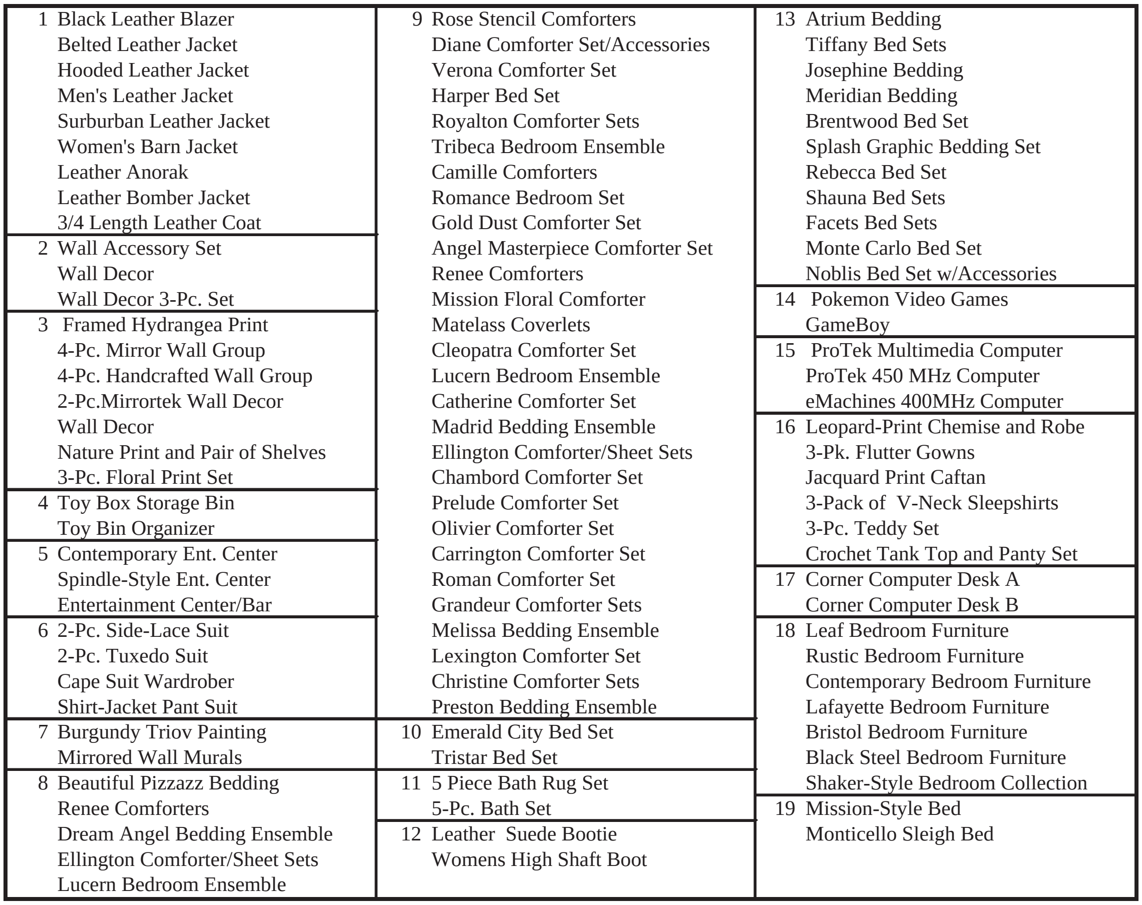
9.3.1 Hypergraph Product Clusters

The test hypothesis for this experiment is that product clusters discovered through the hypergraph clustering method described in Chapter 6 can be ranked according to subjective interestingness based on the content hierarchy of a Web site.

Methods

The input and output schema for this experiment are shown in Figure 9.3. Data cleaning and session identification were performed by parsing an ECLF log, removing all graphics requests, removing all log entries that did not contain an embedded session ID, and setting the sessionid field equal to the sessionid CGI value for the request. This included removing a number of hidden POST requests, since the embedded session ID was not recorded in the Server log. All product pages for the Large E-commerce site are identified by a product code contained in the *product* CGI variable for a URI. The *product_log* table was created by selecting the unique non-null product values for each session ID.

The Apriori algorithm was run on the *product_log* table with a support threshold of 0.3% to discover frequent itemsets. The non-subset frequent itemsets were then input into the Hypergraph clustering algorithm described in Chapter 6 to create product clusters. A cutoff parameter of 0.5, connectivity parameter of 0.01 and an overlap



1	Black Leather Blazer Belted Leather Jacket Hooded Leather Jacket Men's Leather Jacket Surburban Leather Jacket Women's Barn Jacket Leather Anorak Leather Bomber Jacket 3/4 Length Leather Coat
2	Wall Accessory Set Wall Decor Wall Decor 3-Pc. Set
3	Framed Hydrangea Print 4-Pc. Mirror Wall Group 4-Pc. Handcrafted Wall Group 2-Pc.Mirrortek Wall Decor Wall Decor Nature Print and Pair of Shelves 3-Pc. Floral Print Set
4	Toy Box Storage Bin Toy Bin Organizer
5	Contemporary Ent. Center Spindle-Style Ent. Center Entertainment Center/Bar
6	2-Pc. Side-Lace Suit 2-Pc. Tuxedo Suit Cape Suit Wardrober Shirt-Jacket Pant Suit
7	Burgundy Triov Painting Mirrored Wall Murals
8	Beautiful Pizzazz Bedding Renee Comforters Dream Angel Bedding Ensemble Ellington Comforter/Sheet Sets Lucern Bedroom Ensemble
9	Rose Stencil Comforters Diane Comforter Set/Accessories Verona Comforter Set Harper Bed Set Royalton Comforter Sets Tribeca Bedroom Ensemble Camille Comforters Romance Bedroom Set Gold Dust Comforter Set Angel Masterpiece Comforter Set Renee Comforters Mission Floral Comforter Matelass Coverlets Cleopatra Comforter Set Lucern Bedroom Ensemble Catherine Comforter Set Madrid Bedding Ensemble Ellington Comforter/Sheet Sets Chambord Comforter Set Prelude Comforter Set Olivier Comforter Set Carrington Comforter Set Roman Comforter Set Grandeur Comforter Sets Melissa Bedding Ensemble Lexington Comforter Set Christine Comforter Sets Preston Bedding Ensemble
10	Emerald City Bed Set Tristar Bed Set
11	5 Piece Bath Rug Set 5-Pc. Bath Set
12	Leather Suede Bootie Womens High Shaft Boot
13	Atrium Bedding Tiffany Bed Sets Josephine Bedding Meridian Bedding Brentwood Bed Set Splash Graphic Bedding Set Rebecca Bed Set Shauna Bed Sets Facets Bed Sets Monte Carlo Bed Set Noblis Bed Set w/Accessories
14	Pokemon Video Games GameBoy
15	ProTek Multimedia Computer ProTek 450 MHz Computer eMachines 400MHz Computer
16	Leopard-Print Chemise and Robe 3-Pk. Flutter Gowns Jacquard Print Caftan 3-Pack of V-Neck Sleepshirts 3-Pc. Teddy Set Crochet Tank Top and Panty Set
17	Corner Computer Desk A Corner Computer Desk B
18	Leaf Bedroom Furniture Rustic Bedroom Furniture Contemporary Bedroom Furniture Lafayette Bedroom Furniture Bristol Bedroom Furniture Black Steel Bedroom Furniture Shaker-Style Bedroom Collection
19	Mission-Style Bed Monticello Sleigh Bed

Figure 9.4: Product Clusters from Hypergraph Partitioning

parameter of 0.6 were used for the Hypergraph clustering algorithm.

Results

With a minimum support threshold of 0.3%, 644 itemsets were discovered. 118 distinct products were represented in the frequent itemsets. The output of the Hypergraph clustering algorithm produced 19 clusters, as shown in Figure 9.4.

Discussion

All of the 19 clusters listed in Figure 9.4 contain products from a single low level category. There are no cross-category relationships identified by the frequent itemset clusters. The information filter does rank clusters 2, 3, and 7 as the most interesting because they are all products from the same low level category - Miscellaneous Wall Decor. For some reason, there are three distinct subsets of users who view certain products with in the Wall Decor category. Similarly, clusters 8 and 9 are listed as slightly interesting since they are both from the Comforter category in Domestics. The rest of the clusters match case 1 or 2 from Figure 7.7, where there is complete agreement between the usage cluster and the content category.

The disadvantage of the Hypergraph clustering algorithm is that it limits the discovered items and clusters to those that are frequently occurring and tightly connected. While this produces very clean clusters that represent the behaviors of a significant amount of the population (as defined by the minimum support criteria), many products that are viewed do not end up in any cluster. The Apriori algorithm depends on the minimum support threshold to prevent the computations from becoming intractable. At 0.2% support, the Apriori algorithm was allowed to run for 24 hours, and was unable to complete the frequent itemset generation process. Due to the small number of discovered clusters, the use of Hypergraph clustering to test the hypothesis that information filter can rank clusters according to subjective interestingness is inconclusive.

9.3.2 Concept Clusters

In order to get clusters that include some less frequently occurring products, the *Concept Indexing* method described in [62] was used on the product episodes. The test hypothesis for this experiment is that product clusters discovered through the concept clustering method can be ranked according to subjective interestingness based on the content hierarchy of a Web site.

Methods

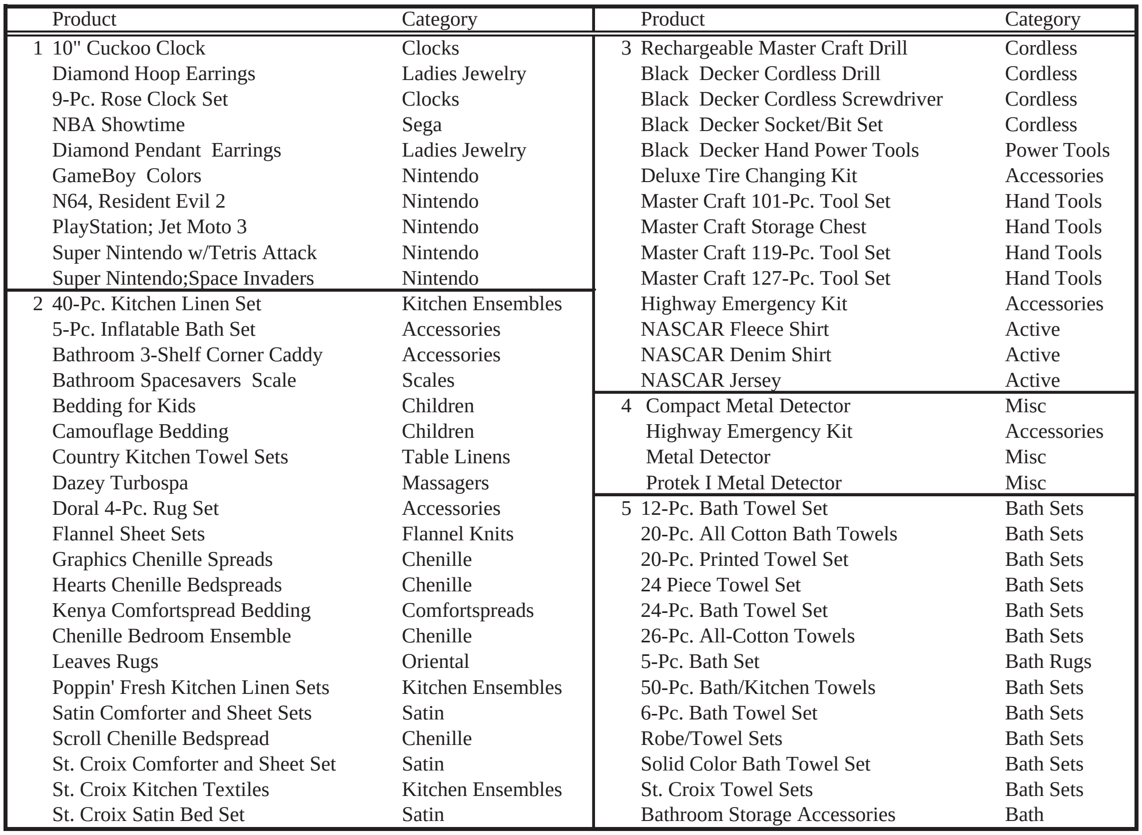
The input and output schema for this experiment is shown in Figure 9.3. Data cleaning and session identification were performed by parsing an ECLF log, removing all graphics requests, removing all log entries that did not contain an embedded session ID, and setting the sessionid field equal to the sessionid CGI value for the request. This included removing a number of hidden POST requests, since the embedded session ID was not recorded in the Server log. All product pages for the Large E-commerce site are identified by a product code contained in the *product* CGI variable for a URI. The *product_log* table was created by selecting the unique non-null product values for each session ID. The [62] concept clustering method was used to discover product clusters from the *product_log* table.

Results

127 multi-product clusters were discovered containing 1228 unique products. 27 of the 127 clusters were equally ranked with the highest amount of interestingness. For all 27 clusters, the minimum local root was the global root for the entire site. Figure 9.5 shows five of these clusters. 55 clusters contained products from the same low level category, and 21 clusters contained products from the same second-level category. In most cases, the height of the content hierarchy was four.

Discussion

Most of the 27 interesting clusters are similar to the first cluster listed in Figure 9.5, where a quick manual inspection is sufficient to notice the difference in the listed



	Product	Category		Product	Category
1	10" Cuckoo Clock	Clocks	3	Rechargeable Master Craft Drill	Cordless
	Diamond Hoop Earrings	Ladies Jewelry		Black Decker Cordless Drill	Cordless
	9-Pc. Rose Clock Set	Clocks		Black Decker Cordless Screwdriver	Cordless
	NBA Showtime	Sega		Black Decker Socket/Bit Set	Cordless
	Diamond Pendant Earrings	Ladies Jewelry		Black Decker Hand Power Tools	Power Tools
	GameBoy Colors	Nintendo		Deluxe Tire Changing Kit	Accessories
	N64, Resident Evil 2	Nintendo		Master Craft 101-Pc. Tool Set	Hand Tools
	PlayStation; Jet Moto 3	Nintendo		Master Craft Storage Chest	Hand Tools
	Super Nintendo w/Tetris Attack	Nintendo		Master Craft 119-Pc. Tool Set	Hand Tools
	Super Nintendo;Space Invaders	Nintendo		Master Craft 127-Pc. Tool Set	Hand Tools
2	40-Pc. Kitchen Linen Set	Kitchen Ensembles		Highway Emergency Kit	Accessories
	5-Pc. Inflatable Bath Set	Accessories		NASCAR Fleece Shirt	Active
	Bathroom 3-Shelf Corner Caddy	Accessories		NASCAR Denim Shirt	Active
	Bathroom Spacesavers Scale	Scales		NASCAR Jersey	Active
	Bedding for Kids	Children	4	Compact Metal Detector	Misc
	Camouflage Bedding	Children		Highway Emergency Kit	Accessories
	Country Kitchen Towel Sets	Table Linens		Metal Detector	Misc
	Dazey Turbospa	Massagers		Protek I Metal Detector	Misc
	Doral 4-Pc. Rug Set	Accessories	5	12-Pc. Bath Towel Set	Bath Sets
	Flannel Sheet Sets	Flannel Knits		20-Pc. All Cotton Bath Towels	Bath Sets
	Graphics Chenille Spreads	Chenille		20-Pc. Printed Towel Set	Bath Sets
	Hearts Chenille Bedspreads	Chenille		24 Piece Towel Set	Bath Sets
	Kenya Comfortspread Bedding	Comfortspreads		24-Pc. Bath Towel Set	Bath Sets
	Chenille Bedroom Ensemble	Chenille		26-Pc. All-Cotton Towels	Bath Sets
	Leaves Rugs	Oriental		5-Pc. Bath Set	Bath Rugs
	Poppin' Fresh Kitchen Linen Sets	Kitchen Ensembles		50-Pc. Bath/Kitchen Towels	Bath Sets
	Satin Comforter and Sheet Sets	Satin		6-Pc. Bath Towel Set	Bath Sets
	Scroll Chenille Bedspread	Chenille		Robe/Towel Sets	Bath Sets
	St. Croix Comforter and Sheet Set	Satin		Solid Color Bath Towel Set	Bath Sets
	St. Croix Kitchen Textiles	Kitchen Ensembles		St. Croix Towel Sets	Bath Sets
	St. Croix Satin Bed Set	Satin		Bathroom Storage Accessories	Bath

Figure 9.5: Interesting Product Clusters

products. The most likely explanation for these clusters is holiday shoppers looking for gifts for a number of different people. For the other four clusters listed in Figure 9.5, it is not immediately apparent why they have been ranked as interesting. However, for each cluster, there is at least one product that is part of a completely different high level category than the rest. For cluster 2, the “Dazey Turbospa” is under Housewares and the rest of the products are under Domestics. In cluster 3, the three pieces of “NASCAR” clothing are under Apparel and for cluster 4 the metal detectors are listed under the Sporting Goods category. All of the other products from clusters 3 and 4 are listed under Hardware. For cluster 5, the last product, ”Bathroom Storage Accessories” is listed under the Home Accents hierarchy, while the other products are under Domestics. Because the site structure somewhat mirrors the content hierarchy, in each of these cases, several links had to be followed to get to the desired products in a different category. These interesting clusters indicate an opportunity to increase cross-sells for the e-commerce site by directly linking the products from different categories. The results of the concept clustering confirm that the information filter can be used in conjunction with the content of a Web site to rank clusters according to subjective interestingness.

Chapter 10

Conclusion

Web Usage Mining has become an important area of research, as evidenced by the growing number of research projects and commercial enterprises engaged in analyzing and performing pattern discovery on Web clicksteam data. Its full potential is just starting to be tapped with static aggregation, OLAP, and simple pattern analysis. The ability to observe potential customers as they browse through a virtual store promises to raise business intelligence to a new level, and at the same time increases concerns about consumer privacy. While demographic data can always be added into the Web Usage Mining process, this thesis has shown that interesting results can be generated with just the anonymous clickstream data. The discovered patterns do not depend on any personal data about the site users, only their aggregated browsing patterns.

For the task of identifying the subjectively interesting patterns from the total set of discovered patterns, the information filter was shown to be successful for several types of patterns. For the Large E-commerce site, the subjectively interesting frequent itemsets turned out to be the ones with low objective parameters, indicating a lack of expected cross-traffic between certain areas of the site. The product clusters that represent products from multiple categories indicate a potential for additional links to lead users to products that aren't necessarily related in the content hierarchy. Perhaps the greatest benefit of the subjective interestingness measure is its ability to quickly confirm that a Web site is being used as designed. While the marketing analyst may be searching for that previously unknown pattern that can be used to increase sales, the lack of unexpected results is probably an ideal result from a Web site designers point of view.

The results of the Chapter 8 experiments show that meaningful knowledge does not come simply by throwing Web logs through a generic automated process. First, a positive method of session identification is crucial to getting accurate results. Ideally, an identification method that allows for the identification of repeat visitors should be used. While this thesis focused on the patterns that can be discovered from individual sessions, the behaviors of users over time as they re-visit a site would provide a much richer data set for analysis. Next, without knowledge of the site structure, the page view identification step can not be completed. Whether a Web site is single or multi-framed has a large effect on the amount of preprocessing that must be performed and the types of filtering that will be needed for the discovered patterns. In addition, dynamic sites pose different preprocessing challenges than predominantly static Web sites.

The field of Web Usage Mining is very young, and accordingly the list of open issues is quite long. From this thesis alone, further work needs to be done to automatically determine page usage types, verify the path completion heuristic, and develop and test further quantification methods for the information filter. The effect of the difference between logged and actual page view times needs to be investigated. Also, the pattern analysis results need to be tailored to the various application areas such as usability testing, personalization, and business intelligence.

As the handling of the page files moves away from Web servers and towards the content and application servers, the location for automated data collection needs to be moved. For the Large E-commerce site analyzed in Chapters 8 and 9, an extraordinary amount of work went into preprocessing the raw Web server log in order to infer information that would have been readily available from the content server used by the site. Also, the task of URI translation, which is now a predominantly manual process, could be easily automated from within a content server, since it is responsible for the content creation.

Ultimately, whether the clickstream data comes from the client, Web server, or content server, effective Web Usage Mining will continue to grow as an important tool for analyzing, optimizing, and personalizing Web sites.

Bibliography

[1] A standard for robot exclusion. http://info.webcrawler.com/mak/projects/robots/norobots.html, 1994.

[2] Global reach internet productions. http://www.global-reach.com, 1997.

[3] Accrue insight. http://www.accrue.com, 1999.

[4] Alladvantage. http://www.alladvantage.com, 1999.

[5] Andromedia aria. http://www.andromedia.com, 1999.

[6] Broadvision. http://www.broadvision.com, 1999.

[7] Funnel web professional. http://www.activeconcepts.com, 1999.

[8] Hit list commerce. http://www.marketwave.com, 1999.

[9] Likeminds. http://www.andromedia.com, 1999.

[10] Netgenesis. http://www.netgenesis.com, 1999.

[11] Netperceptions. http://www.netperceptions.com, 1999.

[12] Netzero. http://www.netzero.com, 1999.

[13] Surfaid analytics. http://surfaid.dfw.ibm.com, 1999.

[14] Webtrends log analyzer. http://www.webtrends.com, 1999.

[15] World wide web committee web usage characterization activity. http://www.w3c.org/WCA, 1999.

[16] Alexa internet. http://www.alexa.com, 2000.

[17] Blue martini inc. http://www.bluemartini.com, 2000.

[18] Charu C Aggarwal and Philip S Yu. On disk caching of web objects in proxy servers. In *CIKM 97*, pages 238–245, Las Vegas, Nevada, 1997.

[19] R. Agrawal. Data mining: Crossing the chasm. Invited talk at the 5th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining(KDD99), 1999.

[20] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.

[21] Rakesh Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zait. Querying shapes of histories. In *21st Int'l Conference on Very Large Databases*, 1995.

[22] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the www. Technical Report TR-96-11, Boston University, 1996.

[23] Martin F Arlitt and Carey L Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5):631–645, 1997.

[24] M. Balabanovic and Y. Shoham. Learning information retrieval agents: Experiments with automated web browsing. In *On-line Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments*, 1995.

[25] J. F. Baldwin. Evidential support logic programming. *Fuzzy Sets and Systems*, 24(1):1–26, 1987.

[26] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state. *Ann. Math. Stat.*, 37:1554–1563, 1966.

[27] P. P. Bonissone. Summarizing and propagating uncertain information with triangular norms. *International Journal of Approximate Reasoning*, 1:71–101, 1987.

[28] Piero P. Bonissone and Karsten S. Decker. Selecting uncertainty calculi and granularity: An experiment in trading-off precision and complexity. *Uncertainty in Artificial Intelligence*, pages 2217–2247, 1986.

[29] B. Boser, M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Conference on Computational Learning Theory (COLT)*, pages 144–152, 1992.

[30] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (xml) 1.0 w3c recommendation. Technical report, W3C, 1998.

[31] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *ACM SIGMOD International Conference on Management of Data*, 1997.

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

[33] Alex Buchner and Maurice D Mulvenna. Discovering internet marketing intelligence through online analytical web usage mining. *SIGMOD Record*, 27(4):54–61, 1998.

[34] L. Catledge and J. Pitkow. Characterizing browsing behaviors on the world wide web. *Computer Networks and ISDN Systems*, 27(6), 1995.

[35] 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*, pages 385–392, 1996.

[36] V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. John Wiley and Sons, Inc., New York, 1998.

[37] Ed Chi, James Pitkow, Jock Mackinlay, Peter Pirolli, Rich Gossweiler, and Stuart Card. Visualizing the evolution of web ecologies. In *CHI 98*, pages 400–407, Los Angeles, CA, 1998.

[38] Chris Clifton and Robert Cooley. Topcat: Data mining for topic identification in a text corpus. In *Principles of Knowledge Discovery in Databases*, 1999.

[39] E. Cohen, B. Krishnamurthy, and J. Rexford. Improving end-to-end performance of the web using server volumes and proxy filters. In *Proceedings of ACM SIGCOMM*, pages 241–253, 1998.

[40] Robert Cooley. Classification of news stories using support vector machines. In *International Joint Conference on Artificial Intelligence Text Mining Workshop*, Stockholm, Sweden, 1999.

[41] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Grouping web page references into transactions for mining world wide web browsing patterns. In *Knowledge and Data Engineering Workshop*, pages 2–9, Newport Beach, CA, 1997. IEEE.

[42] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Web mining: Information and pattern discovery on the world wide web. In *International Conference on Tools with Artificial Intelligence*, pages 558–567, Newport Beach, 1997. IEEE.

[43] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1), 1999.

[44] Robert Cooley, Pang-Ning Tan, and Jaideep Srivastava. Websift: The web site information filter system. In *WEBKDD*, San Diego, CA, 1999.

[45] Robert Cooley, Pang-Ning Tan, and Jaideep Srivastava. Discovery of interesting usage patterns from web data. In Myra Spiliopoulou, editor, *LNCS/LNAI Series*. Springer-Verlag, 2000.

[46] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. Learning to extract symbolic knowledge from the world wide web. In *National Conference on Artificial Intelligence (AAAI)*, 1998.

[47] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 1999.

[48] A. P Dempster. Comment. *Journal of the American Statistical Association*, 77:339–341, 1982.

[49] J. Doyle. A truth maintenance system. *Aritificial Intelligence*, 12(3), 1979.

[50] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 53–62, San Diego, CA, 1999.

[51] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Proceedings of ACM KDD*, 1994.

[52] J. H. Friedman. Flexible nearest neighbor classification. Technical report, Stanford University, 1994.

[53] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Inferring web communities from link topology. In *Conference on Hypertext and Hypermedia*. ACM, 1998.

[54] Valery Guralnik and Jaideep Srivastava. Event detection from time series data. In *5th International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 1999.

[55] Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar. Clustering based on association rule hypergraphs. In *SIGMOD'97 Workshop on Research Issues on Data Mining and Knowledge Discovery*. ACM, 1997.

[56] Robert J. Hilderman and Howard J. Hamilton. Knowledge discovery and interestingness measures: A survey. Technical report, University of Regina, 1999.

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

[58] Bernardo Huberman, Peter Pirolli, James Pitkow, and Rajan Kukose. Strong regularities in world wide web surfing. Technical report, Xerox PARC, 1998.

[59] 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.

[60] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)*, 1998.

[61] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekar. Multilevel hypergraph partitioning: Applications in vlsi domain. In *ACM/IEEE Design Automation Conference*, 1997.

[62] George Karypis and Eui-Hong (Sam) Han. Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval and categorization. Technical Report TR-00-0016, University of Minnesota, 2000.

[63] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley and Sons, 1990.

[64] Ralph Kimball and Richard Merz. *The Data Webhouse Toolkit*. John Wiley and Sons, Inc., 2000.

[65] Kevin Larson and Mary Czerwinski. Web page design: Implications of memory, structure and scent for information retrieval. In *CHI 1998*, Los Angeles, CA, 1998.

[66] H. Lieberman. Letizia: An agent that assists web browsing. In *Proc. of the 1995 International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.

[67] Bing Liu, Wynne Hsu, and Shu Chen. Using general impressions to analyze discovered classification rules. In *Third International Conference on Knowledge Discovery and Data Mining*, 1997.

[68] A. Luotonen. The common log file format, 1995.

[69] Stephen Lee Manley. *An Analysis of Issues Facing World Wide Web Servers.* Undergraduate, Harvard, 1997.

[70] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 210–215, Montreal, Quebec, 1995.

[71] Daniel Menasce, Virgilio Almeida, Rodrigo Fonseca, and Marco Mendes. A methodology for workload characterization of e-commerce sites. In *Electronic Commerce*, Denver, Colorado, 1999. ACM.

[72] K. Menger. Zur allgemeinen kurventheorie. *Fundementals of Mathematics*, 10:96–115, 1927.

[73] Tom Mitchell. *Machine Learning.* McGraw Hill, 1996.

[74] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Creating adaptive web sites through usage-based clustering of urls. In *Knowledge and Data Engineering Workshop*, 1999.

[75] Jerome Moore, Eui-Hong (Sam) Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, and Bamshard Mobasher. Web page categorization and feature selection using association rule and principal component clustering. In *7th Workshop on Information Technologies and Systems*, 1997.

[76] Olfa Nasraoui, Raghu Krishnapuram, and Anupam Joshi. Mining web access logs using a fuzzy relational clustering algorithm based on a robust estimator. In *Eighth International World Wide Web Conference*, Toronto, Canada, 1999.

[77] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. of the 20th VLDB Conference*, pages 144–155, Santiago, Chile, 1994.

[78] D.S.W. Ngu and X. Wu. Sitehelper: A localized agent that helps incremental exploration of the world wide web. In *6th International World Wide Web Conference*, Santa Clara, CA, 1997.

[79] Balaji Padmanabhan and Alexander Tuzhilin. A belief-driven method for discovering unexpected patterns. In *Fourth International Conference on Knowledge Discovery and Data Mining*, pages 94–100, New York, New York, 1998.

[80] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufman, San Mateo, CA, 1988.

[81] Mike Perkowitz and Oren Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998.

[82] Mike Perkowitz and Oren Etzioni. Adaptive web sites: Conceptual cluster mining. In *Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

[83] G. Piatetsky-Shapiro and C. J. Matheus. The interestingness of deviations. In *AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 25–36, 1994.

[84] Peter Pirolli, James Pitkow, and Ramana Rao. Silk from a sow's ear: Extracting usable structures from the web. In *CHI-96*, Vancouver, 1996.

[85] James Pitkow. In search of reliable usage data on the www. In *Sixth International World Wide Web Conference*, pages 451–463, Santa Clara, CA, 1997.

[86] James E Pitkow. Summary of www characterizations. In *Seventh International World Wide Web Conference*, 1998.

[87] James E. Pitkow and Colleen M Kehoe. Results from the thrid www user survey. *The World Wide Web Journal*, 1(1), 1995.

[88] J. Ross Quinlan. Consistency and plausible reasoning. In *International Joint Conference on Artificial Intelligence*, 1983.

[89] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[90] Paul Resnik, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Conference on Computer Supported Cooperative Work*. ACM, 1994.

[91] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[92] Arnaud Sahuguet and Fabien Azavant. Wysiwyg web wrapper factory (w4f), 1999.

[93] G. Salton and M. J. McGill. The smart and sire experimental retrieval systems. pages 118–155. McGraw-Hill, New York, 1983.

[94] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[95] 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.

[96] Glen Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ, 1976.

[97] Glenn Shafer. Constructive probability. *Synthese*, 48:1–60, 1981.

[98] Glenn Shafer and Amos Tversky. Languages and designs for probability judgement. *Cognitive Science*, 9:177–210, 1985.

[99] Cyrus Shahabi, Amir M Zarkesh, Jafar Adibi, and Vishal Shah. Knowledge discovery from users web-page navigation. In *Workshop on Research Issues in Data Engineering*, Birmingham, England, 1997.

[100] Edward Shortliffe and Bruce Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23:351–379, 1975.

[101] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Eng.*, 8(6):970–974, 1996.

[102] E. Spertus. Parasite : Mining structural information on the web. *Computer Networks and ISDN Systems: The International Journal of Computer and Telecommunication Networking*, 29:1205–1215, 1997.

[103] Myra Spiliopoulou and Lukas C Faulstich. Wum: A web utilization miner. In *EDBT Workshop WebDB98*, Valencia, Spain, 1998. Springer Verlag.

[104] Myra Spiliopoulou, Carsten Pohle, and Lukas C. Faulstich. Improving the effectiveness of a web site with web usage mining. In *WEBKDD*, San Diego, CA, 1999.

[105] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Fifth Int'l Conference on Extending Database Technology*, Avignon, France, 1996.

[106] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and application of usage pattens from web data. *SIGKDD Explorations*, 1(2), 2000.

[107] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[108] Kun-lung Wu, Philip S Yu, and Allen Ballman. Speedtracer: A web usage mining and analysis tool. *IBM Systems Journal*, 37(1), 1998.

[109] T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linking. In *Fifth International World Wide Web Conference*, Paris, France, 1996.

[110] Yiming Yang and Jan Pedersen. A comparative study of feature selection in text categorization. In *International Conference on Machine Learning*, 1997.

[111] L. A. Zadeh. A theory of approximate reasoning. *Machine Intelligence*, 9:149–194, 1979.

[112] 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*, pages 19–29, Santa Barbara, CA, 1998.

[113] Amir Zarkesh, Jafar Adibi, Cyrus Shahabi, Reza Sadri, and Vishal Shah. Analysis and design of server informative www-sites. In *Sixth International Conference on Information and Knowledge Management*, Las Vegas, Nevada, 1997.