Improved Web Usage Mining Techniques through Data Mining, User Modeling, Clustering, and Behavioral Heuristic Methodologies

by

Kelly Storm

(Under the direction of Eileen Kraemer)

Abstract

As the web has grown and evolved, researchers have developed techniques to extract useful information or knowledge from web resources such as web server access logs, a collection of techniques known as web usage mining. Web usage mining includes several techniques capable of identifying, grouping, and classifying online user behavior. We have created a series of methodologies as well as an automated framework capable of providing contextually meaningful usage information with minimal guidance from evaluators. The techniques discovered through our research have either improved or extended those found in existing data mining, user modeling, and user characterization techniques, and our automated framework assists evaluators in forming connections among the various aspects comprising the users' online experience.

Index words: Web Usage Mining, User Modeling, Web Server Access Log Analysis, Session Analysis, Behavioral Heuristics, Clustering, Visualizations of Usage Behavior

Improved Web Usage Mining Techniques through Data Mining, User Modeling, Clustering, and Behavioral Heuristic Methodologies

by

Kelly Storm

B.S., University of Georgia, 2005

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2011

Improved Web Usage Mining Techniques through Data Mining, User Modeling, Clustering, and Behavioral Heuristic Methodologies

by

Kelly Storm

Approved:

Major Professor:	Eileen Kraemer
Committee:	John Miller
	Jessica Kissinger

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2011

Dedication

To my brother Ryan. Every day you are missed. I will continue trying twice as hard and giving twice as much, for the both of us, until we meet again.

Acknowledgments

I'd like to thank my mother April, for believing in me when no one else would, for always being there for us, for instilling in me an unwavering sense of dignity and loyalty, for teaching me the importance of kindness and the significance of compassion, and for showing me that helping is its own reward. "All that I am or ever hope to be, I owe to my angel Mother."

My father Doug, for teaching me the value of hard work, to persevere, to be vigilant, to strengthen under pressure, to stand for what's right, and to never give up - "Tough times never last, but tough people do."

My brother Jason, for keeping me in check, for constantly reminding me I'm not as think as I smart I am, for helping me laugh when I couldn't remember how, for making me smile when I had forgotten why. "When brothers agree, no fortress is so strong as their common life."

Finally, My advisor Eileen Kraemer, for helping me strengthen my ideas, for showing me the importance of taking my time, and for teaching me to think more like a doctor and less like a salesman. "A teacher is one who makes themselves progressively unnecessary."

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the inception of ARPANET on October 29th, 1969, the Internet has evolved from a communication protocol allowing peers to send and receive packets of information among a few computers around the United States to a massive hierarchical routing system allowing for the support of the World Wide Web, an integral platform that almost all businesses, organizations, educational institutions, and individuals utilize and rely upon for their daily activities. As the web evolved, researchers developed techniques to extract useful information or knowledge from web page contents, a collection of techniques known as *web content mining* [89, 15, 42]. Researchers also developed *web structure mining* techniques to help site administrators understand the relationship between resources through analysis of the link structure of hyperlinks within web documents and web sites, and across multiple web sites [73, 22, 135]. *Web usage mining* involves discovering users' patterns of online behavior, and usage mining techniques were developed to extract useful information from resources containing usage data such as *web server access logs*, aggregate logs of user request information created and maintained by the web server [106, 82, 92]. Web mining can thus be described as a collection of *content*, *structure*, and *usage mining* techniques.

Several content mining techniques were adapted from data representation methodologies traditionally applied to plain text documents. These methodologies were initially used on text documents to extract common words and phrases [89], find correlations between words and phrases [41], and classify words and phrases within the text based on various additional attributes [15]. The adapted techniques were then applied to web site content for pattern matching by word association [42] and prediction modeling [78].

Structure mining techniques were created to help better understand the structure and connectivity of web documents and web sites. In some instances, researchers adapted content

mining techniques, such as the word and phrase association employed by [41], to investigate the significance of similarities in the link structure of several different web sites [135, 53]. In other instances, researchers adapted content mining techniques to determine the relevance of the hyperlinked structure based on the content of the documents connected by those links [62, 18].

Researchers created techniques for usage mining to help better understand the access patterns exhibited by the users as they traversed the hyperlinked structure of a web site for web content. For example, Chen, *et al.* adapted a path traversal pattern mining technique, maximal forward referencing, to better understand when and why users backtracked through a web site [23]. Traditional web mining techniques report metrics such as the number of *page views*, or number of times a web page has been requested, and *sessions*, or instances of users requesting information from a web site within a 30 minute time delimiter for a given period of time, typically by day, week, month, and year. The metrics used by these traditional techniques may not yield contextually meaningful usage information from an application or web site. For example, counting page views yields results that may indicate an increase in traffic, but the interpretation of the increase in page views may vary significantly given the context of the usage information and the domain knowledge of the evaluators analyzing the mined data. Knowledge of the increase in page views can be combined with other information to strengthen or validate broader assumptions, but only if the evaluators are provided with the ability to further isolate the patterns of behavior based on their observations and familiarity with the site.

We believe that usage mining techniques should provide evaluators with the ability to combine various usage data analyses that represent the users' characteristics, requests, tasks, and goals. Evaluators may form irrelevant or incorrect assumptions of online usage behavior when provided only with traditional metrics such as the frequency of visits to a particular page over the duration of a month. However, when provided with the ability to identify

and group users based on their characteristics, such as their browser, operating system, and location, as well as their tasks and goals, such as episodes and exit pages that contain specific types of requests, we believe that evaluators are better able to extract information and form assumptions that are more relevant, more accurate, and more contextually meaningful to their analyses.

Server access logs contain not only records of the activities of users accessing a site with a web browser, but also records of requests made by programs capable of traversing web sites, such as search engine "spiders" and feed readers, software that reads RSS feeds into various devices such as smart phones and desktop applications. Filtering non-human interactions is particularly difficult for server-side logging as no additional information is available beyond what is contained in the access log [1]. Our research shares several problems with those encountered by CAPTCHA recognition approaches, such as how to accurately conclude whether a web user is a human or a bot, where the main issue is determining the level of dissimilarity necessary to be classified as a bot rather than a human [64]. Separating the requests made by human visitors from those made by such programs can be difficult when web bots remove or alter their User-Agent information to disguise themselves as human visitors, when new bot subnets are created outside of the known bot subnets, or if the User-Agent field does not contain enough information to decide if the visitor is using an unknown browser or operating system or is actually a program such as a feed reader. While our approach employs similar techniques to existing methodologies, such as identifying bots based on the request's IP or User-Agent information, we have strengthened the identification process by applying behavioral heuristics that focus on implicit session information, such as the session's distinct requests and staying time.

Information derived from usage mining was applied by Eirinaki, *et al.* to create personalized navigational structures for a web site based on the user's traversal patterns [36]. Similar information derived from usage mining was applied by Martin to analyze the usability of

applications created in the age of *Web 2.0*, an era marked by web technologies relying on remote procedure calls and toolkits allowing web applications to look, feel, and act more like desktop applications [74]. An example of the significance of usage mining can be seen in studies such as Johnson's study of the frequency of requests for influenza related documents as an early predictor of impending outbreaks of flu [48].

Eirinaki's research is an example of the close relation between usage mining and *web personalization*, the dynamic altering of the content and structure of a web site based on a user's traversal patterns and online behavior, and Martin's research is an example of the close relation between usage mining and *web site usability*, the degree to which a web site satisfies the needs of its users. Emphasis on usability allowed developers and administrators to focus their attention on particularly troublesome or confusing components of a web site [57] while web personalization allowed web sites to alter their structure and appearance while the user traversed the site to better serve the user's predicted needs [36]. Pattern extraction and recognition algorithms were integrated into the methodologies to strengthen personalization and usability enhancement techniques [73], and user typing and modeling were integrated to account for the behavioral differences between the growing number of different types of users [79].

Researchers strengthened web site analysis to provide new methodologies offering valuable usage information such as usage clustering techniques, which group users based on their traversal patterns and often provide *profiles*, lists of requests with their corresponding frequency of occurrence, or similar representations of usage for each group [79]. Researchers also strengthened site analysis by creating new tools, and visualizations such *Transaction-Oriented Views*, graphs visualizing user requests, in which requests are grouped as transactions and transactions are annotated and weighted according to their individual importance in relation to the site's functionality [103].

The sessionized access logs used by usage mining techniques proved to be an exceptionally

powerful data source allowing researchers to create models of users based on their requests, visits, and various combinations and derivations thereof. User modeling techniques were adapted to focus on the site's usage, and researchers were able to create new methodologies and metrics such as *Path Analysis Diagrams* (PADs) [108], visualizations of the sequence of requests found within a user's visits, and *Customer Behavior Model Graphs* (CBMGs), visualizations where pages are nodes, edges are transitions, and edges are annotated with the probabilities of transition from one page to the next [75].

When analyzing and performing usage data mining and statistical analysis of user attributes, prior researchers have quantified usage behavior by using page staying times [84], page frequencies [92], visit/revisit ratios [23], "depth" of sessions [82], traversal patterns [80, 106, 36], structural link analysis [53] and session entropy [35].

Web analytics software consists of packages, platforms, or applications that create reports or views of information with the data retrieved through on-site analytics approaches. A few of the key metrics used by analytics packages are hits, a user request for a file, page views, a user request for a page, and visits/sessions, a sequential series of page views requests by a single user. While analytics packages often provide a plethora of information that may help site administrators better understand their usage data, current methodologies and current software frameworks lack the functionality and interactivity necessary to support administrators in answering certain classes of questions [133, 34, 72, 97, 46].

Several studies have been conducted and a number of tools have been created to help developers and evaluators better under the usage patterns their own applications and web sites based on the observable behavior of their users such as browsing patterns [26, 91], determining and highlighting relevant links within the web site's internal structure [8], usage patterns of online services [44], and tools with artificial intelligence that learn with each additional user session [70]. Analyzing usage data has provided researchers with information that helps characterize workloads and group users based on their path traversal patterns

[76, 68, 75, 103]. While these methodologies aid in translating, visualizing and understanding complex usage data, they do not provide evaluators with the tools necessary to form their own distinct groups of users for analysis [65, 57].

Cognitive psychology is a discipline within psychology that investigates the internal mental processes of thought such as visual processing, memory, problem solving, and language. Cognitive psychology researchers who look at the effectiveness of graphical user interfaces (GUIs) typically employ some form of cognitive model or architecture. Their evaluations involve analyzing the actions employed by users attempting to locate specific selectable elements through toolbar, context, and cell phone menu traversal [19, 12, 6], integrating the users' ability to reach their goals into the development cycle of web applications [25], analyzing mental models dynamically created from information generated by user behavior [94], providing cognitive models direct access to simple user interfaces [96], and generating graphs of usage data visualizing the users' actions in an application similar to Customer Behavior Model Graphs but with states representing user actions rather than requests or locations and visual cues such as line thickness representing a higher frequency of occurrence from one action to the next rather than transitional probabilities.

While researchers have created several techniques that successfully extract meaningful usage information such as common user tasks [58], transactions [103], and areas within a site that users may find confusing or tedious [134], the power of these methodologies is limited by their inability to capture important aspects of usage behavior such as the users' ability to reach their goals and an understanding of the process of user type evolution, such as when a novice user becomes an expert user. Furthermore, the techniques employed by these methodologies are rarely translated into software frameworks, and the existing, publicly available frameworks are often difficult to install, lack the ability to parse or to efficiently parse large access logs, generate static information with a low level of granularity, provide limited interactivity with the generated information, and create information-poor

visualizations of usage behavior [32, 97, 72, 34].

Also, while generating usage information based on results created by analyzing users' observable actions has led to viable suggestions and optimizations for a web site or web application, current methodologies remain largely unsupervised and current software frameworks are only able to provide static, aggregate views of usage information.

(i) Usage mining techniques have grown to include faster resource parsers and more accurate machine behavior detection, yet there still exists a need for improvement due to the inability or inefficiency of existing techniques to parse large data resources, to remove usage data created by machines, and to better sessionize web server access logs to more accurately depict the actual movements of the online user. In order to overcome these obstacles we have created preprocessing and sessionizing techniques that employ software improvements and behavioral heuristics allowing evaluators to quickly access and explore more accurate and meaningful representations of the usage information. Furthermore, the majority of sessionization algorithms create aggregate views of usage information that do not allow evaluators to drill down into the information comprising the statistics themselves, thus a need exists for efficient processes to generate hierarchically structured sessionized access logs, in order to provide evaluators with such functionality.

(ii) Evaluations of web sites based on a user's observable actions such as page staying time are used to compare different versions of web sites and often conclude that modifications to a site are effective if a statistically significant, positive difference in staying time or other measure is shown to exist in the modified version [84, 23]. Several algorithms and methodologies have been created that allow evaluators to determine the sections of the site users may consider tedious, group users based on their traversal patterns through the web site, and formulate assumptions to aid developers when they attempt to improve the web site [86, 43, 79, 10, 108]. However, staying time and other such measures are only a few of the users' observable actions. Staying time itself may not accurately convey the general

usage patterns; it does not inform the evaluator as to whether or not the user was able to accomplish their tasks or reach their goals, it merely annotates pages or areas within a web site with the amount of time spent by users. Paired with the content and total size of the page, and an analysis of the occurrence of the page within available user sessions, this provides data from which evaluators are able to form assumptions based on their own knowledge and understanding of the application, but requires a level of supervision rarely offered to evaluators.

(iii) In the field of Cognitive Psychology, user modeling techniques are capable of determining whether the user was able to reach their goal, the frequency of goal acquisition, and the user's proximity to the goal based on the employed productions [19, 94, 12, 6, 30]. Evaluators can better understand user behavior by identifying sequences of requests, or episodes, as productions and specific episodes or requests as goals [19, 45, 83]. By identifying user actions as productions, evaluators are able to observe the productions employed by users to accomplish their tasks or reach their goal state. This also allows evaluators to observe average time spent on specific productions, average error rate, and traversal patters of productions used to reach a goal [96, 12, 6, 45, 30]. While our research only touches on the significance of cognitive architectures and cognitive user modeling, we have combined the notion of productions and goals with the notion of transactions and existing data modeling techniques to create visualizations that better depict online usage behavior.

Computer science and cognitive psychology have their own methodologies and architectures for modeling users. In computer science, *Path Analysis Diagrams (PADs)* visualize the session level traversal path [82], *profiles* detail the pages' frequency of occurrence in accordance to their location within the path traversal [80, 79], Customer Behavior Model Graphs (CBMGs) visualize the transitional probabilities between pages or groups of pages [75], Time Series Analysis Selected Episode Graphs (TSA-SEGs) visualize the evolution of transitional probabilities between pages or groups of pages over time [108], and Transaction

Oriented Views (TOVs) group sequences of pages as transactions and weight transactions according to their usage of server resources before visualizing the transitional probabilities between transactions and states [23]. In cognitive psychology, cognitive architectures are used to create models, some examples of cognitive architectures being ACT-R [19], GOMS [54], and CLARION [50]. The cognitive frameworks create user models that function based on productions, goals, operators, predefined methods, and selectors and determine the cognitive load, rate of task completion, and error rate for a given task [83]. The information used to model users is derived from web server access logs, while most of the information used for creating user models with cognitive architecture requires information provided directly by the user such as questionnaires and think aloud studies.

Our research is motivated by the need to create techniques that address the following areas of weakness in leading usage data mining techniques:

- Accurately distinguishing usage information generated by machines from that generated by humans.
- Identifying types of requests where explicit requests, requests made explicitly by the visitor either through the selection of links on a web page or typing in a URL directly into the browser, are separated from implicit requests, requests that are issued silently by the resource without the visitors knowledge.
- Parsing and sessionizing large server access logs.
- Formatting and storing hierarchically structured usage information into relational database models in order to facilitate analyses comprised of complex combinations and specific isolation of the sessionized access log.
- Grouping visitors and sessions based on similarities found within their usage behavior.
- Visualizing usage information of evaluator specified users or groups of users.
- Automating these processes in the form of a software framework to allow non-technical evaluators to employ the techniques efficiently and effectively.

We have developed techniques employing **data mining techniques**, **user modeling techniques**, **clustering techniques**, and **behavioral heuristics** that help evaluators form connections among various aspects comprising the users' online behavior. We have also created a software framework that provides evaluators with the ability to apply our techniques to their site's web server access log, and contains a suite of modules automating usage mining and user modeling methodologies.

Specifically, our techniques allow site evaluators to:

- Easily and efficiently sessionize web server access logs comprised of their site's user requests.
- Drill down into usage data summaries to extract detailed usage information and dynamically create and refine interactive visualizations of usage behavior based on the users' actions and characteristics.
- Cluster users based on their navigational behavior.
- Identify meaningful session episodes as productions or transactions.
- Isolate sets of pages as specific goal states.
- Iterate through detailed graphs depicting the grouped users' combined requests and productions when attempting to reach goal states.

An example of the visualizations created by our research are the *TSA-SEGs*, Time Series Analysis Selected Episode Graphs, which show the difference in transition probabilities between requests over a selected period of time found within sessions grouped by their usage patterns, and are created by dynamic clustering, data extraction, and model creation and analysis.

To evaluate our methodology we performed several case studies applying our software framework to data sets derived from the access logs of production web sites belonging to organizations such as The University of Georgia Division of External Affairs [85] and The

Center for Tropical and Emerging Global Diseases [109]. To strengthen our software framework, described in section 6.3, we performed several formative evaluations that yielded results we used to directly improve both the usability of the application and the usefulness of the information the framework generates.

The contributions of our research can thus be summarized as follows:

- Bot detection algorithms that more accurately distinguish web bots from human users through the use of localized behavioral heuristics
- Session refinement algorithms that better depict usage patterns by identifying and suppressing implicit requests through the use of localized behavioral heuristics
- Web Server Access Log parsers that more efficiently clean, parse, and store hierarchically structured sessionized access logs
- Semi-Supervised clustering algorithms capable of grouping sessions based on the behavioral patterns exhibited by users within manually selected session representatives
- Interactive selection algorithms that allow evaluators to group users together for analyses based on primary characteristics, such as their location, browser, or operating system, as well as secondary characteristics, such as their usage patterns
- User modeling and visualization techniques that more accurately depict the behavior of a user, or groups of users
- A software framework that automates these processes and allows evaluators to perform complex analyses on their own server access logs

The data preprocessing techniques described in chapter 3 allow us to efficiently parse web server access logs as a hierarchically structured sessionized data set stored in a relational database. The sessionized access log has also been improved by applying localized behavioral heuristics described in section 3.6, which more accurately distinguish between requests made by machines and those made by human users. We more accurately depict

sessions by identifying explicit and implicit requests, described in section 3.7, and suppress implicit requests when creating visualizations or views comprised of usage information. Our evaluations of the accuracy of our behavioral heuristics are described in sections 3.6.1 and 3.7.1 and evaluations of the efficiency of our preprocessing and data integration techniques are described in sections 3.8.1 and 3.9.1.

The unsupervised and semi-supervised classification techniques described in chapter 4 allow evaluators to seed semi-supervised clustering algorithms with sessions indicative of particular user types. Evaluators can then be provided with profiles, or aggregate views of request information containing the frequencies for each request found within the sessions of a given cluster, and can observe visualizations of the similarities for all available sessions.

We then describe usage modeling techniques in chapter 5. These techniques provide evaluators with the ability to guide the process of grouping users based on a combination of several different, manually selected characteristics, then visualize the behavioral patterns of these groups through graphical representations of the subset of usage information.

In chapter 6 we describe our software framework, which automates these processes in a way that allows evaluators to utilize these techniques and contributions in practice.

In chapter 7 we conclude with a brief discussion about our research, plans for future work, and possible case studies we may hold in order to test the next evolution of our modeling techniques.

Chapter 2

Related Work

Online usage data analysis is an area of great interest to both the academic and commercial worlds. While our contributions lie within the areas of web usage mining and web user modeling, our ideas evolved from research in web content and web structure mining, web usability, web personalization, and workload characterization. Several techniques found in the research conducted in these fields have contributed to our work.

2.1 Overview of Web Data Mining

- **Web Content Mining:** Content mining is the process of extracting useful information or knowledge from web content such as the text and images found within web pages.
- **Web Structure Mining:** Structure mining is the process of extracting useful information or knowledge from *hyperlinks*, a structural component that connects the web page to a different location or resource, and document structures such as HTML and XML by analyzing the Document Object Model (DOM).
- **Web Usage Mining:** Usage mining is the process of extracting useful information or knowledge from resources containing usage data such as web server access logs, and is primarily concerned with discovering user access patterns.

2.1.1 Web Content Mining

Web content mining is an area within data mining focused on extracting information from content within a web site, such as digital documents or web pages. Effectiveness methodologies employed in combination with content mining techniques may apply the extracted information to better understand a web site's content.

Content Representation

Researchers created several data representation techniques for web content mining. For example, Ahonen, *et al.* applied descriptive phrase extraction to digital documents by comparing the documents' frequency of terms, *episodes*, phrases containing a sequential list of terms, and frequency of episodes [89]. The research conducted by Ahonen, *et al.* provides an example of researchers applying existing mining techniques to new types of data, with the new type of data in this case being digitized documents.

Researchers created many of the techniques used for content mining by adapting techniques used to analyze textual documents. For example, Billsus, *et al.* created a system that provided news stories to users based on their preferences and interests [15]. The preferences and interests are automatically derived by classifying the text comprising the articles previously read by the user with the *Naive Bayes classifier*, a probabilistic learning algorithm that classifies items based attributes or features [124]. Billsus represented the news stories as Boolean feature vectors, where each feature indicates the presence or absence of significant keywords. The research conducted by Billsus, *et al.* provides an early example of personalization through the use of content mining by creating a primitive data model using the preferences derived from the stories previously read by the user, where preferences consist of a list of dominant attributes.

Feldman, *et al.* extended Billsus' notion of feature vectors containing keywords by applying association rules to the text within digitized documents and extracted key terms instead [42], and created methods for finding relations in ambiguous or weakly connected content by analyzing the relative entropy of categorized words and phrases [41].

Nahm and Mooney created a system for prediction modeling based on rule creation through the statistical analysis of words and phrases extracted from digital documents [78]. In their system, information extraction (IE) becomes an adaptive process where each iteration is strengthened by the rules created from the previous iteration's analysis. The

importance of strengthening IE rules by training the processes with data can be seen in the emergence of search engines such as Google and Yahoo. There quickly existed a need for more efficient and more effective systems that transform unstructured text and semi-structured documents and sites into structured databases, and Nahm and Mooney's work can be seen as an example of the ongoing efforts to improve current methodologies in data mining.

The Beginning of Access Log Analysis

The differences between digital documents such as PDFs and digital resources such as web pages yielded significantly different techniques that borrowed heavily from traditional analyses. Instead of analyzing the content within a digital document, the techniques analyzing digital resources focused on the information created by the resources' secondary characteristics found through access log analysis, such as its temporal characteristics [84] and fluctuating demand [134]. The shift in focus from analyzing content to analyzing *the requests for content* found within the web server access log allowed researchers to approach content mining from an entirely different direction while providing another data source to strengthen and enrich the existing techniques.

The researchers discussed in the previous section created techniques for content mining based on the need to extract useful information or knowledge from web content. As sites evolved, the information generated by content mining techniques did little to answer site administrators' questions concerning the site's structure as well as the behavior of the users online.

Research efforts applying content mining techniques were initially focused on analyzing the information within the web site. The need to analyze a site's structure arose from the inception and use of navigational structures such as menus, breadcrumbs and back links. Content mining is focused solely on the information derived from documents such as web pages and access logs through *localized structural analysis*, analysis of a single digital docu-

ment, but the growing complexity of web sites led to the need for *global structural analysis*, analysis of the web site and its components as a whole. The divide between content mining and structure mining is that of scope; the former analyzing **intra-document structure**, *the structure within a document*, the latter analyzing **inter-document structure**, *the structure linking the documents together*. The noticeable similarities between structural analyses in content and structure mining, combined with the new perspective offered by access log analysis, helped researchers create web structure mining techniques.

Table 2.1: Web Content Mining Approaches

Descriptive phrase extraction based on generalized episodes and episode rules [89]
Text attribute classification using a Naive Bayes classifier [15]
Term Extraction module employing pattern matching using association rules [42]
Predictive, Adaptive Information Extraction [78]
Analyzing the entropy of elements within content hierarchies [41]

2.1.2 Web Structure Mining

Techniques created for web structure mining analyze the page and link structure of web sites, and consist mainly of link pattern extraction, web document structure analysis, and site to site relationships [129]. An example of link pattern extraction can be found in the research conducted by Zhou, *et al.*, which measured the *first, second, and third-order topological properties*, properties based on the connectivity between one, two, and three nodes, of several sites' internal link structures for a comparative evaluation [135]. Zhou, *et al.* generated results indicating a strong correspondence across web sites, which suggests that while varying structures of different web sites appear unique, they tend to share more similarities than differences.

The need to analyze social networks led researchers to adapt content mining techniques,

such as Feldman's research applying association rules to digital documents to extract patterns [42], into structure mining techniques. For example, Hsu's research approached the problem of predicting, classifying and annotating relations between people within a social network based upon the network's link structure [53]. By extracting and analyzing several features that exist within a user profile, such as the person's interest, schools, communities, and friends, Hsu was able to generate collaborative and structural recommendations for the users. Specifically, through normalized mutual interests Hsu was able to generate a fairly precise list of potential friends for each person's profile.

Armstrong, *et al.* extended Feldman's idea of focusing on key terms by applying similar term extraction techniques to a site's hyperlinks and creating learning models through the use of *Hidden Markov Models*, modeling a system as a Markov Process. Armstrong's system, WebWatcher [8], helped evaluators better determine how pages within a web site should be linked together, thus presenting evaluators with a methodology that provides information that can be used to strengthen the internal structure of a web site by linking web content based on term and content similarity. Analyzing WebWatcher's techniques led our research in the direction of understanding the significance of site structure in relation to usage pattern analysis.

However, researchers soon discovered that useful information could be derived through structural analysis alone, "With social network analysis we could discover specific types of pages such as hubs, authorities, etc. based on the incoming and outgoing links" [65].

Due to its exponential growth in the 90's, the need to understand how the World Wide Web was being used quickly became an issue and several algorithms that could accurately model its topology, such as Google's PageRank [18], were proposed and put to use. The notion of usage is often correlated with quality and relevance, but at first the corporations and collaborative efforts responsible for creating these structure mining techniques were primarily concerned with efficiently mining [22], categorizing [18], and creating hierarchies

of authority [63] for the content on the web, content within micro communities [67], and content within web sites [21]. Structure mining attempts to, "determine how the flow of information affects the design of the web sites," while also, "discovering the nature of the hierarchy of hyperlinks in the web sites of a particular domain" [65].

Content and structure mining share several similarities, such as isolating patterns based on frequency of occurrence, but the divide between employing structure mining to better categorize the web and structure mining to better categorize a particular site led to distinctly different areas of interest and created new areas within the field of data mining. The amount of information, breadth of the domain, and particular complications usage mining presents, such as the sharp increase in size of content comprising the web, led to the creation of usability and personalization as separate areas within data mining. Armstrong's research applying the information extracted from web structure mining techniques to dynamically alter a web site in what would later be considered a web personalization technique, an area within the domain of web usage mining, is an excellent example of how researchers adapted existing techniques to overcome these obstacles.

Structure Mining for the Web

Structure mining techniques for the web apply network analysis techniques to the collected global link structure derived by crawling any and all available domains. Research conducted by Kautz, *et al.* uses the site's structure to help identify and locate reliable experts in the domain of expertise defined by the site [62]. For example, the Referral Web, the project created by Kautz's research that generates models of social networks from the site's structure, assists users in locating experts and web sites likely to contain related information.

In [63, 18] we observe that several techniques to model the structure of the web quickly followed as HITS, a link-structure analysis algorithm which ranks pages by "authorities" (pages that have several incoming links and are more likely to contain relevant information)

[63] and PageRank, a link analysis algorithm used by the Google Internet search engine that assigns a numerical weighting to each element of a hyperlinked set of documents with the purpose of determining its relative importance within the set, [18] became an integral part of categorizing pages and sites as well as searching for information within the model's stored connections.

Researchers continued to strengthen structure mining techniques through *link structure content injection*, a technique used to create and dynamically insert content into web sites that lacked a discernible, unifying internal structure [22], and applying filters based on the annotations found within the models to generate lists of links to quality documents related to a user's query [14].

Structure Mining for Web sites

The research of Madria, *et al.* involves discovering structural hierarchies by analyzing requests for information, determining how the results may be affected by the site design, and measuring the connectivity of a web site by analyzing the inter-domain link structure to determine the strength of the pages connectivity [73]. Madria's contributions to research concerned with inter-site structure mining provided a starting point for other researchers to derive and apply information found by analyzing a site's internal link structure.

Web structure mining techniques are currently limited by resources, such as web server access logs, that contain site specific request information. Few organizations, such as Google, Microsoft, and Yahoo, are capable of crawling and categorizing the entire web topology, thus the results generated by researchers applying structure mining techniques are typically limited to the inter-document and intra-document information obtained by crawling a small subset of domains. Moreover, site structures are constantly changing, impacting changes made to sites that heavily depend on inter-site traffic. For example, tailoring a site's structure based on the list of identified in-links may prove beneficial as long as the site's containing

the in-links continue to maintain and provide the links to the public.

A significant amount of overlap exists between structure mining and usage mining. For example, Chen used *maximal forward references*, the requests comprising the list of directed, distinct paths traversed by a user, to describe strongly connected documents or objects and to facilitate interaction with the site's internal link structure [23].

The rapid rise of usage mining research was largely due to the entrance of several large companies financially capable of supporting independent research, and who also stood to benefit directly from the strengthening of these domains. For example, Sergey Brin, co-founder and president of Google, Inc, began publishing research focused on extracting patterns and relationships from web content [17] and scaling techniques for mining structures [105] in 1998. Since then, Google has been directly responsible for hundreds of publications in the fields of Human-Computer Interaction, Usage Data Mining, and Information Extraction [56]. Google's indirect influence can be seen by observing both the high number of citations of Google related research and the quality of work conducted by those citing Google's research.

2.1.3 Web Usage Mining

Mining usage data from web server access logs is a practice performed to better understand online usage behaviors, and a range of methods have previously been applied to mine usage data from these resources [27]. Web usage mining is primarily focused on creating and strengthening techniques that analyze and predict the behavior of users on a web site. Typically, usage mining techniques attempt to discover usage patterns by analyzing the site's server access logs. Results generated by usage mining techniques take various forms, such as Nasraoui's profiles, which itemize high frequency requests as a snapshot of usage behavior [80]. Results derived from usage mining techniques are used for:

- Web Site Modifications (as per Table 2.4 in section 2.2)
- Web Site Personalization (as per Table 2.5 in section 2.3)

Table 2.2: Structure mining Approaches and Evaluation Methodologies

1st, 2nd and 3rd Link Structure Characterization [135]
Attribute measurement and suggestion creation through Structural Link Analysis [53]
Extracting learning models using Hidden Markov models [8]
Mining Micro-Community Link Structures with Strongly Connected Bipartite Subgraphs [67]
Determining the Relevancy of Linked Content with Link Class Annotations [21]
Determining the Relevancy of Referred Information using Social Network Models[62]
In-link Analysis to Discover Authoritative "Hub" Sites [63]
Information Retrieval from Controlled and Uncontrolled Collections [18]

- Physical Systems Improvement (as per Table 2.6 in section 2.4)
- Behavioral Characterization (as per Table 2.7 in section 2.5)

Behavior as Data

Web usage mining techniques analyze user behavior by focusing on the traversal patterns formed by a user during one or several sessions.

The research community's demand to understand and utilize user behavior can best be seen by the popularity of the utilities that were created to automate usage mining techniques. For example, Wu's user-oriented data mining research led to the creation of the SpeedTracer project, which identifies sessions and episodes and utilizes referrer and user agent data to more accurately identify and group users [133]. The research conducted by Srivastava, *et al.* led to the creation of the WebSIFT system, which is designed to parse web server access logs

to identify users, server sessions, page references, traversal patterns, and similar pages [106].

Researchers were also involved in improving the algorithms employed by usage mining techniques to perform tasks such as pattern and session extraction and identification. For example, Pei, *et al.* conducted research to improve the efficiency of access pattern mining through the use of *Web Access Pattern trees*, a data structure that stores compressed access pattern information by collapsing identical transitions into nodes containing the transitions total count [92]. While Wu and Srivastava provided tools to provide the functionality of usage mining techniques to the community, Pei was focused on improving the pattern extraction algorithm when applied to abnormally large data sets.

Johnson contributed to usage mining by tallying and analyzing requests for influenza related articles found within the web sites created and maintained by the Centers for Disease Control to determine if the fluctuation in activity for particular resources could be used to determine an impending outbreak, but found the results on timeliness were inconclusive [48]

Yang, *et al.* extended the *LAPIN algorithm*, LAst Position INduction, to create the *LAPIN_WEB algorithm*, LAst Position INduction for Web Logs, which provided a more efficient sequential mining technique used to extract user access patterns from traversed paths found within the access logs [134]. The LAPIN_WEB algorithm helped predict the demand of resources based on the derived information and thus allowed for developers and administrators to better prepare for fluctuating demand in the future. Yang's research also serves as an early indicator of Workload Characterization.

Li, *et al.* conducted research to improve a website's effectiveness by analyzing the temporal properties associated with each page within a web site [71]. The staying-time of each request can be derived from server access logs, and Li's methodology applied this temporal information to user requests found within sessions to provide additional information about the requests such as the duration of a user's stay on a specific page and how that related to the average staying time of the page in relation to the average staying time of other

pages. Li, *et al.* contributed to usage mining by providing an alternative view to site usage that integrates derived page-temporal information into procedures that assist developers and evaluators in determining the significance of a page, outside of the notions of content and connectivity.

The research conducted by Nicholas, *et al.* aimed to provide a detailed understanding of usage behavior with regard to information searching and retrieval, to characterize and categorize users by their geographical location, content requests, order history, and referring site, and employ "deep log analysis" [82]. Nicholas, *et al.* were the first to utilize "deep log analysis", a framework containing the techniques listed below, which was indicative of the growing demand to provide multiple views of usage behavior by employing a combination of techniques.

- *Site Penetration* is a session characterization technique that categorizes sessions based on its number of requests.
- *Defined By Referral* is a session characterization technique that categorizes sessions based on the session's referral link, or the site the user was previously traversing before selecting the link to the current site.
- *Defined By Subject* is a session characterization technique that categorizes sessions based on the subject of the information requested.
- *Defined By Purchase* is a session characterization technique that categorizes sessions based on the existence or nonexistence of a purchase order.
- *Defined By Subject* is a session characterization technique that categorizes sessions based on the subject of the information requested.

Berendt's research extended the notion of usage mining to include the ability to understand patterns of traversal without having information about the site's structure [13]. Berendt utilized coarsened stratograms to analyze the difference between sessions based on

the syntax of the URLs comprising the session's requests. Berendt was able to establish a hierarchy of several different types of users and to generate accurate predicitons of behavior without having any additional information about the site being traversed. Berendt's research presented a technique that relies solely on the patterns of behavior extracted from the server access log, and characterizes the power and significance of usage mining techniques.

The research conducted by Ritter, *et al.* extended the notion of usage mining to analyze search queries in relation to user responses to evaluate the site's design [95]. Their research indicated the need for usage mining techniques to incorporate user opinions and contains several characteristics also found in insight-based methodologies, such as annoting behavior with information derived from the responses of user interviews.

Most methods, such as Joshi's user profile mining for better personalization techniques, focus on extracting visitor usage information [61]. Nasraoui's research employed profiles to characterize web users. A *profile* is defined as a list of URLs and their corresponding frequency of occurence found within a set of user sessions that have previously been clustered or typed [81]. Profiles can be used to determine the popularity of content by user type. For example, if a cluster of user sessions contains a high frequency of occurence for a particular episode, the evaluator can assume that this area of the site is helpful to the type of user the cluster represents. Profiling allows evaluators to understand the website's areas of interest for specific types of users.

Tangentially, there is also a need to improve the accuracy of bot detection. The difficulties encountered and solutions created when distinguishing between human and bot behavior in our research are shared by Kulopaev and Ogijenko in their attempt to increase the accuracy of popular CAPTCHA approaches when unobtrusively determining legitimate users [64]. When employing CAPTCHA techniques to determine whether a visitor is a human or bot, the visitor is requested to enter a word, or words, that are graphically displayed on the screen. The visitor's input is then compared both directly with the information displayed as

well as with input entered by several other visitors.

Table 2.3: Usage Mining Approaches

Creation of the WebSIFT system [106]
The WAP-mine and WAP-tree Equations[92]
Employing Maximal Forward Referencing to Better Understand Backtracking [23]
Creating Multiple Views of Usage Behavior with Deep Log Analysis [82]
Perfomring Predictive Analyses using Coarsened Stratograms [13]
Determining Site Task Coverage using Query/Interview Cross-section[95]
Determining Demand Prediciton with the LAPIN_WEB Algorithm [134]
Applying Page Temporal Information to Usage Data [71]
Basic User Profile Mining [61]
Characterizing Groups with Extracted User Profiles [81]

2.2 Web Usability

Web usability focuses on the level of difficulty inherent to and the understanding afforded by the users' interactions with the site, and "studies the elegance and clarity with which the interaction with a computer program or a web site is designed" [128]. Usability, as measured by [102], consists of the following factors:

- *Efficiency:* The site's ability to enable users to acquire resources or discover targeted information in an appropriate amount of time.

- *Effectiveness:* The site's ability to enable users to acquire resources or discover targeted information accurately.
- *Productivity:* The amount of useful output obtained by the user through interacting with the site.
- *Satisfaction:* A subjective measurement comprised of the users' response to using the system.
- *Learnability:* The ability of users to fully understand the meaning and logistics of the site's components.
- *Safety:* The ability of the site to limit the risk of harm to users or resources.
- *Trustfulness:* The faithfulness the site offers its users.
- *Accessibility:* The ability of the site to function properly for users with disabilities.
- *Universality:* The ability of the site to accomodate users with different cultural backgrounds.
- *Usefulness:* The ability of the site to enable users to accomplish their goals.

Seffah, *et al.* state that while these factors are not assumed to be independent, factors are often grouped differently when applied to various existing models. For example, web developers charged with creating and supporting the web presence for major banks may be more concerned with the trustfulness, satisfaction, accessibility, and universaility factors, while developers charged with creating and supporting the web presence for major newspapers may be more concerned with efficiency, effectiveness, and satisfaction.

For the purposes of this thesis we focus our attention on usability tools and research concerned with *effectiveness* and *usefulness*, which measure the site's ability to help users accomplish their goals by acquiring resources or discovering targeted information accurately.

Helfrich and Landay introduced a usability evaluation system, Quantitative User Interface Profiling (QUIP), that creates visualizations of aggregated usage data similar to Customer Behavior Model graphs [49]. The main differences between the two graphical representations

are that the visualizations created by QUIP represent actions as nodes, as opposed to states as nodes, making the transitions for these visualizations inherent. In other words, since the nodes within the QUIP visualizations represent actions an edge is automatically created between two nodes when one action follows another. QUIP also provides several visual cues, such as edge width to indicate a high transition rate from the previous action to the current action, and darkened paths to indicate the expected series of sequential actions. Helfrich and Landay's research attempted to create a system that generates objective, quantitative results, as opposed to the subjective opinions received when more traditional techniques are employed, such as questionnaires and user evaluations.

Helfrich also conducted research with Hong, *et al.* at the University of California at Berkeley that led to *WebQuilt*, a web logging and visualization system that allows developers and designers to conduct usability studies through the use of a *proxy server*, a system that acts as an intermediary between the client requesting content and the server delivering the content [52]. The proxy server collects and aggregates the users' requests, then creates graphical representations of the users' transitions through the website. It is important to note that while the visualizations generated by the WebQuilt system presented a sparse amount of information, the process of extracting information and generating representations of user behavior from server access logs had become standard fare.

Hong, *et al.* also indicate the disadvantages of traditional usability testing in [52], stating that usability testing "is very time consuming to run with large numbers of people," and the data generated by the usability studies, "tends to reflect only a few people and is mostly qualitative." Later, when comparing traditional usability techniques to their system, they state that, "server log analysis is one way of quantitatively understanding what large numbers of people are doing on a web site," expressing the ease of acquiring user data for hundreds to thousands of people that if attempted through traditional metrics would exhaust a researcher's resources.

We observe here that while web usability is defined by several factors, usage behavior analysis is the predominant factor in usability testing. Furthermore, Ivory's thorough survey of automated usability evaluation methodologies indicates that log file analysis is a viable methodology for analyzing usage data, and that a third of the existing techniques employed make use of automated data extraction, log analysis, or model generation techniques [57].

Table 2.4: Usability Approaches and Evaluation Methodologies

Overview of Usability Methodologies and Metrics [102]
Usability Testing Through Analysis of Graphical Representations of Usage Behavior [49]
Automated System for Visualizaing Visitors' Online Usage Patterns [52]
Thorough Survey of Existing Automated Usability Evaluation Techniques [57]

2.3 Web Personalization

Web personalization employs processes to create individualized online experiences according to the user's exhibited behavior [126]. When used, web personalization techniques tailor the structure and content of a page or a site based on the available user information, such as browser history, previously captured browsing behavior, or online purchases. Eirinaki [36] states that web structures are, "large and complicated and users often miss the goal of their inquiry or receive ambiguous results when they try to navigate through them," and that, "predicting user needs in order to improve the usability and user retention of a web site can be addressed by personalizing it."

Web personalization consists of three main categories containg their own unique filtering system:

- *rule-based filtering*: filtering employed by creating responses to user interactions using

decision trees, where user behavior serves as a response to questions implicitly asked by the site's structure and content.

- *collaborative filtering*: filtering employed by combining the preferences of users who share online behavioral similarities.
- *content-based filtering*: filtering employed by relying solely on the behavior explicitely defined or given by the user.

Eirinaki later presented a page ranking algorithm, Usage Based PageRank (UPR), comprised of structure and usage mining techniques [37]. The UPR algorithm is seeded with a weighted directed graph created from transitions found within existing user sessions. Transition probabilites are then computed using the connections found within the directed graph using two techniques, one employing first order Markov models to create personalized recommendations, the other employing second order Markov models to create a *hybrid probabilistic predictive model (h-PRM)* to select potential paths from a list of candidates with the highest probability. The strength of this algorithm exists in its ability to weight pages and connections based on the site's actual usage data. Providing the existing transitional probabilities to the algorithm allowed the h-PRM algorithm to predict 80-90% of the paths known to have a high frequency of traversal.

There are several factors to take into consideration when attempting web personalization [126]:

- *Anonymity*: Users may not want to be identified, even if indirectly.
- *Relevance*: Attempts to target users based on small amounts of usage behavior may personalize and present irrelevant information.
- *Security/Credibility*: Companies collecting usage data may house sensitive information about each user. If these companies fail to keep this information secure they risk losing credibility as a company and ultimately the trust and subsequent traffic from

web users.

- *Limited Environment*: There are currently several types of devices used to access content from the web. If the device cannot support the technologies used to implement the system's personalizaion technique, the user's experience cannot be personalized.

Table 2.5: Web Personalization Approaches and Evaluation Methodologies

Web personalization Based on User Profile Comparison [36]
h-PRM, UPR Page Ranking Algorithms [37]

2.4 Workload Characterization

Workload generation is a common form of web user characterization. This type of performance evaluation technique predicts the performance levels that quality service requirements must meet. By modeling the behavior of users using a web application we can study expected traffic characteristics and the impact of the network on the services and applications by the user [68]. The usefulness of workload generation techniques depends on the accuracy of the usage models they create.

Sengupta developed transaction-oriented views to better characterize the various patterns and "intensities" of different transactions by analyzing a site's server access logs[103]. A transaction is defined as, "a series of interactive activities between the user and the website in which the user navigates through one or more pages," and can be considered specialized forms of episodes, where episodes are subsequences of requests found within any given session. Sengupta's research helped identify types of users based on their requests by analyzing the depth of the request with regard to the services provided by the web site. For example, some users issued queries through web applications that requested information from a database,

while other users were simply requesting static content from the web server. Sengupta was able to group these types of users based on their behavior using transaction-oriented views in order to strengthen capacity planning, and provided positive results of server performance, such as providing content more quickly and to more users simultaneously, based on the results derived from his research.

Menascé developed performance modeling techniques used to analyze authentication protocol features and server contention in order to ensure accuracy, reliability, and scalability [75, 77]. In [75], Menascé introduced state transition diagrams called Customer Behavior Model Graphs (CBMGs) to describe the behavior of customers exhibiting similar usage patterns, and provided techniques that allow additional information to be extracted from the model, such as average session length, buy to visit ratio, and average number of items purchased per customer. Menascé was able to derive significant conclusions using genuine and artificially generated server access logs, such as that the probability of a user purchasing an item through an e-commerce webstore decreases as the session length increases.

Menascé's contributions provided a methodology to better characterize e-commerce workloads through performance modeling and capacity planning. The CBMGs employed by Menascé and the Transaction-oriented views employed by Sengupta influenced our research by providing us with models focused on generalized user behavior as opposed to models focused solely on the transitions between simple states such as pages.

Table 2.6: Workload Characterization Approaches and Evaluation Methodologies

Behavior Modeling to Predict Web Traffic [68]
Customer Behavior Model Graphs [75]
Transaction-oriented Views [103]

2.5 User Modeling

Web user modeling characterizes the overall behavior of a web user. Web user models can be informative or predictive, and they can also be used to generate type-characteristic data. Further, user models may attempt to capture the behavior of all users of the site in one model, or may separately model various user groups. These groups may be defined based on simple user attributes, such as the browser or operating system employed, or may be based on behavioral patterns, such session length, size, and count.

Agichtein, *et al.* presented a study modeling user behavior to better predict search behavior [5]. Agichtein, *et al.* noted that, "user behaviors are only probabilistically related to explicit relevance judgements and preferences," and instead of assuming that each inidivudal user is capable of making reliable search decisions they combine the information obtained from several user sessions and model the deviations from expected user behavior to analyze the distribution of "clicks" and queries in order to predict user preferences and user behavior. Their results indicated that analyzing the aggregated user behavior generates results more accurate than the information derived by interpreting basic statistics such as click-through rate (CTR).

Jin, *et al.* introduced task-oriented web user modeling, where the notion of tasks are similar to those found in Sengupta's search involving transaction-oriented views [58]. However, Jin, *et al.* characterized tasks as components of implicit behavior, where behavior is determined based on the domain the user is currently operating within. For example, they state that when a user is traversing an e-commerce web site, user behavior, "may be reflected by sequences of interactions with web applications [used] to search a catalog, to make a purchase, or to complete an online application," while users traversing sites presenting large amounts of information such as the web site's for new organizations, user behavior, "may be reflected in a series of user clicks on a collection of web pages with related content."

The most common navigational "tasks" found within user sessions for a given web site are used to create a Probabilistic Latent Semantic Analysis (PLSA) model that determines the task-level usage patterns, and the results presented in their research indicates an advantage to using PLSA to generate recommendations based on derived usage patterns.

The research conducted by Jin, *et al.* influenced the direction of our research and subsequent programmatic implementation of our theories by providing us with a way to extract common tasks from usage data without having any additional information about the site's domain.

Other models not explicitely defined or created to analyze or predict usage behavior can also be considered user models, such as Menascé's Customer Behavior Model Graphs discussed in the previous section, and Draier's research combining the behavior of users based on their sessions' frequency, length, and depth to more accurately predict future behavior. While Menascé's research focused on workload characterization and Draier's research focused on clustering users based on their behavior, both presented models that provide information administrators can use to directly increase the effectiveness of a website.

Table 2.7: User Modeling Approaches and Evaluation Methodologies

User Modeling based on Aggregate Search Behavior [5]
Latent Semantic Analysis of Extracted User Tasks [58]

2.6 Usage Clustering

Usage clustering algorithms are often used to generate typed groups before creating a user model. Sometimes the clustering algorithms use methods such as latent semantic indexing, which identifies pages with similar content and groups user sessions based on this information [31]. Other times, clustering algorithms are used to comprehend web sites and web applica-

tions by analyzing longest common subsequences and evaluator specified paths to enhance the navigation structure of web sites based on user behavior [86, 100, 79].

Draier[35] introduced new measures for characterizing web users based on their sessions' sequences, or series of user requests within a session with no backward reference. For each session, Draier used the sequence length, connection duration, number of sequences, sequence size, and frequency of sequences relative to all sessions, to help characterize web users. Draier's work evolved how we determine effectiveness by binding the idea of grouping users based upon their access patterns together with determining effectiveness, but focused mainly on session length and duration to characterize and group the user into a cluster.

The latest techniques for grouping based on usage information [35, 71] and characterizing workloads [68] involve clustering types of users based upon their navigational patterns, essentially grouping users based upon their behavior. Several methodologies exist that can be used to group users together based on characteristics such as navigation paths annotated with categorized URLs and session dissimilarity[20, 79, 80].

Cadez[20] introduced model-based clustering, which employs a mixture of first-order Markov models and uses the Expectation-Maximization algorithm to group types of users together. While Cadez's method helps the interactive exploration and visualization of large access logs, it produces many (tens to hundreds of) unclassified clusters, making it more difficult to categorize the clusters by user type

Nasraoui, et al.[79, 80] developed and employed Hierarchical Unsupervised Niche Clustering (H-UNC), which they describe as "a technique created to exploit the symbiosis between clusters in feature space and genetic biological niches in nature"[80]. In this approach, sessions are encoded as N_u-dimensional binary attribute vectors, where u is the total number of valid URLs in the set of sessions under consideration. Studying Nasraoui's unsupervised niche clustering and approach to mining web profiles has allowed us to create behavioral heuristics, discussed in section 3.6 that analyze and more accurately classify the informa-

tion within a session based on how the session's requests compare to other existing visitor requests .

Table 2.8: Usage Clustering Approaches

Orphaned Request Sequence Characterization [35]
Modeling Behavior within ISP Subnets [68]
Model Based Clustering [20]
Hierarchical Unsupervised Niche Clustering [80]
Latent Semantic Indexing [31]

2.7 Cognitive Architectures and User Modeling

In the realm of cognitive psychology, cognitive user models are created based on user-completed questionnaires or by observing user actions and making inferences based on store knowledge [83]. Cognitive user models can be divided into two types, predictive and personalized. Models are typically developed for specific tasks such as menu search, icon identification or automation, and then compared to human data to determine the validity of the model itself [83]. Predictive cognitive user models, such as the task-performance model created by GOMS and the production/goal model created by ACT-R, are used to predict user behavior based on previously observed user behavior [19, 59]. Predictive user models can be used to test interfaces to make predictions about how effectively the users can complete specific tasks within the interface. Research using these predictive cognitive models has produced accurate, attractive results supporting the idea that by implementing cognitive user models researchers can mimic and predict actual user behavior. However, most of the research conducted in the realm of cognitive psychology that utilizes these predictive user models is

limited to menu interaction [19, 12, 96, 30].

Developing a predictive user model involves performing task analysis to define the complexity of the task. The information provided by the task analysis, used within an architecture of cognition, can be used to make predictions about human performance, such as reaction time and error rate. An example of a predictive user model is the GOMS technique, in which a task is broken down into goals, operators, methods, and selection rules [59]. Using a GOMS analysis allows researchers to understand a task at the cognitive level. Given a task, we can understand the subgoals involved with reaching the main goal, the operators used, and the choices made. Other models also exist, such as MHP, which annotates a GOMS analysis by attaching predicted execution times, EPIC, which bases its analysis on productions, SOAR, which can dynamically create new rules and productions as the task unfolds, and ACT-R, which integrates learning and forgetting [83]. A predictive user model attempts to simulate human behavior by taking long term memory, working memory, perception, cognition, and motor processing into consideration.

Personalized user models take the same information used by predictive user models into consideration. However, a personalized user model takes user interactions into consideration to dynamically alter the task itself. Personalized user models can be used to create adaptable and adaptive systems, the difference being that adaptable systems allow the user to configure parameters explicitly, and adaptive systems configure the system automatically based on a user's behavior. A personalized user model can thus be used to determine which elements should be configurable for an adaptable system and which elements should dynamically adapt for an adaptive system.

Before analyzing server access logs became standard procecure, Rauterberg had developed an Automated Mental Modal Evaluation, AMME, in order to analyze data generated by users when performing tasks in an interactive environment [94]. In Rauterberg's study, AMME analyzes the sequences of behavior exhibited by novice and advanced users when

attempting to answer various questions by querying a database. Specifically, Rauterberg's system generates a description of the two types of users in the form of a task dependent model then computes a transition matrix of each user groups' actions. By restricting the users' actions and defining all possible behavioral routes, Rauterberg was able to show that the novice users exhibit a significantly larger amount of complexity in their observable task solving behavior. While Rauterberg's research serves as an excellent example of usage analysis and usability testing, it is also representative of the community's desire to further understand human cognitive processes. By quantifiying user beavior and establishing mental structures representative of user types, Rauterberg's research helped pave the way for researchers creating the cognitive architectures and cognitive user models.

Ritter's work used a predictive model created with ACT-R to have a robot perform actions within a simulation of real world events, the event being driving a car. The model, created from data generated by humans, was able to make realistic predictions on human behavior such as the degree to which fatigue effects driving performance [19].

Several model based evaluations have been performed on cell phone menus and keypads using predictive user models. Researchers were able to accurately predict how long specific actions would take for human users without having to manually test the devices. The results generated by these studies lead to a pragmatic implementation of the predictive user model which can be used to test new phone interfaces without having to first create the device [96, 6, 30].

Table 2.9: Cognitive Architecture and User Modeling Approaches

ACT-R Cognitive Architecture [19]
GOMS Cognitive Architecture [59]
Automated Mental Modal Evaluation [94]
Cellphone Menu Interaction and Prediction [96, 6, 30]

Chapter 3

Usage Data Preprocessing

In order to apply usage mining techniques, evaluators must locate and obtain resources containing web site usage data. These resources can be grouped and defined based on where they are created and controlled. In this section we describe the different types of resources and discuss their advantages and disadvantages.

Usage Data Preprocessing includes the processes used by most methodologies when provided with Server-Level, User-Level or Inherited Resources. Preprocessing involves acquiring the resource from the client or the server, formatting the resource based on the algorithms used to analyze the usage data, and cleaning the resource by removing superfluous or unnecessary information such as internal requests for included resources such as *CSS* , Cascading Style Sheet information, and *JavaScript*, functions and modules that provide client-side interaction with a web site.

3.1 Web Data Resources

3.1.1 Server-Level Resources

Web Server Access Logs, log files that contain the history of requests made by all visitors frequenting a web site, are typically stored on the physical server running the web server responsible for delivering web site content. Access logs store these requests in various formats such as Common Log Format (CLF) and Combined Log Format. Entries found within CLF and Combined Log Format access logs are structured as follows:

- CLF - %h %l %u %t “%r” %>s %b
- Combined Log Format - %h %l %u %t “%r” %>s %b “%{ref}i” “%{ua}i”

and the requests are comprised of the following attributes:

- **IP address (%h)** - the internet address associated with the visitor's machine that sent the request.
- **User ID (%l %u)** - the username or ID associated with the visitor if required to log in to the system, otherwise the User ID does not exist within the logged request.
- **Access Time (%t)** - the date and time the request was received by the server.
- **Request ("%r")** - the resource requested by the visitor, consists of the method (GET which requests resources, POST which sends resources or HEAD which requests only the headers for the resources), *URI*, Uniform Resource Identifier, and protocol for the resource being requested.
- **Status (%>s)** - the status code applied by the server upon handling the request. Codes 200-299 indicate resource request success, 300-399 indicate a redirection, 400-499 indicate errors returning the requested resource, and 500-599 indicate issues with the server itself. Typical codes are 200 (success), 302 (redirect), and 404 (resource not found on server).
- **Bytes (%b)** - the size of the resource being requested in bytes.
- **Referrer ("%{ref}i")** - the URI source of the request.
- **User Agent ("%{ua}i")** - the browser and operating system being used to issue the request.

and example requests appear as follows:

- CLF - 127.0.0.1 - storm [10/Oct/2000:13:55:36 -0700] "GET /index.html HTTP/1.0" 200 2326

- Combined Log Format - 127.0.0.1 - storm [10/Oct/2000:13:55:36 -0700] “GET /index.html HTTP/1.0” 200 2326 “http://www.example.com/start.html” “Mozilla/4.08 [en] (Win98; I ;Nav)”

Due to client-side caching, the requests found within the server access log may not reflect the entirety of a visitor’s actions within a given session. For example, when a visitor clicks the back button within a browser, the browser will return the cached resource to the visitor without requesting the resource from the server. Assuming that there exist three pages within a web site and the visitor requests the first page, page A, from the server, the server will return page A to the visitor and log the request in the server access log. The visitor then requests page B, is provided with page B, and the server logs the request for page B within the access log. The visitor then clicks on the back button within the browser and is given page A, but not by the server. The visitor then requests page B again and the request is provided to the visitor, and also logged in the access log. Technically the visitor requested pages A, B, A, and B, but the server only indicates the requests for pages A, B, and B respectively. Most web sites currently in existence dynamically generate content before delivering the resource based on the visitor’s request, and dynamically generated information cannot be entirely cached by browsers, so this is becoming less of a problem but still threatens the validity of visitor sessions whenever delivering resources that contain static content.

Access logs contain a wealth of information, are usually readily available, and often span back to when the site was originally launched. The web server reliably records all transactions and access logs have a standardized format as per W3C specifications [3]. However, log files for enterprise strength web sites and web applications often contain tens of millions of user requests and can expand to several gigabytes in size[33].

3.1.2 User-Level Resources

Client-side access logs are created by storing requests sent by *RPCs*, remote procedure calls capable of sending visitor information such as the browser type and request to the server asynchronously and silently. Traditionally this method required the visitor's consent, but the technology necessary for RPCs is included, and automatically activated, for all major browsers currently in existence. Client-side access logs are capable of providing richer resources for analysis, but requires that the evaluators include the JavaScript necessary to issue the RPCs on all available pages within a web site. This task can sometimes be as easy as altering the header within a template, but can often be daunting or impossible in other cases. For larger, more complicated web sites, or web sites whose functionality spans multiple domains, the server-level resources provide more viable information than user-level resources.

3.1.3 Inherited Resources

Web proxies are either physical machines or instances of software that serve as intermediaries between the user and the web server delivering the actual resource based upon the user's request. Proxies are often implemented to reduce the strain on the server housing the resource and to minimize the observed latency between requesting and receiving a resource from a web server. Proxy servers do not provide holistic access logs, but when merged with server access logs can often provide the information that is missing from server access logs due to client-side caching.

3.2 Acquiring and Cleaning Usage Data Resources

Web server access logs are found on the physical servers running the web servers responsible for delivering resources to the visitors of a web site. Client-side access logs are often stored on

the visitors' computers and are either periodically synced with the physical servers supporting the web servers or provided to evaluators by the visitors.

Resources are cleaned by analyzing each line and removing any request for information that does not contribute to the understanding of the site's hierarchical structure, content, or usage, such as images, CSS files, and JavaScript files. Requests made by non-human visitors such as web bots, RSS feed readers, and crawlers are also removed. While analyzing the requests created by programs such as RSS feed readers may be used to understand usage behavior, it does not directly reflect the traversal patterns of visitors on a web site. The cleaned resource is substantially smaller and thus helps reduce the amount of time additional analyses take.

3.3 Visitor Identification

A visitor is a person or machine that visits a web site by requesting resources, such as web pages or documents, from the web server hosting the web site [4]. The two main methods for identifying a visitor, identification via login and identification via cookie, depend on the application domain and are not readily available in the access logs [3]. Typically, however, a visitor is identified by IP address or an (IP address, user agent) pair [3].

3.4 Session Identification

A visitor session, or visit, is defined by W3C as a collection of requests made by a particular visitor to a single web server [2] such that each request is within a specified time interval, usually 30 minutes, of the prior request. Visitor sessions can provide valuable insight into the dynamic behavior of web applications by providing path traversal and staying-time information for every session of every visitor [65].

3.5 Web bot Identification

Server access logs may contain records of both human visitor requests and requests made by programs capable of traversing web sites, such as search engine "spiders" and feed readers, software that reads RSS feeds into various devices such as smart phones and desktop applications. By filtering out non-human visits and requests we decrease the noise in any given data set, thus ensuring that the usage behavior observed by the analyst more accurately reflects that of human visitors. Performing such filtering of non-human interactions is particularly difficult for server-side logging, which is our focus, as no additional information is available beyond what is contained in the access log [1]. Our research shares several problems with those encountered by CAPTCHA recognition approaches, such as how to accurately conclude whether a web user is a human or a bot, where the main issue is determining the level of dissimilarity necessary to be classified as a bot rather than a human [64]. Separating the requests made by human visitors from those made by such programs can be difficult when web bots remove or alter their user-agent information to disguise themselves as human visitors, when new bot subnets are created outside of the known bot subnets, or if the user-agent field does not contain enough information to decide if the visitor is using an unknown browser or operating system or is actually a program such as a feed reader. Traditional web bot identification techniques will compare the visitor request's user-agent to a list of known plain-text web bot signatures and identify the request as that of one being made by a web bot [93]. If a match is found, the request is removed from the access log and the visitor is flagged as a potential web bot and subjected to higher levels of scrutiny throughout the remainder of the log cleaning process. For example, the IP address associated with a visitor who has been flagged is compared against several known *bot net sub ranges*, or IP ranges known to be used exclusively by web bots. If the IP address is within a known bot net sub range, the visitor is considered a bot and subsequent requests made by the visitor are

systematically removed from the access log. The process works similarly in the opposite direction, in which a visitor may appear to be human according to their user-agent but their IP is known to be in a bot subnet.

3.6 Localized Behavioral Heuristics

Heuristics are experience-based techniques for problem solving, learning, and discovery [132]. Localized behavioral heuristics extend these techniques to generate information based on the behavior exhibited by the visitors within a specific server access log. The information generated by the localized behavioral heuristics is applicable only to the access log used to initially train the heuristic. While our approach employs similar techniques to existing methodologies, such as identifying bots based on the visitor request's IP or user-agent information, we have strengthened the identification process by applying localized behavioral heuristics that focus on a session's information, such as the session's distinct requests and staying time. If a bot manages to exist outside the known bot subnets, does not originate from a suspicious IP address, and does not contain any information in its user-agent that is indicative of a web bot, it is still subject to evaluation using localized behavioral heuristics, which we term the **Request Behavioral Heuristic (RBH)** and the **Staying Time Behavioral Heuristic (STBH)**. The research conducted by Tauscher and Greenberg and the software created by Sanfilippo, *Visitors*, allowed us to extend these behavioral heuristics so that we can better distinguish the requests that were made explicitly by the users, explicit requests, from the requests that were made implicitly by the page's silent requests, implicit requests [69, 98]. Specifically Tauscher and Greenberg were concerned with how the site's history mechanism influenced the behavior of returning visitors, thus forcing them to isolate and observe only those pages returned to the users' explicit requests in order to measure longest repeated sequences by applying Crow's pattern detection module (PDM) algorithm [69, 29]. The RBH compares the request patterns of each session to the aggregated request patterns exhibited

by all available sessions extracted from the server access log. Specifically, RBH evaluates whether the total number of requests and total number of distinct requests for a given session is within two standard deviations of the average number of requests and total number of distinct requests found within all other sessions. RBH helps identify sessions created by visitors that were not identified within any known bot subnet, did not contain information within any of the session's requests' user-agents that were identified as having been created by a bot, but were nonetheless created by a bot. Thus, the visitor will be identified as suspicious, or likely to be a bot, even though they contain no identifiable bot information. However, instead of systematically removing the visitor's requests, the visitor's sessions are placed outside of the data set until the evaluator decides to manually reinclude the session. For example, if a web bot masks or alters its user-agent information and uses an IP that is outside of all known bot subnets, it will still be identified as a bot if it requests a statistically larger number of identical pages. This increases the likelihood that the visitors and visitor requests found in the cleaned access log represent legitimate human users. STBH is a behavioral heuristic that compares the staying time patterns of each session to the aggregated staying times derived from all available sessions. STBH evaluates if the staying time for each session is within two standard deviations of the average staying time for all sessions extracted from the server access log. STBH helps identify sessions that perform a large number of requests in a short period of time. For example, if a visitor requests hundreds of pages in a matter of seconds, it will be identified as suspicious by the STBH. If a session is more than two standard deviations away from both averages, all sessions created by that visitor are flagged as suspicious.

The procedure for basic machine behavior detection is as follows:

1. The visitor's IP is checked to determine if it is within a list of known bot IPs as well as if it is within a list of known bot IP ranges.

2. If the visitor passes the IP and IP bot range check, the visitor's user-agent is scanned for identifying keywords known to belong to bots.

If either of these tests fail, the unique visitor ID is added to a list of known bots and the bot, as well as the bot's previous, current, and future requests, are ignored. If the visitor successfully passes both tests, we will then apply the Request Behavioral Heuristic and Staying Time Behavioral Heuristic detailed below.

3.6.1 Staying-Time and Request Behavioral Heuristic (STBH, RBH)

A session's requests are the requests made by the user throughout the duration of the session. The Request Behavioral Heuristic (RBH) isolates session outliers by comparing the session's requests to the requests made in all other available sessions. Unlike a session's staying time, a session's requests do not have comparable values, and comparing sessions based on only the number of requests or the number of distinct requests does not generate any meaningful results. Instead, we calculate a session's request uniqueness, shown in equation 3.1, by assigning a probability to each of the session's individual requests, where N is the number of requests within the session and R is the request being examined.

$$|P_r| = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} +1 & R_i \notin R_0 ... R_{i-1} \\ +0 & R_i \in R_0 ... R_{i-1} \end{cases} \tag{3.1}$$

Next, we isolate the session's *unique* requests, U, and sum the probabilities of each unique request then divide the summation by the total number of requests within the session, shown in equation 3.2.

$$\alpha_{req} = \frac{1}{N} \sum_{i=1}^{U} P_r[U_i] \tag{3.2}$$

Once α_{req} has been computed for each session, we determine the average α_{req} for all sessions, μ_{req}, in equation 3.3, where M denotes the total number of sessions.

$$\mu_{req} = \frac{1}{M} \sum_{i=1}^{M} \alpha_{req} \tag{3.3}$$

Next, we compute the standard deviation, σ_{req}, in equation 3.4.

$$\sigma_{req} = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (\alpha_{req_i} - \mu_{req})^2} \tag{3.4}$$

Once the standard deviation for equation 3.4 is computed, we can determine if the session is an outlier based on whether its requests are more than two deviations away from the averaged requests.

$$S_{outlier} = \begin{cases} true & |\mu_{req} - \alpha_{req}| > 2 * \sigma_{req} \\ false & |\mu_{req} - \alpha_{req}| \leq 2 * \sigma_{req} \end{cases} \tag{3.5}$$

A session's staying time is the summation of the session's requests' staying times and can only be computed for non-bounce sessions, or sessions that are comprised of two or more requests, whose requests are separated by at least one second [131]. The Staying Time Behavioral Heuristic (STBH) isolates session outliers by comparing the session's staying time to the average of all available staying times and computing the session's staying time variance, standard deviation, and number of deviations away from the computed average. The algorithm for applying the STBH is detailed below.

1. Isolate all non-bounce sessions containing requests separated by at least one second.
2. Compute the summation of each session's requests as the session's total staying time.
3. Compute the average staying time for all available sessions.
4. Compute the variance given all sessions staying time and the previously computed average staying time.
5. Compute the standard deviation by taking the square root of the variance.
6. Isolate the sessions whose average staying time is two standard deviations either above

or below the average staying time.

To better explain the STBH we map each step in the algorithm to its own equation. Assuming we have isolated all non-bounce sessions containing requests separated by at least one second, we continue forward with equation 3.6 by computing the summation of each session's requests as the session's total staying time, where N represents the number of requests within a session and X represents the staying time for said request, then storing the session's staying time, or ϕ_{st}, as meta-data on top of the session itself.

$$\phi_{st} = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{3.6}$$

Next, we compute the average staying time for all sessions, shown in 3.7 by summing every session's staying time and dividing it by the total number of sessions. Again, we are using only the non-bounce sessions whose requests are separated by at least one second, represented by M.

$$\mu_{st} = \frac{1}{M} \sum_{i=1}^{M} \phi_{st_i} \tag{3.7}$$

Once the global average has been computed, we compute the standard deviation of the average staying times in equation 3.8.

$$\sigma_{st} = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (\phi_{st_i} - \mu_{st})^2} \tag{3.8}$$

Now that the standard deviation has been computed, equation 3.9 shows us how we can isolate the outliers based on their distance from the average staying time by focusing our attention on those sessions whose staying time is more than two standard deviations away.

$$S_{outlier} = \begin{cases} true & |\mu_{st} - \phi_{st}| > 2 * \sigma_{st} \\ false & |\mu_{st} - \phi_{st}| \leq 2 * \sigma_{st} \end{cases} \tag{3.9}$$

Experiment: Determining Effectiveness of STBH and RBH

In order to determine the accuracy of the STBH and RBH behavioral heuristics, we applied them to a server access log generated by PlasmoDB.org both before and after it had been cleaned. Before doing so, we manually annotated the access log to distinguish web bot visitors from human visitors and requests made and sessions created by web bots from those created by human visitors.

We computed the sensitivity and specificity of our behavioral heuristics with these two sets of sessions to determine the accuracy of our approaches with regard to their ability to correctly identify web bots and web bot behavior as well as correctly distinguish between web bots and human users. The cleaned log had 2788 distinct visitors and 10709 distinct sessions. The behavioral heuristics identified 12 of these visitors as bots and 70 of the sessions as being generated by bots. Two of these visitors were manually determined to be humans and not bots, yielding a 16.6% false positive rate for bot detection. Five of these sessions were determined to be created by humans and not bots, yielding a 7.14% false positive rate for bot session detection. The log that had not been cleaned had 4593 distinct visitors and 24507 distinct sessions. Of the visitors, 1599 were identified as bots and 13243 of the sessions as being generated by bots. In other words, the uncleaned access log held roughly 15,000 additional requests created by 1500 non-human visitors. If these non-human requests are not removed from the access logs, they may alter the representations and models of usage behavior. In doing so, evaluators may form conclusions from these representations that may not necessarily reflect the needs and expectations of human users. The STBH and RBH behavioral heuristics missed 204 known bots and falsely identified 8 human visitors as bots yielding an 88.7% success rate for bot detection and a 0.5% false positive rate. The heuristics also missed 490 sessions known to be created by bots and falsely identified 66 sessions as being generated by bots yielding a 95.9% capture success rate and a 0.49% false positive rate. The sensitivity and specificity measures for human and bot visitors and sessions can

be found in Table 3.1.

Log Type	Clean	Unclean
Visitors	2788	4593
Sessions	10709	24507
Visitor FP%	16.6%	0.5%
Visitor Sensitivity	83.3%	89.8%
Visitor Specificity	99.9%	99.7%
Session Sensitivity	93.3%	96.5%
Session Specificity	99.9%	99.4%

Table 3.1: Bot vs Humans, Sensitivity and Specificity

The results indicate that our behavioral heuristics are capable of detecting and removing requests made by non-human visitors for both cleaned and uncleaned access logs. Given this information, we can further minimize the technical expertise required by evaluators when parsing access logs by gracefully accepting both cleaned and uncleaned access logs.

3.7 Explicit and Implicit Request Identification

Sessions consist of explicit requests, requests made explicitly by the visitor either through the selection of links on a web page or typing a URL directly into the browser, and implicit requests, requests that are issued silently by the resource without the visitors knowledge. Furthermore, the web server cannot distinguish between implicit and explicit requests, logging every requests reveived from the visitors regardless of whether or not they knowingly requested the resource. The majority of implicit requests are comprised of image, CSS and JavaScript includes, and are removed when the server access log is cleaned. However, due to the increase in popularity of incorporating *asynchronous JavaScript and XML (Ajax)*, web development methods used on the client-side to create interactive web applications, requests that were previously explicit in nature, such as requests for a specific web page, can now

be issued implicitly [130]. When extracting sessions from the access log, all requests issued by the visitor within the allowable time threshold of 30 minutes are grouped as part of the visitor's session requests, but the session's implicit requests are not indicative of the visitor's usage behavior and should be removed. However, the majority of the time explicit and implicit requests are logged identically, making it impossible to remove or exclude the request based on the request's primary characteristics.

For any given session with more than one request there are often multiple implicit requests sent from the pages returned by a visitor's explicit requests. For example, when visiting http://google.com an index.html page is sent back to the visitor's browser, but an implicit request for blank.html and *get*, a cgi-bin script, are also received by the web server from the visitor. Although the visitor did not explicitly request these resources, they are still stored in the server access log as distinct requests. When viewing the visitor's requests it appears as if there are three distinct requests made in succession, yet in reality the visitor requested a single resource, index.html.

3.7.1 Lily Pad Behavioral Heuristic

By analyzing the session request's referrer, or the request's previous request, it seems logical to assume that we can simply map the requests to their referrers to determine which requests were explicitly sent by the visitor and which requests were implicitly sent from resources returned by explicit requests. However, Figure 3.1 shows how Apache processes the metadata and content for each request *before* logging the request, which means that annotating requests as explicit based solely on their referrer request may produce invalid representations of a visitor's explicit series of requests.

This can be particularly problematic when attempting to understand broader usage patterns of large numbers of visitors, so we need a behavioral heuristic capable of determining explicit and implicit requests. The Lily Pad Behavioral Heuristic (LPBH) allows us to isolate

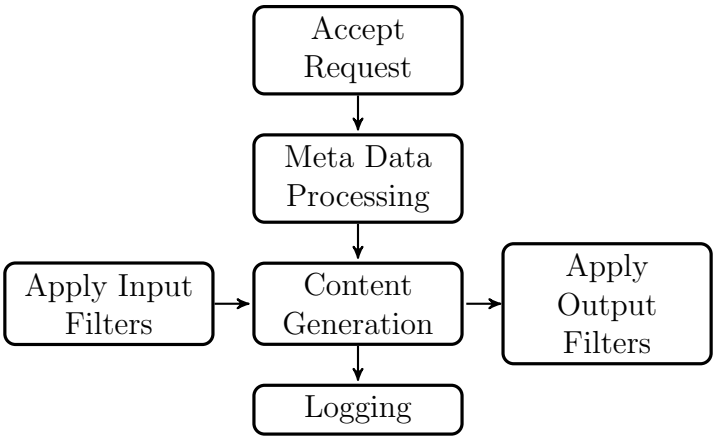

Figure 3.1: Apache Request Handling Process

these explicit requests by analyzing the fluctuating variance of the staying times for these requests. Similar to STBH and RBH, LPBH computes the average staying time and isolates requests based on the deviation from the average. However, instead of isolating requests that are more than two deviations away from the average, we select the request that is the furthest away from its average in order to isolate requests with a higher staying time variance. Implicit requests will typically be stored in a similar order and take roughly the same amount of time regardless of when or how the resource was requested. That said, we believe that these implicit requests will have a significantly lower staying time variance than explicit requests, so when presented with a series of requests originating from the previous request as per their referrer information, the explicit request is most likely the one that is furthest away from its own average. As an example, let us assume that a visitor requested information from a web site and the extracted session consisted of five distinct requests shown in figure 3.2. Analyzing the visitor's requests further we determine that only two of the

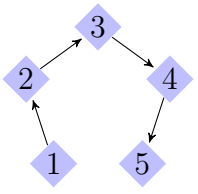

Figure 3.2: Requests By Order

visitor's requests were explicit, while the remaining three were implicit requests sent silently without any action from the visitor. By analyzing each requests' referrer information we are able to determine that request 2, 3, 4, and 5 in Figure 3.3 have request 1 as their referrer, and request 2 appears in the access log before 3, 4 or 5. We assume because 2 preceded the remaining requests that it was the last explicit visitor request while the remaining three were the implicit requests. To overcome this obstacle, we apply the Lily Pad Behavioral Heuristic

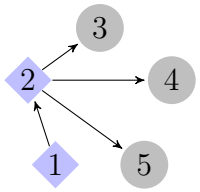

Figure 3.3: Request By Order, Referrer

(LPBH), which extends the STBH in a localized manner. Within each session there are a series of sequential requests. When given the request's referrer information we can classify the subsequent request within the session by comparing that request's referrer information to the current request's information. If the subsequent request's referrer information matches the current request, it is classified as an explicit request. However, if there is more than one request following the current request that all contain identical referrer information, it is unclear as to which request is the true explicit request. For example, in Figure 3.3 we see that pages 2, 3, 4 and 5 are in question, so we will compute the average staying time for all requests for these pages found within all sessions. We know that we cannot rely on the web server to store the requests according to when they were received and we know that we cannot rely upon the primacy or recency of requests sharing similar referrer information, so we are left to determine the explicit requests another way. Similar to the STBH, we compare the staying time of these locally grouped requests to their average staying time within all available sessions containing the request itself. To compute LPBH we first determine the average staying for each request in question. Given M as the number of sessions, N as the

number of requests, R as the request, and R_s as the request's staying time within the session, the request's average staying time is computed as follows:

$$\mu_S = \frac{1}{|R|} \sum_{i=1}^{M} \sum_{e=1}^{N} \begin{cases} +R_s & M_i[N_e] = R \\ +0 & M_i[N_e] \neq R \end{cases} \tag{3.10}$$

We then compute the requests' path's average staying time with equationImplicitPathSummation. The difference here being that the request could be a long or obscure URL such as /index/main/getdata.php?op=1 and may have a low frequency of occurrence within the other sessions. The path in this instance would be requests containing /index/main/getdata.php. Given R_p as the requests' paths, the requests' path's average staying time is computed as follows:

$$\mu_R = \frac{1}{|R_p|} \sum_{i=1}^{M} \sum_{e=1}^{N} \begin{cases} +R_s & M_i[N_e] \supset R_p \\ +0 & M_i[N_e] \not\supset R_p \end{cases} \tag{3.11}$$

We then combine the standard deviations for both requests and paths in equation 3.12 and 3.14

$$\sigma_S = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (N_i - \mu_S)^2} \tag{3.12}$$

$$\sigma_R = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (N_i - \mu_R)^2} \tag{3.13}$$

Once the deviations have been computed, we identify the explicit request, $R_{explicit}$, where L denotes the subset of requests within N being examined, that is *furthest* away from its combined average.

$$R_{explicit} = \arg\max_x f(x) := x | \forall L : |(\sigma_{S_L} + \sigma_{R_L}) - \mu_{R_L}| \tag{3.14}$$

By applying LPBH we are able to determine that 4 is the most likely subsequent explicit

request. Figure 3.4 shows us that 4 was the other explicit user request even though it was stored in the access log after 2 and 3.

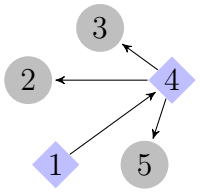

Figure 3.4: Request By LPBH

Experiment: Determining Effectiveness of LPBH

In order to determine the effectiveness of LPBH we randomly selected 100 sessions from an access log created by PlasmoDB.org that were manually determined to contain both explicit and implicit requests and contained at least 10 requests and no more than 100 requests. There were 4402 total requests for all 100 sessions analyzed. We then applied the LPBH to these 100 sessions and then manually verified to determine what percentage of explicit and implicit requests were correctly or incorrectly identified. For example, if a session contained eight requests comprised of six explicit and two implicit requests, any explicit or implicit request that was properly identified counted towards the LPBH and any requests incorrectly identified as the other counted against the LPBH. Of the 4402 requests, 655 were known to be explicit requests and 3747 were known to be implicit requests. LPBH identified 631 requests as explicit requests where 544 were known to be explicit requests and 87 were known to be implicit requests LPBH also identified 3771 requests as implicit requests where 287 were known to be explicit requests and 3484 were known to be implicit requests. The analysis of LPBH resulted in a sensitivity measure of 88.3% and a specificity measure of 92.9%.

3.8 Sessionizing Access Logs

Sessionization is the process of grouping the requests parsed from the server access log into *visits*, also known as *sessions*, comprised of series of requests made by distinct visitors [4]. If a visitor requests information from a web site and its previous request occurred less than 30 minutes before, the requests are determined to be within the same session [92, 133]. Otherwise, the previous request is determined to be in its own session while the new request is considered the starting point for the visitor's latest session.

Before sessions are extracted from the server access log, the access logs are cleaned in order to remove requests that are not interesting when analyzing visitor behavior, such as embedded image files or CSS and JavaScript includes. HTTP errors and records created by web crawlers are also removed because they do not reflect the behavior of actual visitors [134] and are thus not useful for modeling visitor usage behavior. Due to the cumbersome, hefty nature of the access log, there has also been a great deal of research focused on increasing the efficiency of preprocessing and sessionizing techniques. Srivastava and Cooley's research has been largely influential to our approaches as we have adapted our sessionization algorithm from their techniques. Specifically, analyzing WAP-trees allowed us to create stronger sessionizing algorithms by adapting our approach to store and retrieve the indices of the page as a label for comparison instead of the entire page or request itself, and observing Srivastava and Cooley's approach allowed us to expand our approach to include preprocessing and pattern analysis instead of simply sessionizing the access log [107, 27]. In summary, our sessionization techniques, Very Agile Sessionization Heuristics (VASH), builds on existing approaches by furthering the efficiency of preprocessing usage information and the sessionization of server access logs, and it differs by incorporating behavioral heuristics that more accurately classify visitor types and create clearer representations of sessions by distinguishing between explicit and implicit visitor requests.

3.8.1 Very Agile Sessionization Heuristics (VASH)

By optimizing the sessionization algorithms, employing the behavioral heuristics described in previous sections, decreasing the memory footprint, and integrating several code optimizations and third party packages into the sessionization technique's programmatic implementation, we have an approach that more efficiently sessionizes web server access logs.

Very Agile Sessionization Heuristics (VASH) is the practical application comprised of the contributions detailed in previous sections as well as the contributions described here.

To begin, the sessionization algorithm must maintain certain information to ensure that the sessions extracted from a server access log accurately describe the series of requests made by each user, such as each visitor's identifying information and most recent request. With significantly large access logs the amount of information in memory can quickly become unmanageable [33]. VASH is implemented in Java 1.6 and uses MySQL 5.1+ equipped with the transactional storage engine InnoDB. Our first optimization involved incorporating third party packages into the sessionization algorithm in order to inherit the benefits of utilizing classes that have been thoroughly tested and optimized by other parties. The Guava libraries, Google's core libraries for Java 1.5+, and the Apache Commons Collections framework have been the most influential open source packages for our algorithms. The decreased latency due to the use of Google's CharEscapers and GenericUrl classes for DNS lookups and URL manipulation, and the LRUMap class found in the Apache Commons Collections Framework allowed us to create an implementation of our algorithm that competes with leading implementations, and decreases the latency of our parsing algorithm by 31% and the latency of our ordering algorithm by 25% [47, 7].

Our next optimization consisted of optimized methods and algorithms for frequently used simple tasks, such as removing white space from a String, splitting a request based on a common delimiter, matching patterns without using a regular expression pattern matcher, and storing several million representations of Strings, without using the Java **String** class.

We significantly reduced the time and resources required to sessionize the access logs by writing our own classes for basic data manipulation such as pattern matching, indexing, and information storage and retrieval. To decrease our memory footprint we indexed all textual representations of data, such as the page and user requested and the request's user-agent information, using java.lang.String.hashCode in tandem with java.util.HashMap before storing the text into a collection of comma-separated value (CSV) files annotated with unique indexes that would later represent the tuple's primary key when inserted into the database. Due to the heavy memory usage of java.lang.String and the latency of java.util.regex.Pattern and java.util.regex.Matcher, text manipulation, such as concatenating String objects or searching for patterns within Strings, was handled entirely by customized methods that relied entirely on sequential pattern matching of character arrays [111]. Also, the java.lang.StringBuffer class was used when the algorithm required the use of String concatenation. Through the use of Java 6's HashMap and String class, specifically String's hashCode() method, we increased the efficiency and decreased the space and memory required to manage the primary and foreign key constraints before writing the information to comma separated value (CSV) files and inserting the information into the database. We also replaced any instance of basic String manipulation or Regular Expression matching through the Match and Pattern classes with customized methods built to logically traverse character arrays and build output with StringBuffer objects. The algorithm relies largely on heavily indexed HashMaps containing the primary keys for elements that can be directly identified by the integer value returned from their hash ID returned through the hashCode() method.

The optimizations integrated into the algorithms are integral to sessionizing large access logs because there may be hundreds of thousands of requests, tens of thousands of sessions, and thousands of visitors in memory at any given point in time. By playing to the strengths and avoiding the weaknesses of Java's features, we significantly reduced the memory usage and time required to parse large access logs. The sessionization procedure is as follows:

1. Employ modified parsing method to extract request information from each line within the access log

2. Create index in memory and write full description as well as its respective index to file if not found of the following:

   (a) a unique visitor ID given the IP and user-agent information if one does not already exist, otherwise use unique visitor ID found in memory

   (b) a new session for the visitor ID if one does not already exist, otherwise use the session for the visitor ID found in memory

   (c) a new request ID, referrer ID, and user-agent ID if they do not already exist for the current page request, otherwise use the existing IDs found in memory

3. Compute the time difference of the session's current and previous requests

4. If the difference is greater than 30 minutes, or 3600 seconds, write the session's information to file and clear the session from memory and detach the unique visitor ID from the session

5. Otherwise, update session by appending these indexes to indicate the latest request belonging to the session, belonging to the visitor

Experiment: Determining Effectiveness of VASH

We conducted a performance evaluation on our parsing algorithm both with and without the third party software. The performance evaluations here, as well as those found in section 3.9.1, were performed on a machine running 64-bit Windows 7 Ultimate with two Core 2 Duo P8400 2.26GHz CPUs, 4GB DDR2 400MHz (P2-6400) of RAM, and a 5400RPM Hitachi Travelstar 5K320 hard drive. The eight log files used throughout the performance evaluations contained data generated by PlasmoDB, http://plasmodb.org, and ranged in size from 10MB

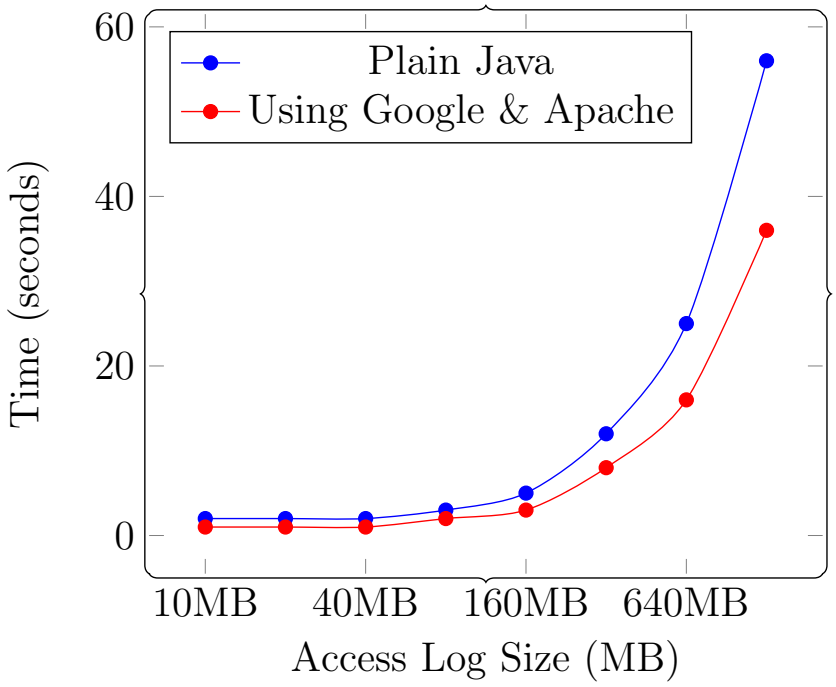

Figure 3.5: Parsing Time (seconds), Basic Statistics

to 1024MB, or 1.028GB, representing requests made by users between February 14th, 2010 and October 8th, 2010. Figure 3.5 below shows the time taken by the parsing algorithm using the java.lang.regex and java.net packages compared to using Google's Guava libraries and the Apache Commons Collections framework.

This performance evaluation of the sessionization algorithm was conducted by comparing the time taken by our log parser and the two leading open source log parsers, AWStats and Analog, to parse and sessionize eight log files of various sizes. Figure 3.6 shows that we were able to outperform both AWStats and Analog when computing basic statistics. In order to determine if these results differ when given larger log files we performed the same test but increased the size of the logs. Figure 3.7 shows the results of our approach, AWStats and Analog when provided with log files of sizes 2.48GB, 4.96GB and 9.92 GB. While Analog only showed a slight increase in the time taken to generate basic statistics from these log files and our approach was still more efficient, AWStats was unable to parse the 4.96GB and

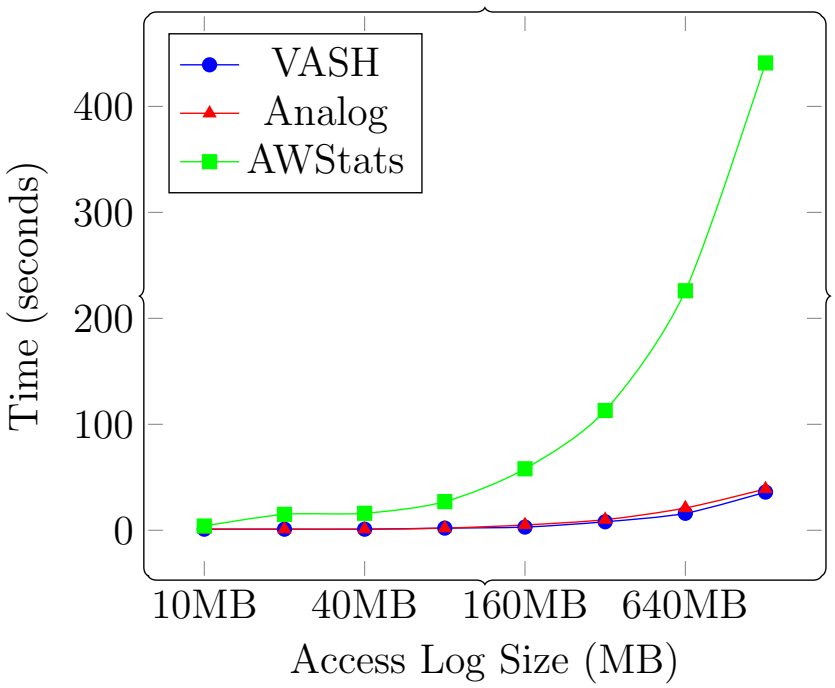

Figure 3.6: Parsing Time (seconds), Basic Statistics

9.92GB access logs due to memory constraints. Figure 3.6 shows that while the performance of AWStats degrades when given larger access logs, our approach and Analog scale linearly regardless of log file size. Due to the size of access logs for web sites with thousands of users and millions of sessions per month it is vital for a framework to be able to handle access logs whose sizes are larger than the system's available memory.

3.9 Generating a Usage Information Hierarchy

A *usage information hierarchy* is a lossless, strongly connected data source that represents a sessionized access log stored in a relational database. Being lossless, the initial web server access log can be holistically created from the information found within the usage information hierarchy. The majority of sessionization techniques create aggregate statistics from the sessionized access log due to the difficulty in efficiently creating and storing a holistic, strongly connected sessionized access log [33, 110, 46]. We created the usage information hierarchy,

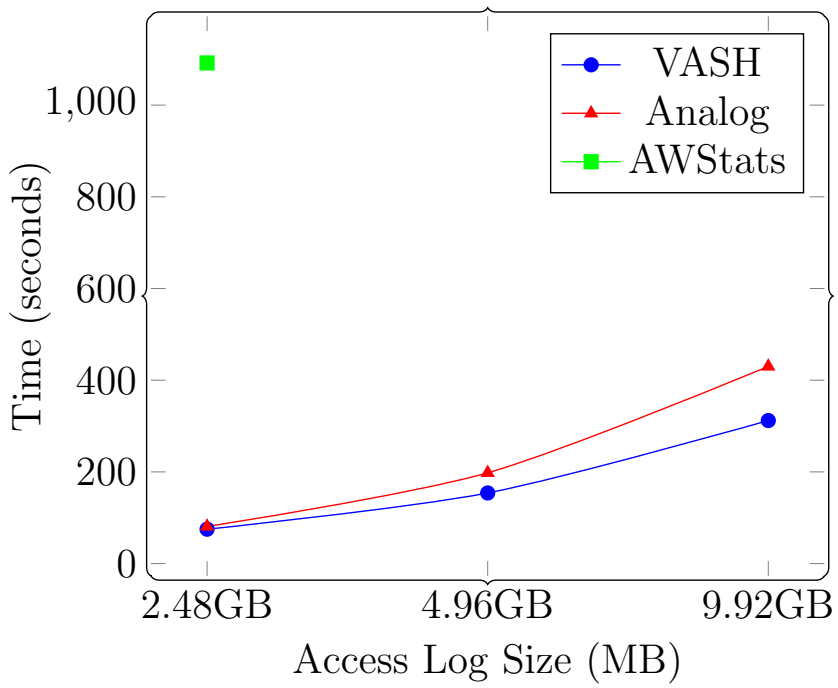

Figure 3.7: Parsing Time (seconds), Basic Statistics

shown in figure 3.8, in order to provide evaluators with the ability to drill-down into, as well as iteratively query against, the usage information derived from web server access logs during the sessionization process.

Specifically, the usage information hierarchy allows evaluators to begin with broad generalizations and then isolate information based on connections found within the hierarchy. For example, if the evaluator needed to determine the *average session depth*, a metric that determines the average number of requests made within a set of sessions, for all visitors within the city of Atlanta, Georgia, they would begin by selecting the average session depths by county, drilling down into the US to view the average session depth for each state, then selecting GA to view its state overview of the average session depth. From there the evaluator could continue increasing the granularity of information by viewing the visitors found within Atlanta, Georgia, viewing the sessions created by each visitor, and ultimately viewing the requests comprising each of the sessions. The hierarchy can be traversed forwards

and backwards. For example, isolating visitors by browser type, isolating sessions by search engine keyword, etc.

In order to utilize the usage information hierarchy, a hierarchically structured sessionized access log must be created. Generating a hierarchically structured sessionized access log requires a great deal of time and resources, thus we extended the notion of sessionization to include highly efficient processes capable of integrating such data into the information usage hierarchy in the form of a relational database.

When relying on the database to incrementally store the hierarchically structured sessionized access log and retrieve the information necessary to legitimately bind information together, such as primary and foreign keys, we suffered significant performance deterioration. In order to utilize the usage information hierarchy we integrated several programming language optimizations and incorporated optimized data loading techniques described in the following section.

3.9.1 Optimizing Data Integration

The most significant obstacle to generating a hierarchical representation of a sessionized access log is the amount of time and memory necessary to store a sessionized access log into a strict, normalized, heavily indexed database with numerous foreign key constraints binding the tables together [99, 55]. To overcome this obstacle, we modify the sessionization algorithm to export the sessionized access log as CSV files, each representing a table within the database. Each CSV is dynamically divided into files that are less than 10MB as the access log is being sessionized, to decrease database insertion latency. We also modified the sessionization algorithm to spawn threads and insert the CSVs as soon as they were created. We explore the performance of our log parsing algorithm in the context of a MySQL database and InnoDB transactional storage engine [87]. MySQL is the most popular open source database available and has been adopted by an overwhelming number of applications

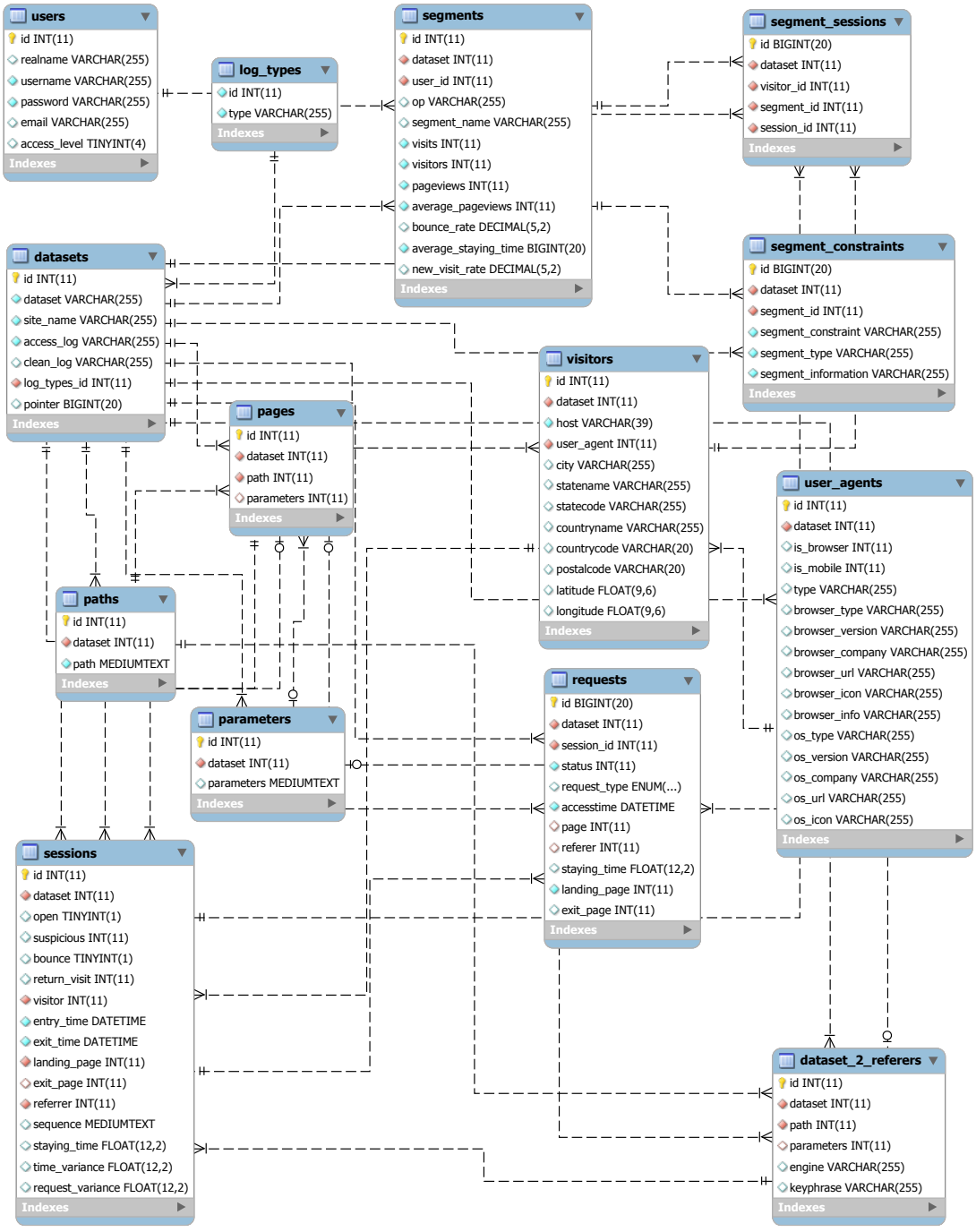

Figure 3.8: Schema for Hierarchically Structured Server Access Log

built on the LAMP (Linux, Apache, MySQL, PHP / Perl / Python) stack. While MyISAM, MySQL's default engine, may appear to be the better choice for our approach, InnoDB's ability to use foreign key constraints significantly reduces the query latency experienced at the end of our procedure when extracting information from the hierarchical information imported into the database. Choosing InnoDB over MyISAM is thus a decision made to benefit the overall performance of the system. InnoDB provides the standard ACID-compliant transaction features and foreign key support (Declarative Referential Integrity) needed to bind the sessionized access log together in an efficient manner [90]. InnoDB's ability to quickly match foreign and primary keys as well as its ability to index single or multiple combined attributes thus makes it possible to create an analytics application that can consistently send complex queries without suffering from severe latency [87]. We first created a configuration that would allow us to maintain the foreign key constraints found within the usage information hierarchy while importing portions of the sessionized access log throughout the sessionization process. Without doing so, we would be barred from importing information at various stages due to InnoDB's strict foreign key constraints that disallow data insertion when the tuple in the table containing the primary key identified by the foreign key constraint can not be located. We explored a number of configurations of MySQL and InnoDB parameter settings and evaluated these configurations for their effect on the efficiency of the SQL process "LOAD DATA INFILE", used in our approach to quickly load the sessionized access log into a normalized database. MySQL provides configuration files *my-small*, *my-large*, and *my-huge* for small, large, and very large systems, respectively [88]. In addition, we created two configuration of our own which we label *my-heavy* and *my-custom*, in which we altered parameters related to InnoDB performance.

The *innodb_thread_concurrency* parameter specifies the preferred maximum number of threads inside InnoDB. The default value of 0 is interpreted as infinite concurrency (no con-

Table 3.2: MySQL InnoDB Configurations

Parameter	small	large	huge	heavy	custom
thread_concurrency	INF	INF	8	8	INF
innodb_doublewrite	1	1	1	1	0
innodb_support_xa	1	1	1	1	0
innodb_buffer_pool_instances	1	1	1	1	2
innodb_write_io_threads	4	4	4	4	64
innodb_buffer_pool_size	64M	256M	384M	2048M	2048M
innodb_additional_mem_pool_size	2M	20M	20M	16M	20M
innodb_log_file_size	5M	64M	100M	256M	256M
innodb_flush_log_at_trx_commit	1	1	1	1	2
innodb_lock_wait_timeout	50	50	50	120	5

currency checking). The *my-small*, *my-large*, an *my-custom* configurations use the defaults, and the *my-huge* and *my-heavy* configurations set a limit of 8.

The *innodb_doublewrite* parameter controls whether or not InnoDB stores all data twice, first to the doublewrite buffer, and then to the actual data files. This feature is enabled by default and in the *my-small*, *my-large*, *my-huge*, and *my-heavy* configurations. In the *my-custom* configuration we disable this feature, which is permissible because we manage the primary and foreign key constraints in memory ourselves during the sessionization process and do not need the redundancy precautions in case of failure because all primary and foreign key constraints are stored within CSVs.

The *innodb_support_xa* parameter enables InnoDB support for two-phase commit in "XA transactions", global transactions that may span multiple resources. This feature is enabled by default and in all configurations except *my-custom* since the integrity of our data is expressed within the CSVs, not memory, and thus can be re-imported if any failure occurs during the import process.

The *innodb_buffer_pool_size* parameter specifies the size in bytes of the memory buffer InnoDB uses to cache data and indexes of its tables. The *my-small* configuration specifies 64MB, *my-large* specifies 256MB, and *my-huge* specifies 384MB. The *my-heavy* and *my-custom* configuration files both specify 2GB. A related parameter is *innodb_buffer_pool_instances*, which controls the number of regions that the InnoDB buffer pool is divided into, if the innodb_buffer_pool_size is 1 GB or more. The buffer pool was significantly increased to allow large amounts of information into InnoDB's buffer from memory to efficiently receive the information being inserted by the data loading threads. Each buffer pool manages its own set of structures for the buffer pool and is protected by its own buffer pool mutex. Dividing the buffer pool into separate instances can improve concurrency by reducing contention as different threads read and write to cached pages.

The *innodb_write_io_threads* parameter controls the number of I/O threads for write operations in InnoDB. The default value is 4, which is used by all of our configurations except *my-custom*, which uses the maximum value of 64 write threads. These multiple concurrent I/O threads benefit the performance of the import process because of the number of 10MB CSVs generated for any given log file.

The *innodb_log_file_size* parameter is a numeric that specifies the size in bytes of each log file in a log group. The *my-small* configuration specifies 5M, *my-large* specifies 64M, and *my-huge* specifies 100M. We specify 256M in both the *my-heavy* configuration and the *my-custom* configuration. The larger the value, the less checkpoint flush activity is needed in the buffer pool, saving disk I/O. Increasing the log file size ensures that the processes will not be temporarily halted in order to expand the log size to hold more transaction and rollback information.

The *innodb_flush_logs_at_trx_commit* parameter can take on the values 0, 1, and 2. With a value of 0, the log buffer is written out to the log file once per second and the flush to disk operation is performed on the log file, but nothing is done at a transaction commit. When

the value is 1 (the default), the log buffer is written out to the log file at each transaction commit and the flush to disk operation is performed on the log file. When the value is 2, the log buffer is written out to the file at each commit, but the flush to disk operation is not performed on it. By default, all default configurations have a trx_commit of 1 and our *my-custom* configuration has a trx_commit of 2 so that no additional disk operations are performed unnecessarily. We also do not incur any additional latency from writing out to the log buffer at each commit since we later disable autocommit.

Since our programmatic logic manages the primary keys and actively polices the foreign key constraints while the log is being parsed, we can ensure that when the data is loaded and the database attributes, such as foreign_key_checks, are re-enabled we will not be penalized or met with a barrage of errors. We also enabled aggressive garbage collection and provided a substantially large heap space for the programs (Xmx/Xms128M). However, we have found that the program works just as efficiently with as little as 64M. Certain Java and MySQL parameters were also set at runtime. Specifically, the *max_heap_size* is increased to allow for large chunks of memory to be used by each process at any point in time. This allows the auxiliary processes, or threads spawned by the main process, to load multiple 10MB files into memory at once and to quickly insert them into the database. Also, *autocommit* is disabled to ensure that tables aren't re-indexed after every insertion into the database. The *foreign_key_checks* foreign key constraint checks are disabled until the process is finished and all the data is successfully loaded into the database to avoid latency issues involved with integrity constraints. Instead of validating the data upon every batch of inserts, the data is instead validated after all the information is loaded. Finally, *unique_checks*, the parameter that controls unique key constraint checks, is temporarily disabled to ensure that primary keys are not actively checked upon every insert.

Given the high number of configurable parameters and ultimately the high number of possible configurations for MySQL and InnoDB, we tested various typical configurations

against our own custom configuration to measure the effectiveness of our approach [87].

Performance Evaluation

Our evaluation computed the time taken to import the hierarchical information generated by our approach into a MySQL database using various configurations. The configurations' parameters, explained above, can be seen in Table 3.2 Our methodology differs from other methodologies in that it creates a complete hierarchical view of the sessionized information that is imported into a database using MySQL's LOAD DATA INFILE statement. This process can be rather time consuming using the basic configuration for a database, so we've compared various packaged configurations with our own configuration to determine the time taken for each. Figure 3.9 shows the time in seconds taken to import the CSVs into a MySQL database using the five configurations previously described. Given the results, we were able to determine that our configuration leads to a more efficient data integration. Figures 3.11 and 3.10 provide similar views of the results but with the *my-small* configuration removed to better visualize the advantages our configuration has over the other configurations provided with MySQL. We believe this is due to the increased concurrency and lack of concurrency limit checking, the additional buffer pool and write I/O threads, and the fact that our configuration disables doublewrite, xa, and transaction commits that often increase the latency experienced by large data insertions with LOAD DATA INFILE due to the need to validate or record database insertions. Our methodology handles these validations within the log parsing process, allowing us to disable these parameters since they are no longer required for the data integration process.

3.10 Conclusion

Usage data preprocessing consists of several distinct procedures that when employed translate web data resources, such as web server access logs, into usage information evaluators

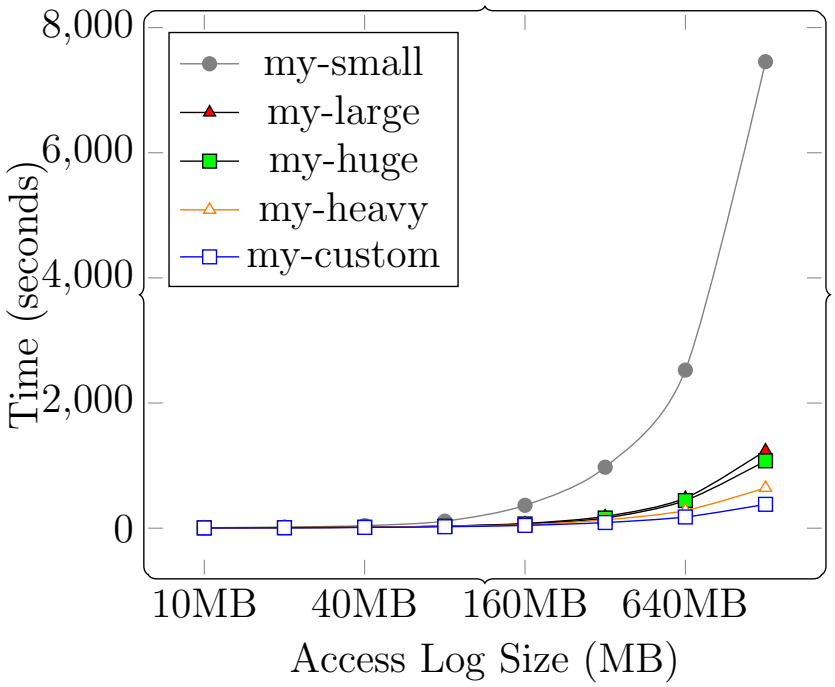

Figure 3.9: Import Time (seconds), Hierarchical Data, All

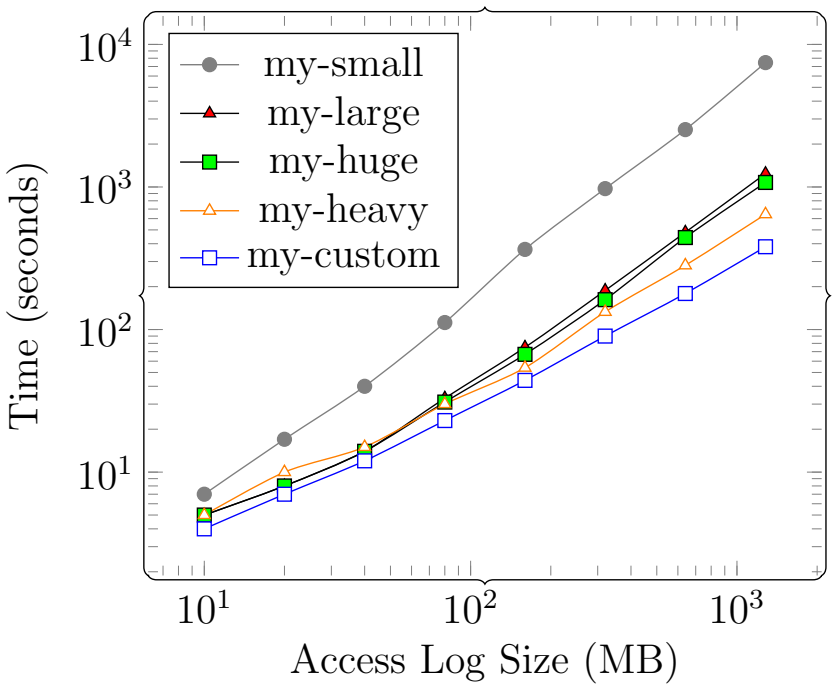

Figure 3.10: Import Time (seconds), Hierarchical Data

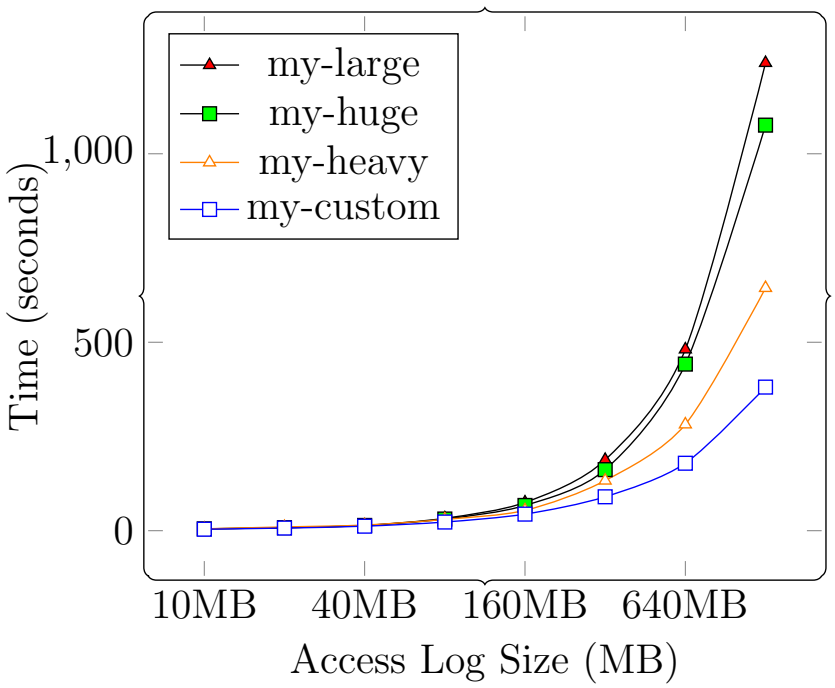

Figure 3.11: Import Time (seconds), Hierarchical Data, Without my-small

can analyze and better understand. Our research in the area of preprocessing is comprised of increasing the accuracy of visitor type identification and increasing the relevancy of visitor sessions through the use of localized behavioral heuristics, increasing the efficiency of sessionization algorithms, and creating an information hierarchy capable of storing a hierarchically structured server access log.

The contributions made by our research to the area of usage data preprocessing are as follows:

1. Java optimizations and language extensions for preprocessing, parsing, and sessionization techniques.

2. Integrating third party packages to increase efficiency and decrease the processes footprint.

3. Creation of the information usage hierarchy.

4. Optimal configurations for integrating hierarchically structured sessionized access logs into relational databases.

5. Several behavioral heuristics that more effectively distinguish bots from humans and more accurately define visitor sessions.

Chapter 4

Unsupervised and Semi-Supervised Classification

Clustering algorithms take as input a set of m items, $x = \{x_1, x_2, ..., x_m\}$, and a similarity measure between pairs of items $Sim(x_i, x_j)$, and produce a partitioning of the item set into clusters so that each member of set x is classified, assigned to one and only one cluster, in a way that maximizes some objective function with respect to the similarity measure.

In unsupervised clustering the correct number of clusters may not be known. Unsupervised prototype-based clustering tries to determine the correct number of clusters, k, without prior knowledge, by applying one of four approaches:

- by repeating the clustering for several values of k and using a validity measure to choose the best partition;
- by performing several passes through the data set, seeking one cluster at a time and then removing these classified points from the data set of the next pass if the newly defined cluster passes a validity test as in the GMVE[60];
- by starting the clustering process with an over-specified number of clusters, and then merging similar clusters and eliminating spurious clusters, as in Compatible Cluster Merging[66];
- by partitioning the data set into an over-specified number of clusters, and then allowing adjacent clusters to compete for data points, with those that lose the competition eventually being removed, known as Competitive Agglomeration[43].

The first two approaches suffer from the problem of designing validity measures for goodness-of-fit that do not require prior knowledge of an inlier or of threshold settings appropriate for the particular data set or domain. For purposes of evaluating the impact of changes to a web site on particular types of users, such unsupervised clustering may suffer

from the problem that clusters that are generated may not be based on the attributes of interest to the evaluators. Thus, we take an approach, known as SCFC, that is considered "semi-supervised" clustering, in that it uses a small amount of labeled, or classified, data to aid and bias the clustering of unlabeled data. The objective of the clustering process, then, is to optimize class purity by creating well defined, tightly bound clusters that are distinctly dissimilar[9]. Existing research on semi-supervised clustering falls into two major categories: similarity-based methods and search-based methods[10]. While similarity-based methods create a modified distance function that incorporates knowledge from the classified examples and use a traditional clustering algorithm to cluster the data, search-based methods modify the clustering algorithm itself but do not change the distance function.

4.1 Hierarchical Unsupervised Niche Clustering (H-UNC)

Nasraoui et al.[79, 80] developed and employed Hierarchical Unsupervised Niche Clustering (H-UNC), which they describe as "a technique created to exploit the symbiosis between clusters in feature space and genetic biological niches in nature"[80]. In this approach, sessions are encoded as N_u-dimensional binary attribute vectors, where u is the total number of valid URLs in the set of sessions under consideration.

Syntactic similarity, $S_u(i, j)$, is a measure of the similarity between two requests, based on the structure of the URLs used to receive the requests. A matrix of these similarity values is generated, containing the similarity value for each pair of URLs in the universe of URLs under consideration.

Session-level similarity, $S_{k,l}$, between two user sessions k and l is calculated as the maximum of two measures: 1) $S_{1,kl}$, a similarity measure that evaluates the extent to which the same URLs appear in the two sessions, k and l; and 2) $S_{2,kl}$, a similarity measure that relies on the syntactic similarities between the pages that make up the sessions k and l. The final value of session similarity between sessions k and l, $S_{k,l}$ takes the max of $S_{1,kl}$ and $S_{2,kl}$.

With these similarity measures in mind, the similarity of two sessions is used to compute the dissimilarity measure, shown below.

$$d_s^2(k,l) = (1 - S_{kl})^2 \tag{4.1}$$

The user sessions are assigned to the closest clusters based on the computed distance d_{ik}, from the *i-th* cluster to the *k-th* session. The algorithm begins with a single cluster, creates the initial prototypes and final prototypes, which are potential centroids for each subsequent, potential cluster, initializes the set of mean squared errors, or minimum acceptable fitness value for each cluster, and clusters recursively using UNC, an algorithm that finds dense clusters and determines the number of clusters automatically[86]. H-UNC assigns sessions to the closest prototype, recomputes fitness values every iteration and continues on with a set of clusters that have the highest fitness value found. Once the algorithm has completed, the clusters generated have the highest possible fitness value and best distance between clusters found. A more detailed description of UNC and H-UNC may be found in[86, 80].

4.2 Semi-supervised Competitive Fuzzy Clustering (SCFC)

For our research in [108] we adapted Semi-supervised Competitive Fuzzy Clustering (SCFC) from the powerful similarity measures found in Nasraoui's CARD[79] and H-UNC[80] algorithms. First, in SCFC we have adapted the notions of syntactic and session similarity. Our notion of syntactic similarity shown in equation 4.2 differs from that of Nasraoui's CARD and H-UNC algorithms in two specific ways: We include the full URI, as opposed to removing the page from the request and observing only the directory structure, and we rewrite the URL to include parameters. Specifically, we cut the trailing parameters from the URL, sort them, and attach the parameters back onto the URL as part of the directory structure. Currently, the sorting of parameters is alphabetical. However, this sorting could be based on latent semantic

indexing of pages as in[100]. In this way, we are able to provide a greater distinction among pages that exist in the same directory or that are generated in the same way but with different parameters. For example, the URL "http://example.com/one/two/page.php?b=2&a=3" would be translated to the string "http://example.com/one/two/page.php/a/3/b/2" rather than to "/one/two/" as in[80]. The equation for syntactic-level similarity is shown below.

$$S_u(i,j) = min[1, \frac{|(p_i \cap p_j)|}{max(1, max(p_i, p_j))}] \tag{4.2}$$

In addition, the session-level similarity measure shown in equation 4.3 below has been optimized to account only for the syntactic similarities of URLs contained within both sessions. This does not affect the result, it merely saves time when executing the algorithm.

$$S_{kl} = \frac{\sum_{i=1}^{|s^k|} \sum_{j=1}^{|s^l|} S_u(i,j)}{|s^k| * |s^l|} \tag{4.3}$$

A distinguishing feature of SCFC is that we rely on evaluators (typically site administrators) to provide input to the algorithm. The evaluator first chooses k, the number of clusters, each cluster representing one *type* of user for whom the evaluator wishes to assess the impact of changes to the web site. Next, the evaluator selects one or more sessions as representatives of each cluster *1..k*. These sessions serve as seeds for the formation of clusters.

The procedure for SCFC can be summarized as follows:

1. Retrieve a list of sessions from the access log, sessions $S_0...S_{N-1}$, where N is the total number of sessions.
2. Allow the evaluator to specify k, the number of clusters, and to select j representative sessions, $j >= k$, such that each cluster is represented by at least one of these selected sessions.
3. Calculate the syntactic similarity for each of the *(N-j)* unclassified sessions in our list of sessions $S_0...S_{N-1}$ paired with each of the j representative sessions.

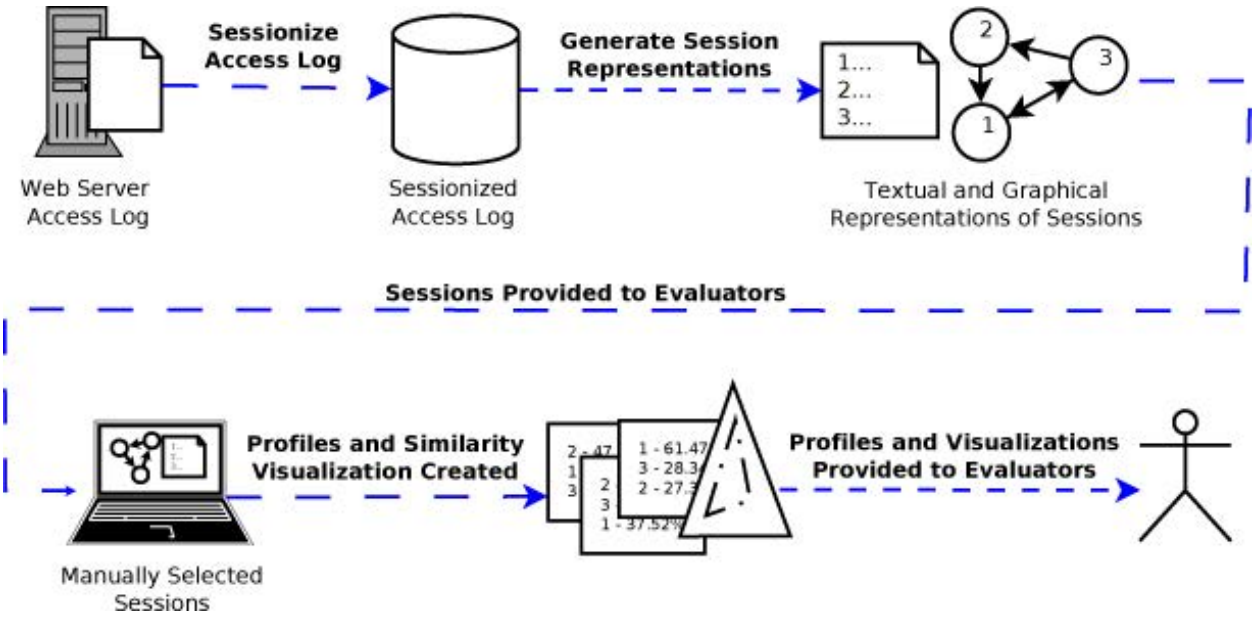

Figure 4.1: Behavioral Clustering and Profile Generation

4. Using the session similarity measures defined for SCFC, obtain a value for each of the *((N-j) * j)* pairs and classify each of the *N-j* unclassified sessions by assigning them to the cluster for which the highest value of session similarity is obtained.

To gauge the quality of our approach, we compared the clusters and profiles generated by H-UNC to those created by SCFC.

4.2.1 Generating Profiles of Clusters

The behavioral clustering and profile generation technique, shown in figure 4.1, allows evaluators to group user sessions based on the session's content. Specifically, evaluators manually select N sessions that exhibit characteristics they believe to be indicative of a user archetype, such as novice, intermediate, and advanced users. The framework then seeds the SCFC algorithm with these manually selected sessions to generate N clusters. Profiles and basic visualizations of the clusters are then generated by the framework.

To facilitate the manual selection process for the evaluators, we created a program that

randomly selects and presents both the graphical and textual representation of a session. The program provides the evaluators with the ability to either select a session as a user archetype or populate the interface with a new textual and graphical representation of a session.

1. Obtain and sessionize a server access log.
2. Generate textual and graphical representations for all sessions.
3. View textual and graphical representations of the sessions and select sessions that exhibit behavior indicative of a type of user.
4. Cluster the remaining sessions by seeding the SCFC algorithm with the sessions selected by the evaluators.
5. Create profiles from the sessions comprising each cluster generated by the SCFC algorithm.
6. Create a visual representation of the remaining sessions' similarity and dissimilarity to the sessions selected by the evaluators.
7. Present the profiles and visualizations of the results for analysis.

4.2.2 Comparing H-UNC to SCFC

Both SCFC and H-UNC were then used to generate clusters of the sessions, based on the navigational patterns found within the sessions. Parameters were selected for the H-UNC algorithm to aid in generating a relatively small number of clusters to allow a more direct comparison of the contents of the clusters generated by H-UNC with those generated by SCFC.

The H-UNC clustering algorithm described in section 4.1 was written in Java 1.6 based on the algorithm's description in Nasraoui's research regarding Hierarchical-Unsupervised Niche Clustering [86].

Table 4.1: Sensitivity and specificity of clusters

	Novice	Intermediate	Advanced
H-UNC Sensitivity	0.546	**0.325**	**0.500**
H-UNC Specificity	**0.432**	0.110	0.325
SCFC Sensitivity	**0.689**	0.200	0.375
SCFC Specificity	0.329	**0.755**	**0.928**

For the SCFC algorithm, the site administrators for CryptoDB.org specified three *types* of users, which they termed novice, intermediate, and advanced. A representative session was selected for each of these three clusters. The attributes of interest in defining the user types were behavioral, based on the types of requests. Three particular request types were specified by the evaluators as of particular interest: GBrowse requests, Download History requests, and Query History requests. A novice session representative was selected by choosing from a list of sessions that had GBrowse requests but no Download History requests or Query History requests. An intermediate session representative was selected by choosing from a list of sessions that had GBrowse requests and either Download History requests or Query History requests, but not both. An advanced session representative was selected by choosing from a list of sessions that had all three request types. The SCFC clustering classified 299 sessions as novice, 103 as intermediate, and 34 as advanced. Table 4.1 shows the sensitivity and specificity of the SCFC and H-UNC algorithms, where bold values indicate the higher values, using the manual clustering as a basis for comparison.

The H-UNC algorithm generated six clusters. To assign a type to the H-UNC clusters we examined the profiles and the representation of manually classified sessions in each cluster. The assignment of class to cluster was not obvious based on profile. We thus selected the assignment of H-UNC cluster to user class that resulted in the highest sensitivity and specificity values. Three clusters were classified as novice, one cluster was classified as intermediate, and two clusters were classified as advanced. Table 4.1 shows that H-UNC exhibited better performance for sensitivity for the intermediate and advanced groups while

SCFC exhibited better performance for specificity for the intermediate and advanced groups, while the opposite is true for the novice group.

The seeding of the clusters with representative sessions for the user behavior of interest allowed the simplistic version of SCFC employed in this study to outperform the more sophisticated H-UNC algorithm for this purpose.

4.2.3 Case Study: Determining Effectiveness of SCFC

In order to determine the effectiveness of SCFC we performed a comparative analysis between two types of clustering algorithms. The first is Nasraoui's hierarchical unsupervised niche clustering algorithm (H-UNC) which uses a basic genetic algorithm (DC) to create children from randomly selected sessions then clusters the remaining sessions based on a series of equations taking dissimilarity, fitness, and weight into consideration. The second is a derivation of Nasraouis H-UNC algorithm called supervised competitive fuzzy clustering (SCFC), a supervised approach relying on the syntactic and session level similarity of sessions selected by a supervisor, or someone who is intimately familiar with the web site being studied.The procedure is defined below.

1. Sessionize a chunk of the CryptoDB server access log
2. Have a supervisor select three sessions representative of novice, intermediate, and advanced visitors
3. Use competitive fuzzy clustering (CFC) to group the remaining sessions based on similarity to each of the three representative sessions
4. Use Nasraouis H-UNC algorithm to cluster sessions and generate profiles
5. Compare the clusters generated using the SCFC algorithm with that of Nasraouis H-UNC algorithm.
6. Create profiles from the three clusters generated using the SCFC algorithm

The procedure required the sessionization of a chunk of the CryptoDB access log and the manual selection of three sessions based on the level of expertise exhibited by the visitor. After three sessions were selected, representing a novice, intermediate, and advanced visitor, the SCFC algorithm was applied to the data set to create three clusters. The same data set was also provided to the H-UNC algorithm to generate N-number of clusters and their respective profiles.

As a final step, observations were made concerning the output of the two algorithms. Comparisons were drawn by superimposing the clusters generated by the SCFC algorithm with those generated by the H-UNC algorithm. We generated graphs showing the similarity of the remaining sessions to the three representative sessions and provided graphs profiles of each group of clusters showing the strength of the URLs. The entire procedure is defined below.

1. The CryptoDB access log was obtained through rysncing /var/log/httpd/mnt/cryptodb/accesslog.clean from Loquat(loquat.rcc.uga.edu) to Lime (lime.cs.uga.edu).

2. A chunk, dating from June 1st, 2007 to June 30th 2007 was taken from the access log obtained from Loquat. This chunk was then sessionized into 2056 sessions. 312 of which will be used for the experiments, all of which are sessions with at least 5 requests and no more than 50 requests. The request threshold was chosen to separate single request, or low request count, sessions because of their lack of content, as well as sessions having an abnormally large request count because of the sessions themselves being generated by bots or scripts.

3. From the sessions identified through sessionization, and the subset of sessions isolated for the purposes of experimentation, three sessions were selected by the supervisors of the CryptoDB web site to represent a novice visitor session, an intermediate visitor session, and an advanced visitor session. The sessions were selected based upon the

existence of certain URLs within the session itself. There were three URLs specifically which were used to isolate three sessions, which were Gbrowse URLs, download history URLs, and query history URLs. If a session contained a Gbrowse URL, but neither of the history URLs, it was deemed a novice visitor session. If a session contained Gbrowse and either of the history URLs, it was deemed an intermediate visitor session. If a session contained all three types of URLs, if was deemed an expert visitor session.

4. The remaining sessions were clustered based on their session level and syntactic level similarities. The clustering algorithm used was our SCFC algorithm.

5. Generating profiles: Profiles were generated from the three clusters showing the percentage of each URL within the sessions contained within the clusters. For instance, each session within a particular cluster would be checked to see if it contains the homepage, whose URL is / or /cryptodb/. If the cluster contains 10 sessions, and 8 of the sessions contain the homepage, then the homepage URL would have 0.8 as a value in the clusters profile.

Results

A subset of sessions isolated for these experiments were clustered with the SCFC algorithm, which was seeded with three sessions previously selected and discussed. Graphs were generated to show how similar each session is to the three representative sessions. Additionally, profiles were generated to show the top 10 URLs within each cluster to help identify key differences between the three clusters.

Analysis and Conclusions

Figures 4.2 depicts the path analysis diagrams showing the traversal of a visitor for three particular sessions, each representing a type of visitor. We can see that the more requests a

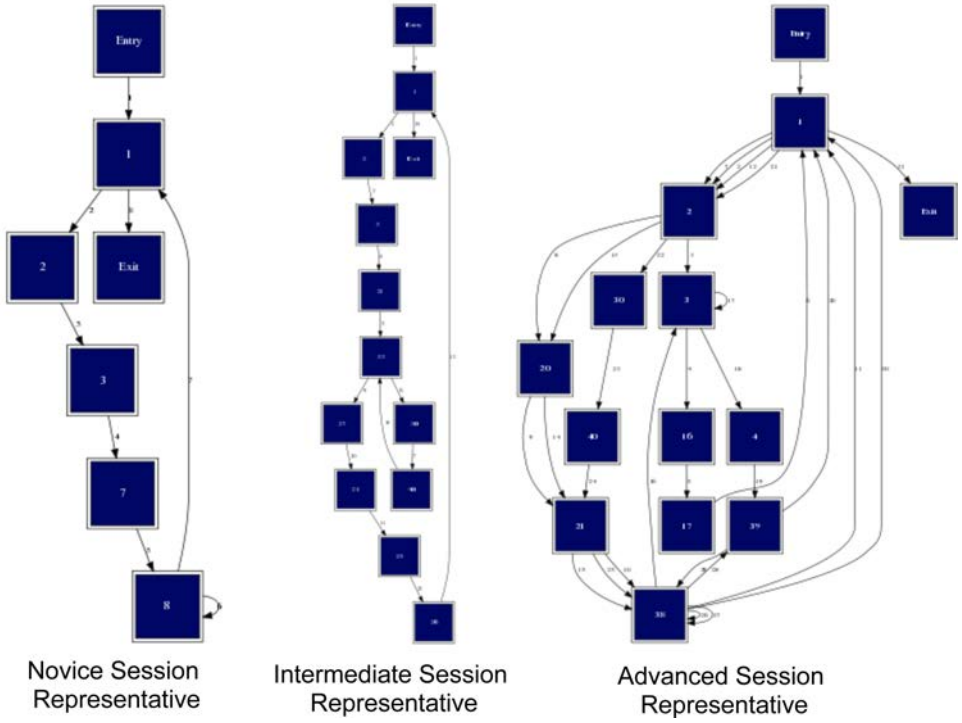

Figure 4.2: Visitor Traversal Patterns - Novice, Intermediate, Advanced

session has, and thus the more complex the traversal through the web site, the more likely the visitor is either familiar with the web site or is lost within the web site. This is not always the case, however, as there were a few sessions containing all three of the specified URLs used to type the visitor which had very few nodes.

Figure 4.3 shows us a radar of session similarities, where the three corners of the triangle are the three sessions in question, and the scale is the ratio of similarity of a session to each of the three representative sessions. Here we can see that almost all of the sessions are simply more similar to a particular representative session, as opposed to being grossly similar to a representative session a grossly dissimilar to the other two representative sessions.

This helps understand one of our previous assumptions: just because something is not A, does not mean it is B. We can only say that if it is closer to B, it is most similar B, not that it is dissimilar to A, and the radar graph helps us understand why. For instance,

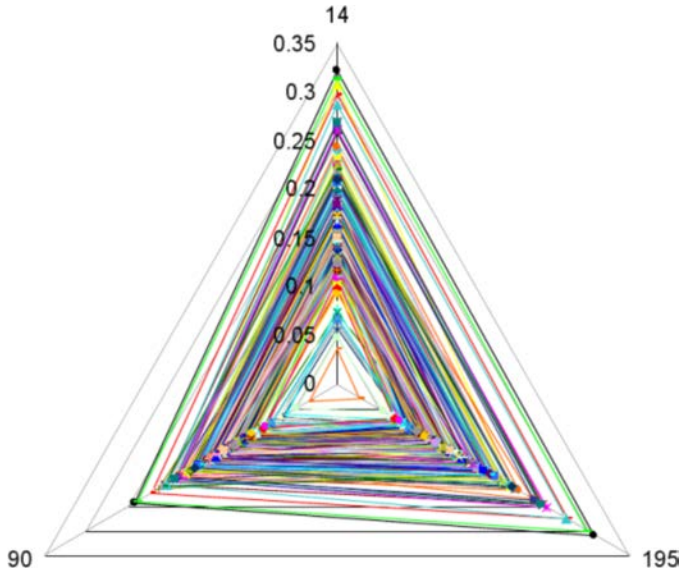

Figure 4.3: Radar of Session Similarities

observe the outer lime green triangle and notice that although it is closer to the intermediate session representative, 195, it is almost as close to the novice session representative, 14, and furthest away from the advanced session representative, 90. This makes for a much more powerful assumption because instead of positing in absolutes (if not A, then B), we can say that although it is B, it looks as if it is evolving from A. This will help us understand how visitors are evolving with the site, and might help indicate whether changes to the site have hindered or helped particular types of visitors become more familiar with the sites structure, content, and functionality.

Figure 4.4 helps visualize the difference between the three similarity ratios for each particular session, and helps us understand where the majority of our visitor base stands with regards to their familiarity with the web site. For instance, between .25 and .1 similarity, most of the sessions have a higher novice similarity with a large difference between that and the intermediate similarity, and the advanced similarity. If we started seeing these lines shrink, we could assume that the visitors were becoming more familiar with the web site. Visualizing these sessions over time to see if this is, in fact, the case would be an excellent

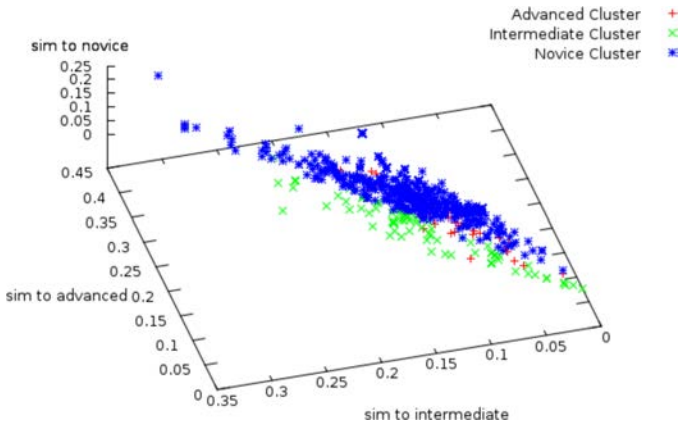

Figure 4.4: Novice, Intermediate, and Advanced Cluster Distances

way to see if the changes to the web site are helping or hindering our visitors.

The tables below shows the top ten URLs within each profile generated from each cluster of sessions grouped by novice, intermediate, and advanced session representatives respectively. We notice that all three profiles have the homepage and the browse page as their top two URLs, showing similarity between the three types of visitors.

Table 4.2: Novice Cluster Profile

Probability	Request
0.98	gbrowse
0.81	/cryptodb/
0.59	showRecord.do?name=GeneRecordClasses.GeneRecordClass
0.55	/
0.30	/cryptodb/processQuestion.do
0.23	processQuestionSetsFlat.do?questionFullName=GeneQuestions.GeneByLocusTag
0.17	showQuestion.do?questionFullName=GeneQuestions.GenesByTextSearch
0.15	showXmlDataContent.do?name=XmlQuestions
0.14	showQuestion.do?questionFullName=GeneQuestions.GenesBySimilarity
0.13	/cryptodb/queries_tools.jsp

The differences, however, help explain what makes an advanced visitor advanced, and a novice visitor novice. The advanced visitor profile is 4 times more likely to visit showXmlDataContent, which is a type of web service and arguably a more complex function to utilize within the web site, and the novice visitor profile shows higher values for simple operations,

Table 4.3: Intermediate Cluster Profile

Probability	Request
0.92	/cryptodb/
0.74	gbrowse
0.69	/cryptodb/queries_tools.jsp
0.51	/cryptodb/processQuestion.do
0.49	/
0.36	/cryptodb/showQueryHistory.do
0.33	/cryptodb/srt.jsp
0.33	showXmlDataContent.do?name=XmlQuestions
0.31	/cryptodb/help.jsp
0.31	/cryptodb/showRegister.do

Table 4.4: Advanced Cluster Profile

Probability	Request
0.85	gbrowse
0.75	/cryptodb/
0.60	showXmlDataContent.do?name=XmlQuestions
0.60	/
0.50	showRecord.do?name=GeneRecordClasses.GeneRecordClass
0.45	/cryptodb/processQuestion.do
0.25	/cryptodb/showQuestion.do
0.20	showQuestion.do?questionFullName=GeneQuestions.GenesBySimilarity
0.20	/cryptodb/queries_tools.jsp
0.15	/cryptodb/showQueryHistory.do

such as showing records and processing questions. Another interesting point is that the intermediate visitor profile was the only profile to have the help.jsp URL within the top 10 URLs, whereas the novice visitor profile had help.jsp as the 39th most popular URL and the advanced visitor profile did not have help.jsp listed at all. Through sessionization of a single month within the CryptoDB access log we were able to successfully cluster out three types of visitors and go on to make pertinent observations of their relation to one another as well as their profiles.

With this report as a basis, the current research could easily be used as a spring board to further explore the differences, and relation, of each type of visitor to the next, as well as the evolution of types for specifics visitors over time.

4.3 Webanalyzer: Java Wrapper for Cluster 3.0

Webanalyzer is a Java wrapper application for the open source clustering software *Cluster 3.0* written by Naveed Ahmed. Cluster 3.0 was adapted by Michiel de Hoon from Michael Eisen's original implementation of Cluster [38]. The wrapper provides functionality to execute various clustering algorithms, such as k-means, using various distance measures, such as Euclidean distance, as well as functionality to compute sensitivity, specificity, and the average of these two values for the resultant clusters.

4.3.1 Case Study: Determining Effectiveness of Webanalyzer

The Division of External Affairs at the University of Georgia [85] advances the goals, objectives and priorities of the university by generating private financial support, building and maintaining relationships with alumni and donors, communicating UGA's strategic messages to various audiences, assisting students with career development opportunities and meeting the needs of visitors through guided tours of the campus.

We selected the Division of External Affairs due to the department's relevance to our research. Specifically, we believe that a detailed analysis of the usage patterns and online behavior extracted from the web server access logs generated by the Division of External Affair's web site would benefit their community. We also believe that the site's content is compartmentalized in a way that would generate distinct patterns of behavior, and the site's structure is large enough to produce questions that may be difficult to answer without a thorough analysis of the site's users' behavior.

We applied several structure and usage mining techniques to the access logs provided by the Division of External Affairs and generated a report to present to the department's point of contact. The information contained within the report was created by the procedures explained in the following sections. We will begin by itemizing the entire procedure employed

for this case study, explain the techniques and algorithms used to mine the server access logs as well as the information these techniques generated, then discuss our results of both our independent results and the insights and conclusions drawn by the Division of External Affairs' point of contact.

Our methodology for analyzing the Division of External Affairs' server access logs is as follows:

- Request and obtain the server access logs generated by the web server handling the requests made to http://alumni.uga.edu
- Due to the segmented nature of the access logs, order the requests found within the access logs by date, remove any superfluous information, such as JavaScript and CSS includes, that do not contribute to the usage patterns or online behavior of the users, then merge the logs into a single server access log
- Sessionize the merged, cleansed access log
- Generate multiple views of the information to facilitate easy access for both our algorithms and the application, Ajaxalytics, that uses the information directly.
- Generate the transitions and transition probabilities for all of the sessions created by a user for each user.
- Group sessions using various clustering algorithms and distance metrics to determine the highest quality cluster groups, where quality here is determined by the highest average of cluster sensitivity over cluster specificity.
- Group sessions based on attributes we considered to be characteristic of users who recently graduated and users who may have graduated several years ago. The procedure used for this grouping is described in section 4.3.1.
- Create profiles, detailed in section 2.1.3, for each grouping to provide a generalized breakdown of the frequency of occurrence for the pages comprising the groups constituents.

- Create graphical representations of the groupings annotated with the pages, transitions, transition probabilities, and entry and exit points.
- Generate a list containing several common subpaths found within all the sessions generated from the Department of External Affairs access logs
- Generate a view within Ajaxalytics and allow Department of External Affairs to explore their information directly.

Clustering Methodology Application

Every web server generates logs which are user interactions with the site. These web server logs are cleaned and sessionized. Sessionized logs are then extracted and transformed into database tables containing the session information using VASH. A list of all available sessions is extracted and provided to the Java Wrapper tool Webanalyzer to run Cluster 3.0. The Cluster 3.0 software then clusters the sessions using different distance measures and selects the clustering methodology having the highest weighted average of sensitivity and specificity. For this investigation, the Spreaman's Rank distance measure using k-means pair wise clustering, where k is 2, generated the highest quality clusters.

Determining Highest Quality Clustering Algorithm

Apache web server logs are required to be cleaned before they can be analyzed. They are then sessionized using VASH. VASH processes the web logs into logical user sessions. The input to VASH is a web server log. The output of VASH is a database dump file. This database dump file contains information such as the web URL, request time, the user IP address accessed, and the entry and the exit times of the users including various other information in database tables. Our software Webanalyzer takes this session information from the database and prepares an input file for the Cluster 3.0 software (an open source software).We performed a case study using the Apache server logs for the External Affairs web site. The

logs cover time period from May-June 2010 and include over 30,000 sessions. We clustered sessions from these logs to obtain "predicted labels". We annotated each of this sessions as coming from 1) Inside Georgia (GA) versus Outside Georgia (NGA) and 2) Inside Athens (ATH) and Outside Athens (NATH). Our Webanalyzer software then takes as its input a 1) sessionized web server log, 2) the maximum number of clusters, and a 3) set of distance measures and runs the Cluster 3.0 software for the different distance measures.. The cluster 3.0 software then performs k-means clustering, a popular clustering algorithm, for each of the distance measures, producing groups of similar user sessions.We then evaluated the performance of the clustering algorithm for the various distance measures using quality measures such as sensitivity and specificity and their weighted average. For k-means algorithm we looked at 7 different measures which are uncentered correlation, Pearson correlation, uncentered correlation (absolute value), Pearson correlation (absolute value), Spearmans rank correlation, Euclidean distance and Manhattan distance. We ran the sessions for subsets of the total sessions of varying size.

The clustered sessions were then taken by our tool Webanalyzer, which calculates the sensitivity (proportion of actual group members found) and specificity (proportion of predicted members that are correct) of each clustering result produced by the software. Also, we calculate the weighted average of sensitivity and specificity for each distance measure. The tool identifies the clustering result having the highest weighted average of sensitivity and specificity for each distance measure. In this way, users can select the best distance measure for an unlabeled data set. Running time taken by each distance measure to perform the clustering was observed and plotted. Given a labeled set of web sessions that we believe has similar properties to an unlabeled set that we wish to label [for which we know the user group characteristics before hand], a web site administrator can take this information and evaluate the distance measures and apply to unlabeled cluster sets. In reality, the web site administrator may choose to differentiate on other parameters such as former vs recent

alumni and buyers vs browsers. or simply evaluate the user profiles of the web site users. We believe this tool will help the web site administrator in choosing the right parameters to in order to generate user profiles and cluster user sessions.

This process generated two clusters, and the profiles for these clusters are located in tables 4.5 and 4.6.

Grouping Based on Behavior

To compare the accuracy of the clusters automatically generated by the method described in section 4.3.1 as well as generate additional views of the usage behavior, we created an algorithm that would create two distinct groups of users where one group contains all visitors assumed to be new alumni and the other contains all visitors assumed to be old alumni.

In order to create the two groups containing new and old alumni separately, we first isolated users based on their physical location. We assumed that anyone within Athens, Georgia, where the University of Georgia exists, should be considered and checked to determine if they are a new alumni, and that anyone within the state of Georgia that is not within Athens, Georgia, should automatically be considered and checked to determine if they are an old alumni. To create the initial groups, we selected all sessions created by users that currently live in Athens, Georgia, and users that currently live in Georgia but outside of Athens. The algorithm used for creating these groups can be found in libraries/GroupAlumni/GroupAlumni.java within the Ajaxalytics package.

To further isolate these sessions we manually selected several pages from http://alumni.uga.edu that we considered pages that would likely be requested by the types of users being analyzed. The manual selection and subsequent refining of sessions is explained in section 4.3.1 below.

Manual Log Analysis

The *flag pages*, pages that we considered would be likely requests made by new and old alumni, are shown in the lists below. Note that the asterisk attached to each page indicates that a substring search was used for each page, meaning that if the user's requests *contained* the flag page then it was considered a positive match for the flag page in question.

- http://alumni.uga.edu/alumni/ring*
- http://alumni.uga.edu/alumni/student*
- http://alumni.uga.edu/alumni/young*

- http://alumni.uga.edu/alumni/programs*
- http://alumni.uga.edu/alumni/tours*
- http://alumni.uga.edu/alumni/regional*
- http://alumni.uga.edu/alumni/products*
- http://alumni.uga.edu/alumni/atlanta*
- http://alumni.uga.edu/alumni/womenofuga*

To properly implement the wild card nature of the flag pages, we first collected all pages that contained any one of the flag pages listed in the figures above. To do so, we extracted and stored all distinct page ids from the *pages* table that matched or contained a partial match of the pages listed above as *UFPIs*, unique flag page identifiers.

Once the UFPIs for each list of flag pages was stored, we refined the current set of sessions assumed to be created by new and old alumni by checking for the existence of the UFPIs within the sessions individually. We set the threshold for each session to remain a constituent within the group to be 20% of the UFPIs for the corresponding user type. If the session did not contain at least 20% of the UFPIs, the session was removed accordingly. For example, if a session was created by a user living in Athens, Georgia then we assumed it was created by

a new alumni and placed it in the group containing sessions created by new alumni. If that session does not contain at least 20% of the UFPIs then the session would be removed. The same process was applied to the sessions within the old alumni session set using the UFPIs created using the old alumni flag pages.

This process generated two clusters containing sessions we believe had been created by both new and old alumni, where sessions created by new alumni were in the first cluster and sessions created by old alumni were in the second cluster. The profiles for these clusters are located in tables 4.7 and 4.8.

Creating Models per Group

Models were created from the clusters generated by the procedures detailed in section 4.3.1. Using the list of distinct session ids for each group, the requests for each session were extracted and merged into a schema containing the sessions' combined requests, transitions, and transition probabilities.

The models inherit the generated schemas and are then translated by our software to create graphical representations visualizing the information in a manner that allows evaluators to easily discern differences in usage behavior. Specifically, the models were created to help the Department of External Affairs better understand their users.

Creating Profiles

Profiles, explained in section 2.1.3, were also created from the clusters generated by the procedures above. The profiles provide a higher level view of the usage patterns exhibited by the site's visitors. With these profiles we are able to see the frequency of occurrence for several popular pages for each group individually and thus determine specific areas within the site each group found to be of some interest or importance.

Below are the lists containing the profiles created from the automatically generated clus-

ters. We note that the clusters are highly related based on the highest ten elements within their profiles, but did notice subtle difference in the behavior exhibited by the clusters' constituents. In general, however, it appears that the behavior of the users who generated the sessions used to create the profiles below is largely similar.

Table 4.5: Profile of First Cluster

Probability	Sessions Within	Request
.14791	610/4124	/alumni/
.12827	529/4124	/alumni/home.php
.08826	364/4124	/alumni/enews/
.04583	189/4124	/alumni/benefits.html
.03395	140/4124	/alumni/products.html
.02716	112/4124	/alumni/phpsearch/search.php
.02546	105/4124	/alumni/programs.html
.01988	82/4124	/alumni/regional.php
.01964	81/4124	/alumni/board.html
.01964	81/4124	/alumni/about.html

Table 4.6: Profile of Second Cluster

Probability	Sessions Within	Request
.37221	1682/4519	/alumni/
.33193	1500/4519	/alumni/home.php
.16751	757/4519	/alumni/enews/
.10976	496/4519	/alumni/benefits.html
.0821	371/4519	/alumni/products.html
.06041	273/4519	/alumni/phpsearch/search.php
.05753	260/4519	/alumni/programs.html
.05112	231/4519	/alumni/about.html
.0416	188/4519	/alumni/regional.php
.04005	181/4519	/alumni/contact.html

Below are the lists containing the profiles created from the clusters generated by the manual log analysis. We note that by focusing our attention on a small set of flag pages to generate the UFPIs we seem to have captured sessions created by users exhibiting behavior that we would expected from recent and seasoned alumni. However, it appears that our attempt to isolate sessions created by recent alumni captured several sessions created by

existing students, as per the high frequency of occurence for several pages of interest for the current students.

Table 4.7: Profile of New Alumni Cluster

Probability	Sessions Within	Request
.53659	44/82	/alumni/home.php
.53659	44/82	/alumni/
.50	41/82	/alumni/student/council.php
.50	41/82	/alumni/art/student_council_headshots/kbell
.50	41/82	/alumni/student/council.html
.4878	40/82	/alumni/student/index.php
.45122	37/82	/alumni/student/programs.html
.42683	35/82	/alumni/ring/index.html
.35366	29/82	/alumni/programs.html
.20732	17/82	/alumni/ring/men.html

Table 4.8: Profile of Old Alumni Cluster

Probability	Sessions Within	Request
.63542	61/96	/alumni/home.php
.58333	56/96	/alumni/
.58333	56/96	/alumni/programs.html
.57292	55/96	/alumni/regional.php
.52083	50/96	/alumni/benefits.html
.45833	44/96	/alumni/tours.php
.4375	42/96	/alumni/atlanta/index.php
.39583	38/96	/alumni/products.html
.38542	37/96	/alumni/womenofuga.php
.375	36/96	/alumni/tours-current.html

Generating Longest Common Subsequences

To strengthen the information provided by the profiles and the graphical representations of the models, we also generated a list of frequently occurring *longest common subsequences (LCS)*, the longest list of sequential requests common to all sessions within a set.

Lists of LCSs were generated for each of the four clusters generated, two by the procedure defined in 4.3.1 and two selected based on their high average ratio of sensitivity over

specificity from the clusters generated by the procedure defined in 4.3.1.

The following two lists are the two most common LCSs found from the sessions contained within the new alumni data set. We note here again that the group classified as new alumni exhibits great interest in the student council, leading us to believe that we captured the behavior of current students as opposed to students who have recently graduated. We also note the similarity between the two LCSs where the users were looking at the programs and council information before moving on to the products page.

1. /alumni/student/council.html
2. /alumni/student/council.php
3. /alumni/art/student_council_headshots/kbell
4. /alumni/student/programs.html
5. /alumni/student/council.html
6. /alumni/student/council.php
7. /alumni/products.html

1. /alumni/
2. /alumni/home.php
3. /alumni/programs.html
4. /alumni/student/index.php
5. /alumni/student/council.html
6. /alumni/student/council.php
7. /alumni/art/student_council_headshots/kbell
8. /alumni/products.html

The following lists are the three most common LCSs found from the sessions contained within the old alumni data set. We note here the interest in the tours, the dates, and the products provided by the Division of External Affairs.

1. /alumni/invite/breakfast/cowart_new.html
2. /alumni/toursfaq.html
3. /alumni/toursfaq.php
4. /alumni/tours-resources.html
5. /alumni/

1. /alumni/
2. /alumni/home.php
3. /alumni/programs.html
4. /alumni/products.html

1. /alumni/home.php
2. /alumni/home.html
3. /alumni/Tours.html
4. /alumni/Album.html
5. /alumni/WhatsNew.html
6. /alumni/Calendar.html
7. /alumni/Regional.html

Three of the LCSs generated from the sessions contained within the first generated cluster can be viewed below. We see here that the behavior of the users found within this group indicates an interest in the Division of External Affair's programs, as well as current nominations and student activity.

1. /alumni/home.php
2. /alumni/programs.html
3. /alumni/regional.php

1. /alumni//bulldog100/
2. /alumni//bulldog100/nominations.html
3. /bulldog100/nominations/add

1. /alumni/home.php
2. /alumni/atlanta/index.php
3. /alumni/atlanta/programs.html
4. /alumni/student/

The lists below are two of the LCSs generated from the sessions contained within the second generated cluster. We notice here that the behavior of the users found within this group indicates an interest in the women of UGA and men's rings.

1. /alumni/womenofuga.php
2. /alumni/womenoverview.html
3. /alumni/programs.html
4. /alumni/tours.php

1. /alumni/home.php
2. /alumni/ring/index.html
3. /alumni/ring/men.html

Report

For the report presented to the Division of External Affair's point of contact, we created graphical representations of the four groups created using the techniques described in the previous sections. An example of one of the visualizations can be seen in figure 4.5.

We also presented the profiles, the lists of LCSs, and additional observations such as distinct areas of interest within the site within the report. The observations, results, and conclusion can be found in the following sections.

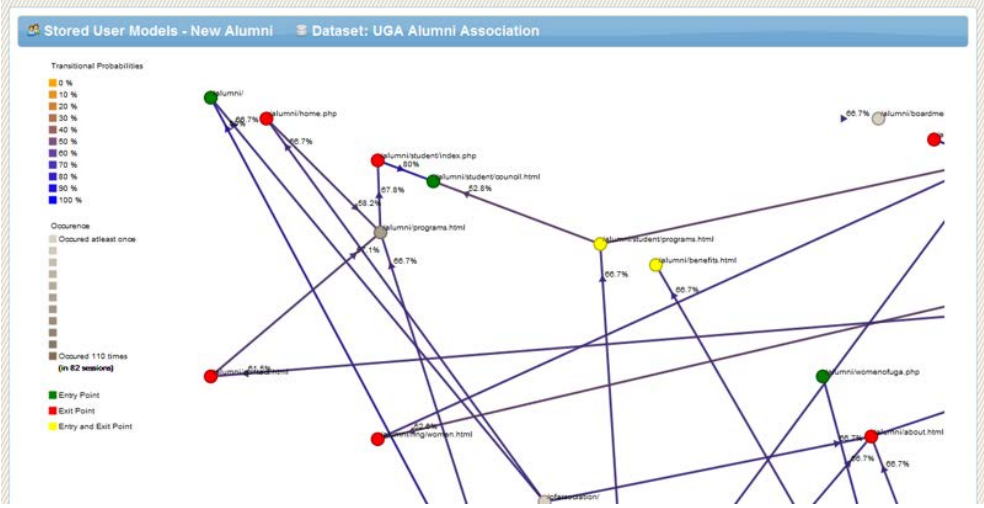

Figure 4.5: Graphical Representation of alumni.uga.edu Visitors

Additional Observations

Additional observations were made by analyzing the graphical representations of the clusters generated using the automated and manual techniques described above.

Our first observation was that user sessions contained within the first automatically generated cluster showed interest in reunions and volunteer work through the Division of External affairs. The traversal path that led to this observation is listed below.

1. alumni.php
2. ea/index.php
3. alumni/products.html
4. alumni/nominations.html
5. alumni/reunions.html
6. alumni/volunteer.html
7. alumni/womenofuga.php
8. alumni/enews
9. alumni/authors.htm

Our second observation was that the user sessions contained within the second automatically generated cluster showed interst in regional programs and events. There were also two large hubs in the traversal patterns of the sessions found within the new alumni group. These two traversal paths are listed below.

1. alumni.php
2. alumni/tours.html
3. ea/index.php
4. alumni/regional.html
5. alumni/womenofuga-supporters.html
6. alumni/atlanta/programs.php
7. alumni/wishlist/08wishlist.html
8. alumni/eventsalbum2005.html
9. alumni/mosainspeech.html

1. alumni/about.html
2. alumni/rign/index.html

Another interesting path found by analyzing the visualization created from the new alumni group shows high transition probabilities between the pages listed below.

1. alumni/home.php
2. alumni/programs.html
3. alumni/student/index.php
4. alumni/studentcouncil.php
5. alumni/art/student-council-headshots.html

We also note that comparing the visualizations created for both the new and old alumni leads us to believe that recent alumni are traversing very far into the website, while old alumni seem to be going deep within the site's structure.

The old alumni group also shows behavior indicative of users who are interested in the e-news articles linked to the newsletters sent out by the Division of External Affairs on a regular basis. A list of pages of interest, where the page has several inbound and outbound transitions with high probabilities, can be viewed below.

1. alumni/e-news2008.html
2. alumni/apsec/welcome.html
3. alumni/apsec/entertain.html
4. alumni/regional2.html
5. alumni/tours.html

4.3.2 Conclusion

The University of Georgia Division of External Affairs was able to infer useful information from the report. Specifically, the report, as well as the approach taken to classify types of users in order to analyze their online behavior, allowed the evaluators to begin questioning the influence of their newsletters. They were satisfied with the results, and requested that further analyses be performed to better understand the behavior exhibited by visitors when the newsletter served as the entry point for their visit.

Chapter 5

Usage Data Modeling Techniques

5.1 Customer Behavior Model Graphs (CBMGs)

Customer Behavior Model Graphs (CBMGs) are user models similar to Data Flow Diagrams (DFD), with a few key differences. DFDs show the flow of data through an information system, while CBMGs show the flow of requests through a web site. DFDs are capable of labeling the edges as generic actions, but the edges of CBMGs are reserved strictly for transitions between the nodes representing the pages within a web site. CBMG edges are labeled with additional information, such as a textual representation of the transition probability from one node to the next and visual cues such as color or thickness to reflect the transition probability.

CBMGs can be created from visitor sessions where the session requests are represented as nodes and the transitions from one request to the next are represented as edges marked with the transition frequency from one request to the next. CBMGs can be created for a single session or a group of sessions clustered by characteristics such as ISP or geographical location.

CBMGs allow evaluators to visualize the transition frequencies of groups of sessions and assist in better understanding the behavior of visitors by analyzing the behavior exhibited by their visits. However, CBMGs become cumbersome and non intuitive when displaying the transition frequencies and nodes for a large number of sessions.

Our implementation of the Customer Behavior Model Graph Algorithm was coded using Java 1.6 and utilized a relational database containing the information from a hierarchically structured sessionized access log. The algorithm takes in sets of information, such as browser names, IP addresses, and dates. The algorithm then extracts the sessions fitting the profile described by the sets of information. For example, if the algorithm is provided with the

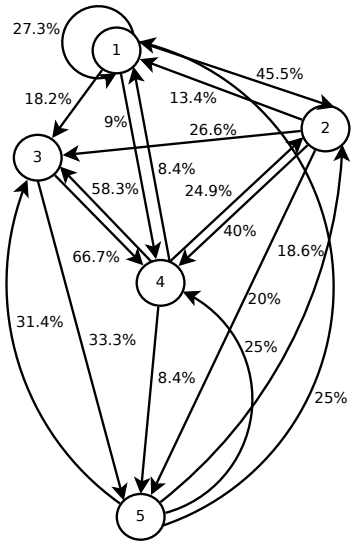

Figure 5.1: Customer Behavior Model Graph

following three sets of information, 1) (Firefox,Opera) 2)(192.168.2.3,192.168.2.7,192.168.3.6) and 3) (2009-06-01, 2009-07-01) then it will extract all sessions created by visitors from these three IP addresses when using the Firefox or Opera browsers between June 1st, 2009 and July 1st 2009.

Given this subset of sessions, the algorithm parses each session individually, storing each request pair as an edge and each request as a node. The remainder of the sessions are parsed and the stored nodes and edges are iteratively updated. Once all the sessions have been parsed, the edges are annotated based on the fan out of the originating node. In other words, if a node has three edges to other nodes, the number of transitions along each edge will be divided by the total number of transitions from that node, and this percentage, or transition probability, will be placed on the edge as a label. This is ultimately visualized as a graph containing nodes and edges, in which the nodes contain a particular request and the edges are labeled with a transition probability.

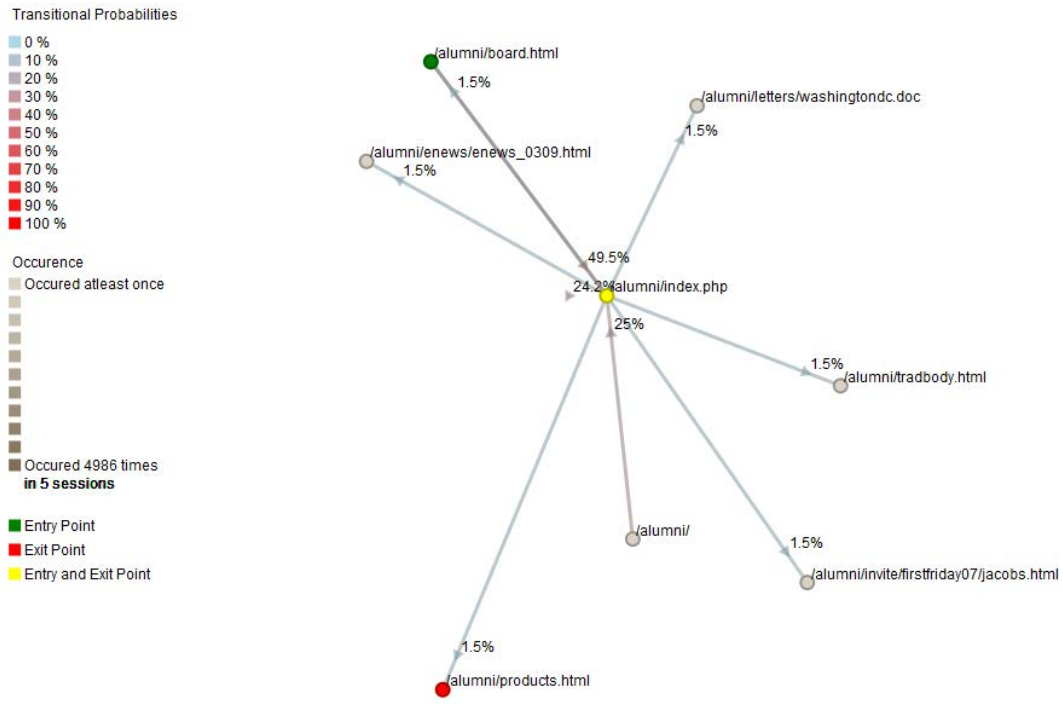

Figure 5.2: CBMG Created By Protovis

To create the graphical representation of the CBMGs we used Graphviz (Graph Visualization Software) [39], an open source graph visualization software package that supports automatic graph drawing of structured data. CBMGs are employed in our first case study, detailed in section 4.2.3. An example of a CBMG created by Graphviz can be seen in figure 5.1.

We later replaced Graphviz due to the inability to graph large numbers of nodes and incorporated Protovis [16]. However, we still rely on Graphviz to generate the coordinates for the nodes and pass the coordinates to Protovis to display in an interactive fashion. An example of a CBMG created by Protovis can be seen in figure 5.2

5.1.1 Computing Transitional Probabilities

A CBMG is created first by tallying the transitions of a user or group of users from one page to the next. Table 5.1 shows the tallied transitions between five pages.

Table 5.1: Transition Counts for User Group

-	1	2	3	4	5
1	3	5	2	1	0
2	2	0	4	6	3
3	0	0	0	8	4
4	1	3	7	0	1
5	3	4	5	4	0

Next, a transition frequency matrix is created by computing the probability of the users within the group to transition from one page to the next. This computation consists of tallying the total number of transitions from one page to its respective out pages. For example, the first row of Table 5.2 shows the probability of the users to transition from page one to page one, page one to page two, page one to page three, and so on.

Table 5.2: Transition Frequency Matrix

-	1	2	3	4	5
1	27.3%	45.5%	18.2%	9%	0%
2	13.3%	0%	26.7%	40%	20%
3	0%	0%	0%	66.7%	33.3%
4	8.3%	25%	58.3%	0%	8.3%
5	18.8%	25%	31.3%	25%	0%

5.1.2 Integrating Productions and Goals into CBMGs

CBMGs are comprised of sessions created by a visitor or visitors. These sessions often contain similar *episodes*, a subset of sequential requests within a session, that can be interpreted as *transactions*, a series of related events that constitute a visitor action, also explained in section 2.4. For example, a visitor purchasing an item from a web store will proceed to the checkout page, enter their personal information, then submit payment. While these three actions are logged as three distinct requests, they can also be identified as the episode indicating a sale. Cognitive user models have a similar notion of transactions called *productions*,

where a series of sequential actions constitute a production. These productions lead to *goals*, actions indicating the achievement of a task, which are similar to the last occurrence of a transaction within a session. We integrated productions and goals into CBMGs to create Production/Goal-CBMGs (PG-CBMGs). PG-CBMGs show:

- The transition frequencies of visitors as they traverse through the web site
- The productions involved with reaching a given goal
- The average staying time per page found with a mapped production

PG-CBMGs can be used as a model for future studies when there are modifications or additions to a user interface. For example, if modifications are made to an interface and PG-CBMGs exist for each known visitor type, evaluators can determine how each visitor type may attempt to reach certain goals.

To fully understand how this hybrid model works, assume that the CBMG shown before in section 5.1 depicts these five pages and five nodes. Next, assume that a production has been mapped to the transition from page one to page four, and the goal is achieved by transitioning from page four to page five. Figure 5.3 allows us to visualize this example.

If we were to create a CBMG that included all pages, transition frequencies, average staying times, productions, and goals for a given web application, we would have a predictive model based on user behavior derived from access logs. In doing so we would create a predictive user model that could be applied to any web application. Creating this model would allow evaluators to answer questions such as:

- Does the visitor's behavior include the necessary productions to reach a goal?
- Is the average staying time significantly higher for some productions?
- Are there productions that can be merged or productions that should be divided?

Having a model to directly answer such questions may help developers further understand their users and create more intuitive user interfaces. Furthermore, this model could also be

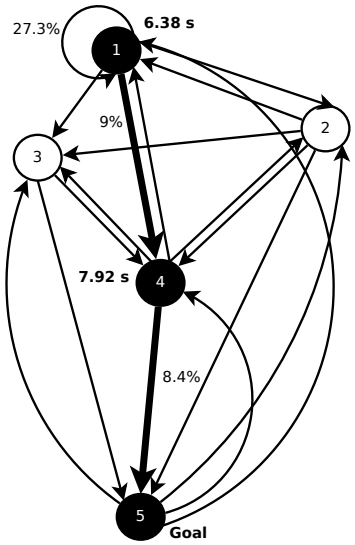

Figure 5.3: Production/Goal CBMG

used to better quantitatively define the effectiveness of a user interface.

However, productions and goals are terms found in the realm of cognitive user modeling. While Sengupta's notion of transaction-oriented views [104] were created for the purposes of workload characterization, transactions and productions are logically identical. Furthermore, an *Episode*, a sequence of requests found within a visitor session, can be logically mapped to the notion of a transaction. Since productions can be interpreted as transactions and transactions can be mapped to episodes, we can assume that certain subsequences of requests found within a visitor session is a transaction that can be interpreted as a production. In doing so, we begin to view visitor sessions as a series of productions employed to reach a specific goal as opposed to a general series of requests. By viewing the visitor session as a series of productions we believe evaluators are able to infer more meaningful information from analyses.

Given this information as a starting point, we developed the idea of a Time Series Analysis - Selected Episode Graph (TSA-SEG), described in section 5.2, by mapping the notion of a production to a transaction and allowing evaluators to annotate pages or transactions as goals.

5.2 Time Series Analysis - Selected Episode Graphs (TSA-SEGs)

Time Series Analysis-Selected Episode Graphs (TSA-SEGs) are graphical representations of a time series analysis of the changing transition frequencies between episodes, which allow the evaluators to easily assess the impact of the site changes based on the evolution of behavior for each group of sessions.

The Time Series Analysis - Selected Episode Graph (TSA-SEG) algorithm is an extension to the Customer Behavior Model Graphs described in section 5.1 and was written in Java 1.6 and Graphviz.

Used specifically for our research in [108], the TSA-SEG algorithm required visitor sessions and episodes as input. Certain episodes were manually identified as transactions by the evaluators and then applied to CBMGs created from the visitor sessions. By applying these episodes to the CBMGs we evolve the CBMG into a *SEG*, or selected episode graph. For example, if pages A, B and C comprised an episode titled 'Seeking Assistance', and one of the CBMGs contained the transitions A to B and B to C, the A, B and C nodes for that CBMG, as well as their respective inbound and outbound transition probabilities, would be merged into a single node titled 'Seeking Assistance'. By identifying the transactions in a CBMG and replacing all nodes comprising the transaction with a single node, the complexity of the graphical representation of usage behavior is significantly decreased while also creating a more informative diagram.

5.2.1 Experiment: Determining Effectiveness of TSA-SEGs

The database CryptoDB (http://cryptoDB.org) is a community bioinformatics resource for the apicomplexan parasite *Cryptosporidium*. Resources available through CryptoDB.org include whole genome sequences, annotations, expressed sequence tags and genome survey sequence data, as well as bioinformatic analysis and data-mining tools. More specifically, the site provides access to over 50 types of queries arranged in 18 categories, and returns 8 types of records. In addition, users may download data through a sequence retrieval tool, view sequence data and annotations through a Genome Browser, and use the BLAST sequence similarity search tool. A query history facility allows users to keep track of, combine, and compare the results of queries that the user has issued.

CryptoDB.org is under continual evaluation and revision based on data analysis, user feedback, and information derived from annual, in-depth usability studies. For this case study we observe one month of requests to CryptoDB.org, which represent the last two weeks of version 3.5 and the first two weeks of version 3.6. We believe that while a month of data does not produce a sufficient number of sessions to effectively measure all the changes in user behavior for all types of users, it does offer enough information to capture the most distinct changes in the three types of users analyzed in this case study. The primary difference between the two versions is the placement of the query grid, as seen in Figure 5.4, which is a table containing links to all of the site's question pages organized into 18 specific categories. Question pages allow users to issue queries to the web site's databases, such as the gene type question page that identifies genes based on gene type. While the query grid was available in version 3.5, the user had to navigate to the "Queries and Tools" section to access it. In version 3.6 the query grid was also placed to the front page offering immediate visibility for all users visiting CryptoDB.org.

One question the administrators were interested in answering was whether moving the Query Grid to the homepage aided the users in further utilizing important functionality of

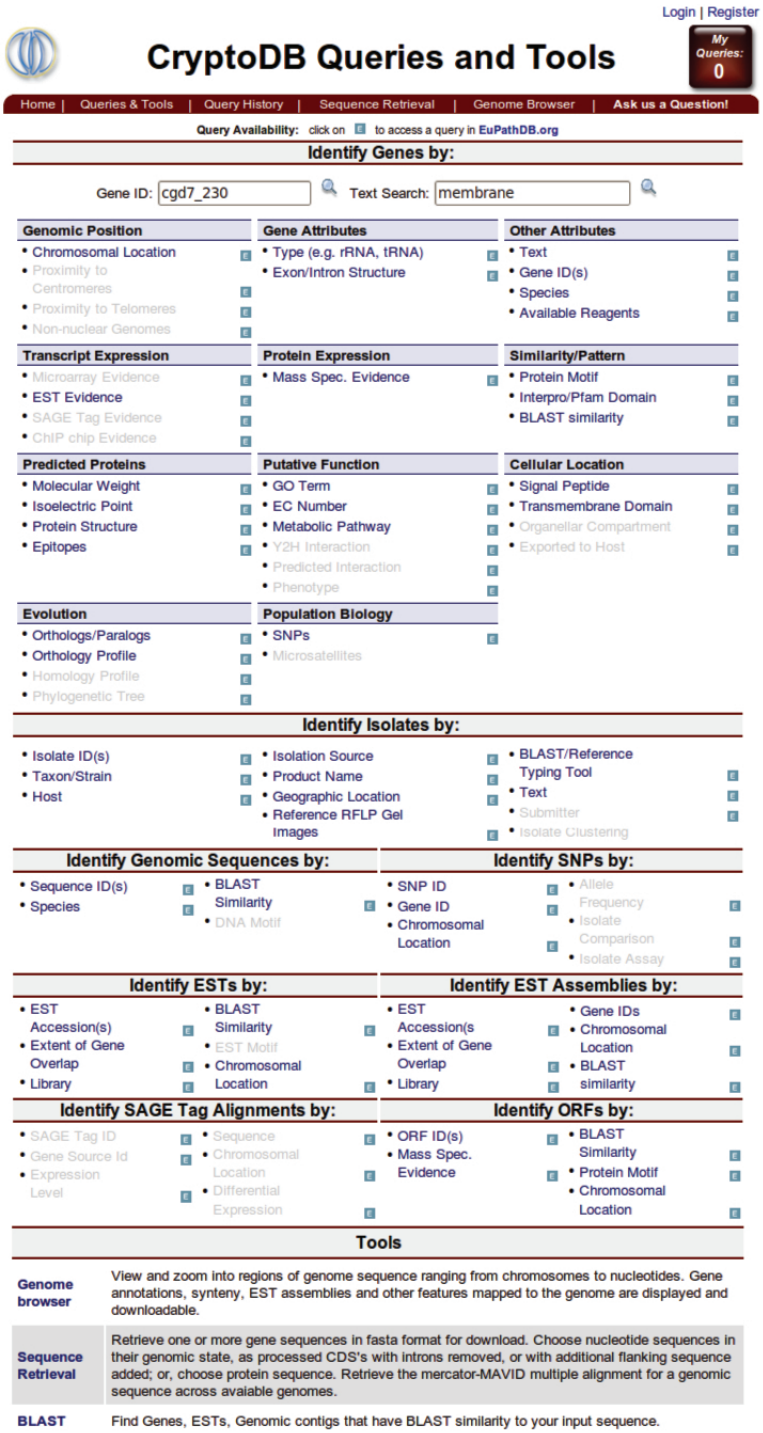

Figure 5.4: CryptoDB query grid

the site. The TSA-SEGs helped to answer this question.

We evaluated a data set collected from the CryptoDB.org site over a period of one month and chose the time windows to be two weeks before and two weeks after a site revision in which the "query grid" was placed on the site's front page. These time windows were selected in order to characterize the behavior of users (novices, intermediate, and advanced) before the revision and after the revision, thus enabling an evaluation of the impact of the revision on user behavior.

To permit evaluation and comparison of the clustering algorithms, we asked three domain experts (site administrators/developers for CryptoDB.org) to manually classify the data set into three distinct groups representing sessions created by novice users, intermediate users, and advanced users respectively. We began with 2261 sessions and then removed any session containing fewer than 3 requests to eliminate one-click sessions (an automatic redirect built into the site creates two unique requests when only one exists), or more than 50 requests, to filter out sessions consisting of redundant requests most likely created by bots. This process yielded a set of 450 sessions. Three site administrators/developers independently reviewed logs of time stamps and requests representing the 450 sessions and classified the sessions as novice, intermediate, or advanced. These characterizations were based on the number and variety of advanced features employed by the user when accessing the site. Of the 450 sessions, all three administrators agreed on 271 classifications, and two out of three agreed on 161 classifications. A majority vote was used to assign a classification. For 14 sessions, all three administrators disagreed; these sessions were eliminated from further analysis. Of the remaining 436, 348 were classified as novice, 80 as intermediate, and 8 as advanced.

Next, we compared the effectiveness of profiles versus TSA-SEGs in isolating changes in user behavior as a result of the site revision. A profile is a list of characterized URLs found within the sessions for each generated cluster. Each profile contained the top 25 most frequently occurring URLs for that cluster. The goal of the profiles is to allow the

evaluators to gain insight into the request preferences for each type of user before and after a revision. Twelve profiles were generated for the clusters created by Nasraoui's Hierarchical Unsupervised Niche Clustering algorithm (H-UNC), six for the "before" session clusters and six for the "after" session clusters. Similarly, six profiles were generated for the clusters created by our Semi-supervised Competitive Fuzzy Clustering algorithm (SCFC), three for the "before" clusters and three for the "after" clusters.

The evaluators were provided with both sets of profiles in a blind study and asked to make observations on their relevance for evaluating the site revisions. In analyzing the SCFC clusters, the evaluators were able to identify the novice, intermediate and advanced clusters. They were able to determine that novice users preferred searching for genes by gene record and locus tag, two of several options readily available to every user. Further, the profiles generated from the three clusters showed different ways of searching for genes, and observing this allowed the evaluators to gauge the difficulty of each method based on which profile contained which method.

The clusters created by H-UNC offered relatively little insight for this purpose. Only one of the profiles could be easily typed by the evaluators, and that typing did not allow for any contextually meaningful observations. The obvious advantage of SCFC over unsupervised methods is that the clusters are already typed.

Although the evaluators were not able to assign a classification to the H-UNC clusters, after further investigation we were able to determine that the profile generated for the pre-revision sessions in the fourth H-UNC cluster shared similarities with the profile of the pre-revision sessions of the novice SCFC cluster. The same held true for the profile generated for the sixth pre-revision H-UNC profile and the advanced pre-revision SCFC profile. Earlier in our analysis we described how we categorized the H-UNC clusters based on the existence of a predominant session type. We find it interesting that although the profiles appeared contextually meaningless at first, there is some observable overlap between the clusters cre-

ated by SCFC and H-UNC. Considering their methodological similarities, however, this is not entirely unexpected.

Finally, we created a Time Series Analysis Selected-Episode Graph (TSA-SEG). To create a TSA-SEG, we asked the evaluators to select episodes of importance to the overall usage of the site. This information, combined with the previously selected time windows and the information derived from the three clusters generated by the SCFC algorithm, allows us to create the TSA-SEG, a graphical representation of the changing transition frequencies in episodes of interest, for each type of user. The TSA-SEG for the episodes of interest for the cryptodb.org revision can be seen in figure 5.5.

The evaluators were provided with the TSA-SEG in figure 5.5. Each edge is labeled with "before" and "after" transition frequencies for each group. For example, the transition frequencies between the Home Page and the Query Grid show that for Novice users (N), the transition frequency was .085 (8.5%) before the revision and .043 (4.3%) after the revision. In other words, 8.5% of the novice sessions before the revision, and 4.3% of the novice sessions after the revision contained a transition from the Home Page to the Query Grid. The graphs seen by the evaluators were further enhanced with color to indicate substantial increases (green) or decreases (red) in transition frequencies.

The evaluators strongly believed that the TSA-SEG helped to directly answer the questions posed while making the revision. For instance, the revision placing an instance of the Query Grid on the Home Page led the developers to expect a spike in the transition frequency from the Home Page to specific query pages for all user types. While this was true for advanced users, the revision did not seem to alter the transition probability between the home page and the old query grid page for intermediate users, and the transition probability between these two pages for novice users decreased by half. Other observations included an increase in the transition frequencies of advanced users for advanced tasks, such as downloading information from the result pages, which strongly indicates that the revision

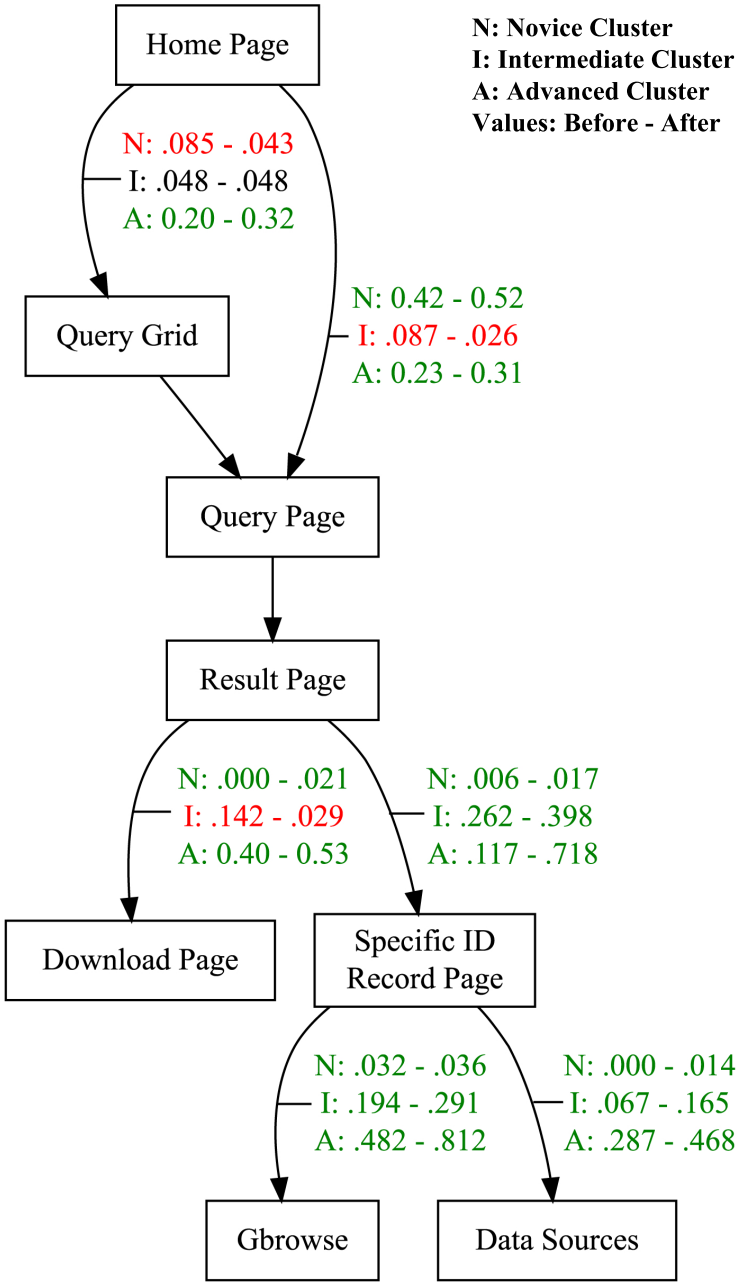

Figure 5.5: Time Series Analysis - Selected Episode Graph (TSA-SEG)

benefited the advanced users.

Chapter 6

Automating Supervised Methodologies

6.1 Web Analytics

Web analytics consists of collecting and analyzing internet data, such as the information found within web server access logs, in order to generate information for the purpose of understanding web usage and web user behavior. Web analytics is divided into two categories: off-site and on-site. Off-site analytics determines the web site's audience and popularity in relation to the internet as a whole, whereas on-site analytics consists of measurements that require the information generated by users once they are on your web site. Web server log file analysis, page tagging, and a hybrid method containing both are the three technological approaches to collecting data for on-site analytics. Log file analysis relies on the interpretation of information found within web server log files in which user requests for pages and information are stored, and page tagging uses JavaScript to notify a server housing the web site's usage information when information is requested by a web user. The hybrid approach simply combines these two approaches in order to capture previous, current, and future usage information. Our research in web analytics can best be described as the analysis of web server access logs in an off-site analytics approach to better understand user behavior.

6.2 Web Analytics Software

Web analytics software consists of packages, platforms, or applications that create reports or views of information of the data retrieved through on-site analytics approaches. A few of the key metrics used by analytics packages are hits, a user request for a file, page views, a user request for a page, and visits/sessions, a sequential series of page views requests by a single user. While analytics packages often provide a plethora of information that

may help site administrators better understand their usage data, current methodologies and current software frameworks lack the functionality and interactivity necessary to support administrators in answering certain classes of questions. For example, current analytics applications are capable of generating reports detailing referring sites that contribute heavily to a site's inbound traffic, but are unable to detail, dissect, or visualize the usage patterns of these users in any meaningful fashion.

6.3 Ajaxalytics - An Automated Framework for Usage Mining

We have addressed several of the limitations described in the rest of this thesis in the form of a framework that provides interpretations of information and levels of interaction that are required to answer questions more closely related to the site's usage: Ajaxalytics.

Ajaxalytics is an open source analytics engine that generates reports, visualizations, and interactive tables and graphs from web server access logs. The information generated by Ajaxalytics can be used by site administrators and developers to provide more insight into their users' behavior and activities. The software provides several traditional views of information derived from web server access logs such as:

1. The total number of user sessions and user requests grouped by hour, day, month and year.
2. Visitor information such as total number of sessions, requests for each session, user agent information (browser, operating system), and physical location
3. Statistics for the pages requested by the users such as the number of times the page has been accessed and the number of sessions the page is contained within.
4. Lists of the keywords that led the users from major search engines such as Google, Bing, and Yahoo to the web site

The software provides more complex views of the data as well, such as:

1. *Interest Ratio:* The page to session ratio for each individual page.
2. *Depth Ratio:* The page to session ratio for each individual page where pages comprising longer sessions with more requests than average are weighted more heavily.
3. *Return Rate:* The rate at which visitors created more than one session thus indicating their return to the web site.
4. *Click Through Rate:* The number of users who traversed past the home page divided by the number of times the web site's homepage was requested.

Ajaxalytics was also built to allow evaluators to utilize the data modeling techniques described in chapter 5. Specifically, the software contains modules that allow site administrators to group information from a sessionized access log using high level, meaningful concepts in order to create visualizations such as Customer Behavior Model Graphs, Time Series Analysis - Selected Episode Graphs, Path Analysis Diagrams, and cluster maps.

Ajaxalytics was built using the programming languages Java and PHP, the relational database MySQL using the InnoDB engine, the JavaScript frameworks Jquery and JqueryUI, the JavaScript graphing toolkits Flot and Protovis, the web server Apache, and HTML and CSS. The database is an instance of the usage information hierarchy described in section 3.9.

The first step of the procedure is to install the necessary software. To run the software we created, the evaluator must first install MySQL, Apache, PHP, and Java. Once the software dependencies are installed, the user places our software in Apache's web directory and navigates to the Ajaxalytics install page where they provide the application with the database information and the location of the web server access log, the administrator username and password to be used by the evaluator, and the location and format of the access log.

Once the evaluator submits the form, the software will create the database dynamically, then create a hierarchically structured sessionized access log from the server access log provided using the VASH technique. Multiple views are also created to reduce latency when

attempting to query temporary storage created from complex table joins.

An important aspect of the evolution of the software framework is the need to create and provide a package that an average computer user would know how to install and run. We have spent a great deal of time making sure that once the software is in place the user can begin analyzing their web site's usage data in a matter of minutes, regardless of the size of the log or the complexity of the site.

Ajaxalytics currently consists of several thousand files and tens of thousands of lines of code, which does not make it feasible to provide the entirety of the source code in this thesis. However, the source code can be viewed by downloading the package from http://ajaxalytics.com.

6.3.1 Supervised Modeling Techniques

The model creator included in Ajaxalytics is a module that allows evaluators to isolate usage data based on the users':

1. IP address
2. user agent information such as browser type and version, and operating system type and version.
3. physical location such as country, state, and city.
4. session creation time based on the session's initial request.
5. requests and subsequent page comprising the request.
6. referrer, or site from which the user traversed.

The module also provides ways to group requests based on the page name as well as to rename groups, pages, and parameters.

The ability to isolate usage data based on these various aspects of usage information are integrated into the framework as rules, and multiple rules can be added by the evaluator in

order to create highly specific views of the information.

Once the rules have been created and any additional grouping or renaming conditions have been provided, the evaluator can create either an inclusive or exclusive model graph. An inclusive model graph is comprised of the sessions that meet the requirements of the rules previously given by the evaluator. An exclusive model graph is comprised of the sessions that failed to meet the evaluator's requirements.

The model graph is a visualization depicting the pages as nodes and the transitions from page to page as edges labeled with the transition probability derived by analyzing the subset of sessions. Several visual cues, such as the darkness of the node indicating a higher request count for the page it represents, the darkness of the edge indicating a higher transition probability between pages, light red circles around nodes that are entry points, and arrows indicating the logical flow of requests, have been programmed into each model graph as well. The graph is interactive, containing functionality that allows the evaluator to remove nodes and edges from the graph, rearrange the nodes within the graph, and zoom in to sections of the graph. An example model graph being used in the context of the application can be seen in figure 6.1

As an example of how this module might be used, let us assume that an evaluator installs our software, provides it with the information necessary to retrieve, sessionize, and store an access log, logs in, and begins using model creator. Suppose the evaluator owns a local venue and needed to determine the effectiveness of a recent marketing campaign run in his city. Using the model creator module, they creates a rule to isolate sessions based on the session host's originating city then selects their city, and a rule to isolate sessions between the first day and last day of the marketing campaign. They then create an inclusive model graph that is dynamically generated and returned to the evaluator. The evaluator removes the pages (nodes) that do not pertain to the marketing campaign and begins analyzing the graph and making observations. They then notice an irregularity with the transition probability from

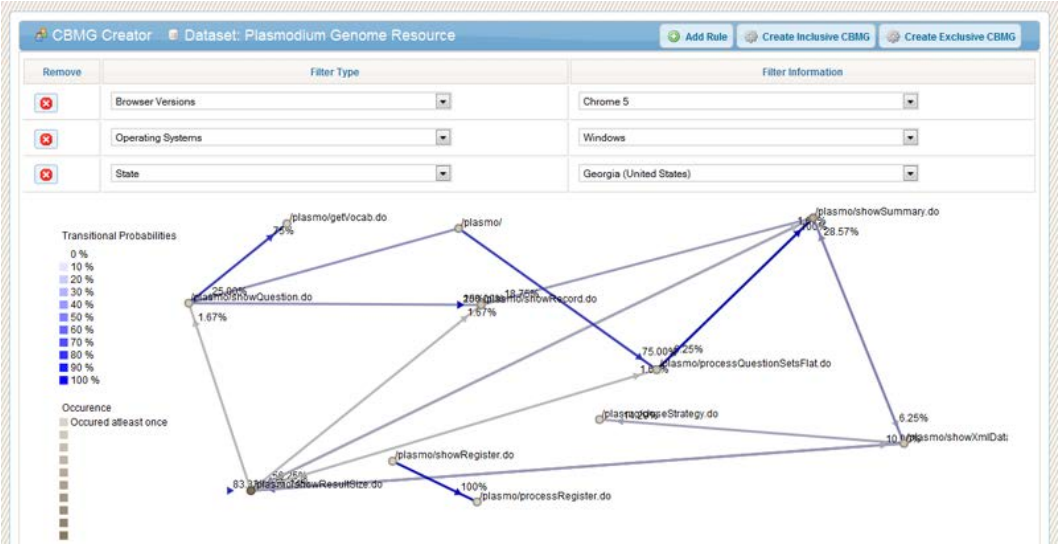

Figure 6.1: Interactive Model Graph

two very important pages, the web store's checkout page and the web store's payment page. To investigate this issue, they create another rule isolating visitors by browser type in order to focus their attention on Firefox users, and then create another inclusive model graph. They observe that the transition probability between the two pages in question is noticeably higher than it was before, then replace the rule to isolate Firefox users with a rule to isolate Internet Explorer users instead and create another inclusive model graph. They may then observe that no edge exists between these two pages then form an assumption based on the existing evidence and his own site-specific knowledge.

Ajaxalytics allows evaluators to create interactive visualizations depicting the requests, average staying time, transitions, and transition probabilities between requests for sets of sessions grouped by IP, user agent, location, referrer information, and existence of requests. Evaluators are able to refine the visualizations further by isolating sessions by date and selecting a threshold for transition probabilities, which removes any requests that do not contain a transition from or to another node with a transition probability above the given threshold. Evaluators are provided with the ability to dynamically create the visualizations, adjust the position of the nodes and edges, and adjust the magnification. By providing this

technique we allow evaluators to create visualizations depicting particular data sets that can then be examined to help answer questions or provide insight into the users' behavior. For example, an evaluator could create a customer behavior model graph using sessions created by users in the surrounding area that contain requests for particular pages of interest made within the previous month, then set the minimum threshold for transition probability to 75% to isolate their attention on the stronger connections made by users within this given set of sessions.

6.3.2 Experiment: Determining Effectiveness of Ajaxalytics

The case study conducted for PlasmoDB allows administrators and evaluators to use Ajaxalytics, our software framework, with a 15 month server access log created by PlasmoDB.org as the sessionized access log used for the primary data source. The case study involves both formative evaluations of the software framework and analysis of the results generated by the software framework from the administrators' and developers' perspective. The formative evaluations serve to provide feedback to help strengthen the software framework, while the analysis of results serve to provide evidence that the information and visualizations created and presented by the applications are useful on some measurable level.

In order to perform our first formative evaluation, we prepared as follows:

1. Sessionize a 15 month portion of the PlasmoDB server access log.
2. Create a workspace for Ajaxalytics using the sessionized access log.
3. Present Ajaxalytics to PlasmoDB evaluators.
4. Conduct formative evaluations to improve the software framework.
    (a) Provide the Customer Behavior Model Graph generator to the evaluators to determine if they are able to derive useful information from application's presentation of results

The case study required the sessionization of a large portion of the PlasmoDB server access log. The case study then required that the sessionized access log be connected to the software framework to be viewed directly through the application. The evaluators then used the framework to view information and generate models in order to extract useful or insightful information about PlasmoDB.org's usage behavior.

The guided portion of our methodology consists of attempting to answer the administrators' and developers' questions with regards to PlasmoDB.org using the behavior model generator.

Procedures

1. The PlasmoDB access log was obtained by acquiring /var/log/httpd/mnt/plasmodb.org/access_log_cl
   20101010.gz from the evaluators
2. A portion, dating from June 24th, 2009, to October 8th, 2010, was extracted from the access log. The extracted portion was then sessionized into 207,180 distinct sessions, comprised of 13,278 distinct pages, 6,771,936 distinct requests, and 34,169 distinct visitors. For the purposes of the case study, no further modifications were made to the data set and all of the information was provided to the administrators and developers through the software framework's interface.
3. Formative evaluations and sessions allowing the administrators and developers to use the software were then conducted in order to determine whether they could use our software framework to answer questions concerning the usage of PlasmoDB.
4. The model generator, serving as a module within the software framework, is used to help answer questions posed by the evaluators.

Technical Challenges

Several challenges were presented during sessionization, information extraction, and module creation that required dramatic modifications to the code base.

To begin with, the size of the access log, 2.8 GB (2867 MB), produces an abnormally large amount of usage information and requires a large amount of resources to be translated into a hierarchically structured sessionized access log. Sessionizing a server access log this large required the software framework's log sessionizer to be partially rewritten in order to complete the sessionization process given limited physical resources, such as 64-256MB of available memory, within a feasible amount of time. The previous version of the software framework's log sessionizer required 7 hours to sessionize the access log used for this case study, and after modifications were made the access log could be parsed in 20 minutes. The primary issue with the log sessionizer consisted of the JVM's garbage collector running too often and too long, and this issue occurred due to basic String concatenation within Java 1.6, which creates a new String object for every instance of concatenation thus overloading the JVM with instantiated objects and ultimately the garbage collector with nullified objects.

There were also issues concerning the latency of several of our complex queries, where the queries were LEFT JOINing several tables each consisting of hundreds of thousands of rows and, for the instance of the "requests" table, almost seven million rows. Even after indexing the tables and altering the MySQL configuration file in order to allocate more resources to the database server, the queries were still exhibiting 30 second to 5 minute latencies for returning information. To solve this issue we created materialized views of the sessionized access log that required additional programs to be run after the log was initially sessionized, but provided the information to the application through specific tables tailored to the view of each module within the application, decreasing the latency by providing a simple, one table look up per module.

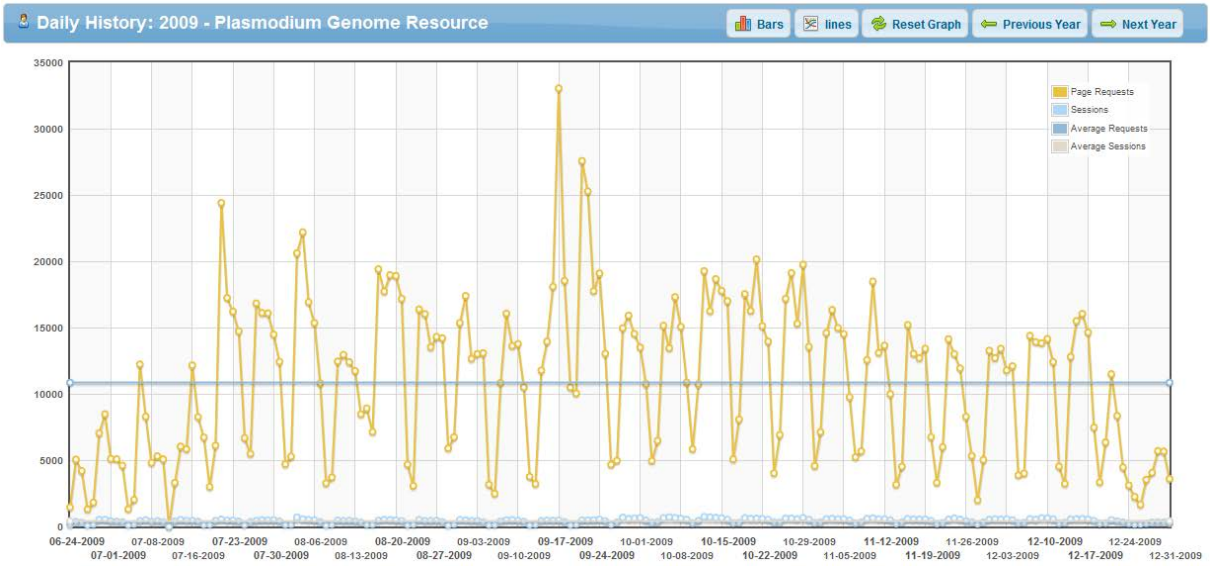

Figure 6.2: PlasmoDB.org Daily History 2009

Results

An evaluator used our software framework to determine the reason behind a sharp and sudden increase in web traffic between the end of December, 2009, and the beginning of January, 2010.

To begin with, the evaluator analyzed the software framework's daily history view, which offers a graphical representation of the number of requests and sessions per day for any given year, seen in figures 6.2 and 6.3.

This analysis helped determine that the increase in traffic began specifically on January 6th, 2010. Using this information, the evaluator analyzed the software framework's daily history details, which offers a textual breakdown of each request and each request's occurrence for each available day found within the sessionized access log. In doing so, the evaluator discovered that one specific request, */cgi-bin/dataplotter.pl*, showed a significant increase in its daily requests. The daily breakdown indicated that there were 5259 requests for dataplotter.pl, well over twice as many requests as the second most requested resource, fives times as many as the third, and seven times as many as the fourth. The request counts

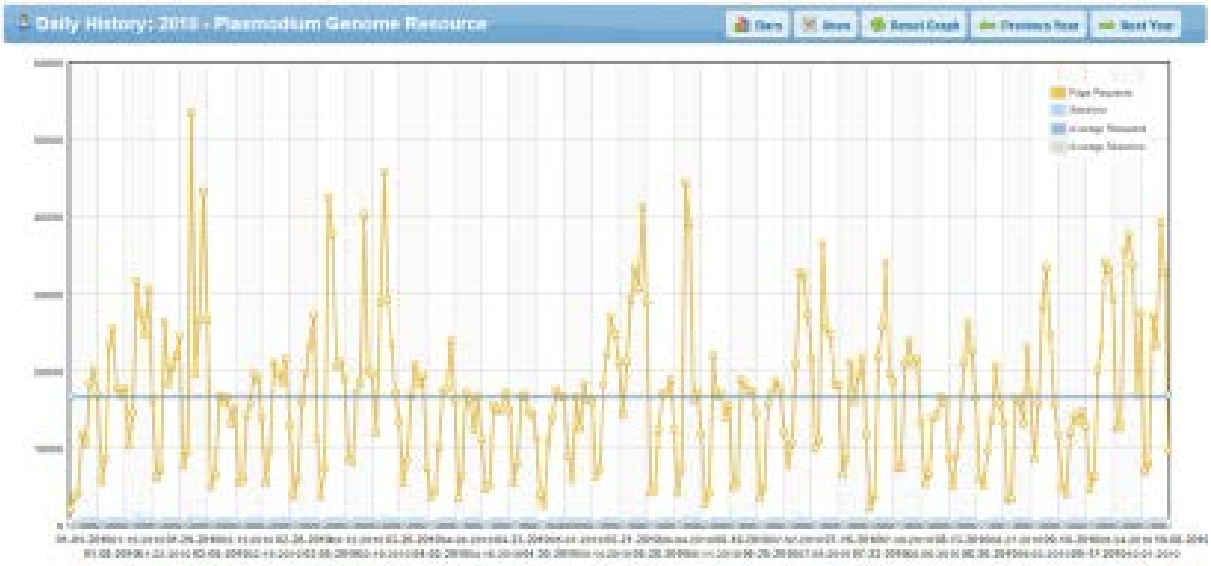

Figure 6.3: PlasmoDB.org Daily History 2010

for dataplotter.pl from December 9th, 2010 to January 10th, 2010 is shown in figure 6.4.

To determine the cause of the increase for this particular resource, the evaluator created a Customer Behavior Model graph using the software framework's CBMG generator and limited the view by creating a rule specifying that only the transitions of visitors traversing away or traversing towards the request should be included in the creation of the CBMG. The evaluator then limited the view further by creating another rule to restrict the sessions included in the CBMG to those created between the dates of January 6th, 2010 and January 31st, 2010. The generated CBMG, shown in figure 6.5, allowed the evaluator to determine that the increase in requests for dataplotter.pl were originating from several gene record pages. Specifically, the nodes labeled with URLs containing */cgi-bin/dataPlotter.pl* were not made explicitly by the visitor, but rather by the request depicted as the top left node. While it appears that the visitors' were requesting several distinct pages, the gene record pages were merely sending remote procedure calls to the dataPlotter.pl scripts thus logging each call as an individual request.

When examining the requests automatically created and sent by the web page each

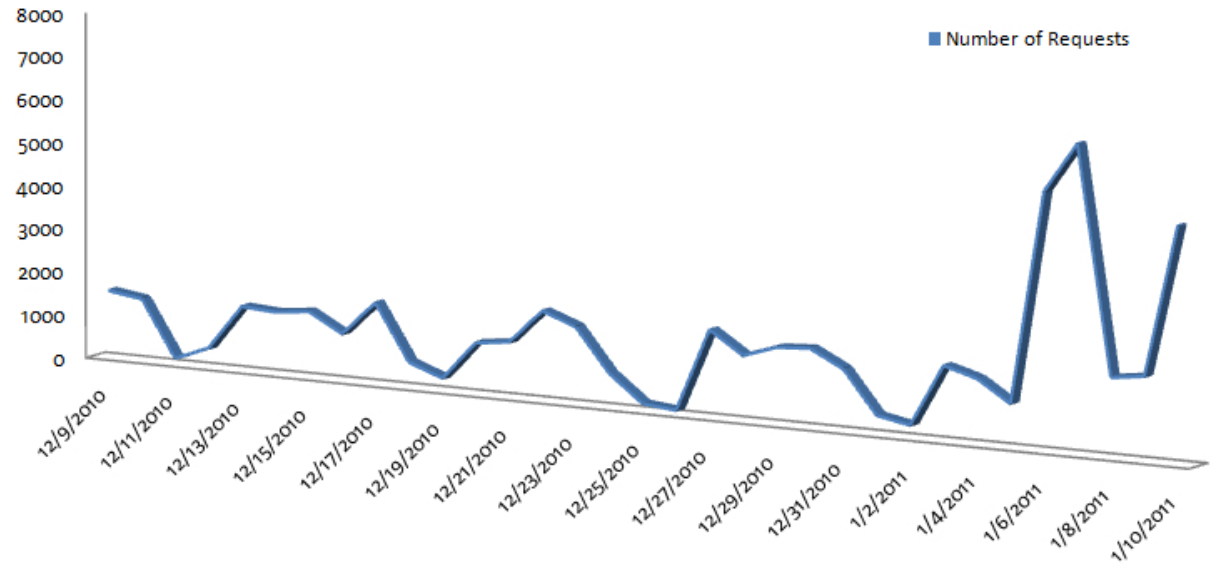

Figure 6.4: Dataplotter Daily Request Count, December 9th, 2010 to January 10th, 2011

time a gene record page is accessed, the evaluator noticed that six individual requests for dataplotter.pl were being sent to the server. Upon further examination, the evaluator realized that the module responsible for the trigger to request dataplotter.pl six separate times was released with PlasmoDB version 6.3, a version of PlasmoDB.org that was released December 22nd, 2009. Given this information, the evaluators were able to locate the rogue code responsible for the high number of requests for dataplotter.pl and resolve the issue.

6.3.3 Formative Evaluations

Evolving Usage Modeling Visualizations

In an attempt to increase the visibility of potentially significant information, to offset the signal-to-noise ratio, we enhanced the customer behavior model graphs by color coding the transitions based on their probability and increasing or decreasing the size of nodes based on their frequency of occurrence. These ideas came from analyzing basic graphical representations. When a state is larger, it implies some numerical representation is greater than that of a smaller state. If something is blue, or cold, it represents a lower transition probability.

Figure 6.5: Customer Behavior Model Graph, Limited View

Vice versa for red, or hot.

Sengupta introduced the concept of transaction-oriented views (TOVs) [103] and we thought applying TOVs might help minimize the state-space. We manually generated customer behavior model graphs after grouping users based on their behavior, selecting series of requests which could be meaningfully categorized as transactions, and determining two distinct periods of time for a comparative evaluation. The evaluators were able to infer meaningful information from the observations made against the generated TSA-SEG; our research and results related to these representations are found in [108].

Upon further investigation we determined that evaluators required the ability to create TSA-SEGs dynamically, allowing for rule sets and graphs to be created and refined in real time at the evaluators' own pace. We then theorized about the real advantage of our approach and determined the genuine strength of the approach was its ability to allow evaluators to choose what information was used when creating the TSA-SEG. The starting point of our formative evaluations is the release of the first version of Ajaxalytics, shown in figure 6.7

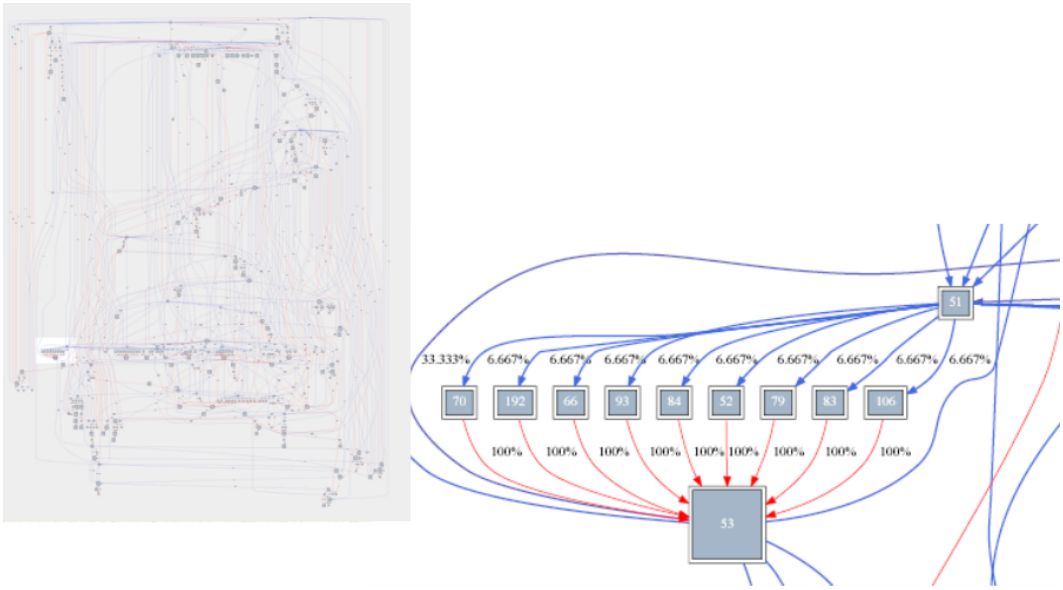

Figure 6.6: ECBMG created from ApiDB usage data

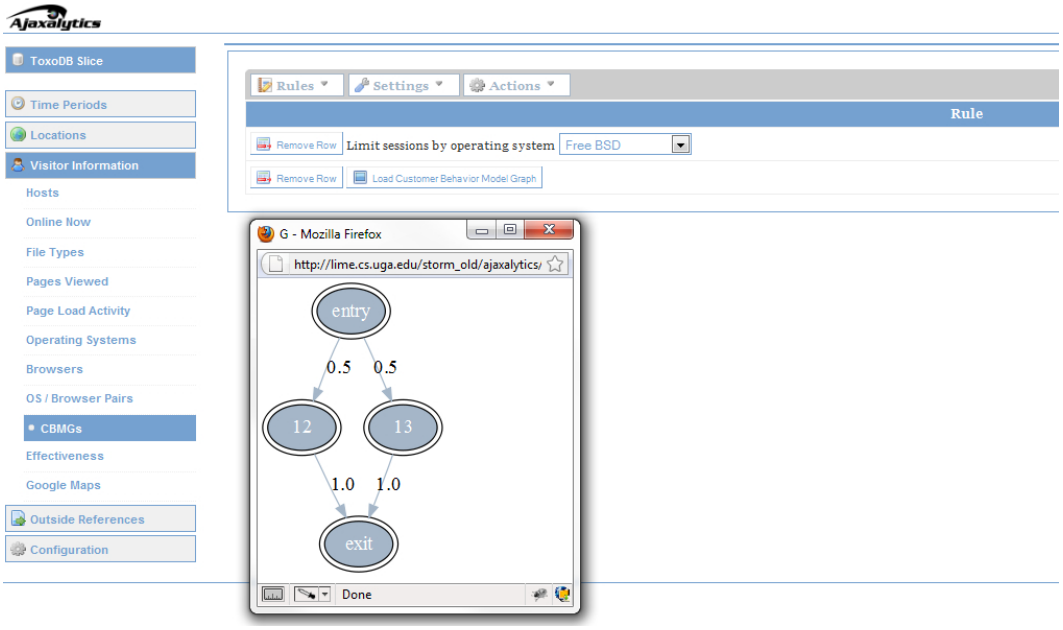

Figure 6.7: Ajaxalytics, Version 1

First Formative Evaluation

During our meeting on October 19th, 2010, we introduced the application to the evaluators and received the following suggestions to improve the effectiveness and usefulness of the application:

- The application should provide the ability to drill down into dates to retrieve their session and session request information.
- The user model graphing module should provide evaluators with the ability to group several pages into a single node.
- The user model graphing module should provide evaluators with the ability to temporarily rename pages, sets of pages, and parameters.
- The graph module should have a rule to isolate users by IP address.
- The graph should be labeled with the information used to generate the graph itself.
- The graphing module should provide evaluators with the ability to export a generated graph as an image to save and/or share.
- The graphing module should provide evaluators with the ability to play through sets of sessions.

Since sessions are delimited by time stamps, and the requests are ordered by their occurrence within the log file, we integrated the time spent by users' on any given page into the sessionization process as well as the models and visualizations to better determine implicit and explicit requests.

We were also able to determine that the evaluators required the ability to group users based on their behavior in contextually meaningful ways. Doing so required us to integrate

additional supervised processes, such as the *evolving breadcrumb*, an interactive module that tracks the level of depth an evaluator has achieved, and the *interactive model graphs*, graphical representations of usage information that allows evaluators to zoom in, zoom out, and shift the nodes within the graph.

Second Formative Evaluation

With the changes outlined by the first formative evaluation in place, we requested for the evaluators to use Ajaxalytics with a sessionized PlasmoDB web server access log in an attempt to answer some of their questions regarding the web site's recent UI modifications and strategy component release.

We obtained, cleaned, parsed, and stored a 2.3GB web server access log with sessions ranging from June 26th, 2009 to October 6th, 2010. The instance of Ajaxalytics presented to the evaluators for this iteration of the formative evaluation directly interfaced this data set.

During our meeting on November 14th, 2010, we introduced version 1.1 to the evaluators and received the following suggestions to improve the effectiveness and usefulness of the application:

- The model graphs should suppress requests that are not indicative of user behavior.
- The application should install easily and parse logs efficiently.

In order to allow for evaluators who did not possess the technical expertise necessary to install and use our software framework, we created simplistic install processes and additional interfaces that automate the installation of the framework and the processes necessary to translate a server access log into a information usage hierarchy within a relational database. The interfaces allow for the dynamic creation and control of several information usage hierarchies, each representing its own server access log.

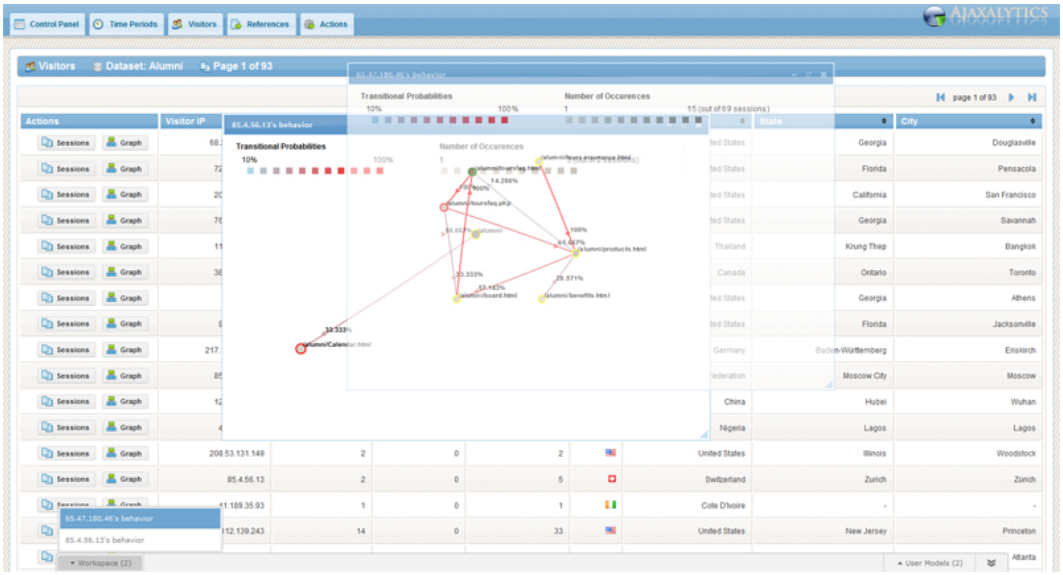

Figure 6.8: Ajaxalytics, Version 1.2

The suggestion to improve the model graphs by suppressing particular requests led to the research responsible for creating the behavioral heuristics found in section 3.6. These heuristics were then integrated into the framework. The suggestions provided by the evaluators for the second formative evaluation led to the release of version 1.2 of our framework, shown in figure 6.8

Chapter 7

Conclusion and Future Work

Our research has uncovered key opportunities in methodologies for web analytics. Specifically, we make contributions in data preprocessing, data integration, bot detection, sessionization, and user modeling techniques. We have detailed procedures that more efficiently parse usage information from server access logs, efficiently store significantly large, hierarchically structured sessionized access logs into relational databases, more accurately distinguish human visitors from bots, and effectively distinguish explicit user requests from implicit user requests in order to create sessions that better represent the visitor's actual usage patterns. Through our case studies, and performance and formative evaluations, we have shown that our contributions have furthered the advances of preprocessing techniques by creating more efficient, more accurate procedures.

The contributions of our research are again summarized as follows:

- Bot detection algorithms that more accurately distinguish web bots from human users through the use of localized behavioral heuristics
- Session refinement algorithms that better depict usage patterns by identifying and suppressing implicit requests through the use of localized behavioral heuristics
- Web Server Access Log parsers that more efficiently clean, parse, and store hierarchically structured sessionized access logs
- Semi-Supervised clustering algorithms capable of grouping sessions based on the behavioral patterns exhibited by users within manually selected session representatives
- Interactive selection algorithms that allow evaluators to group users together for analyses based on primary characteristics, such as their location, browser, or operating system, as well as secondary characteristics, such as their usage patterns
- User modeling and visualization techniques that more accurately depict the behavior

of a user, or groups of users

- A software framework that automates these processes and allows evaluators to perform complex analyses on their own server access logs

The data preprocessing techniques described in chapter 3 allowed us to create programmatic implementations of our sessionization algorithms that efficiently parse web server access logs as hierarchically structured sessionized access logs. We then increased the accuracy of sessions within sessionized access log by applying localized behavioral heuristics described in section 3.6, which more accurately distinguish between requests made by machines and those made by human users.

We created behavioral heuristics that more accurately depict sessions by identifying explicit and implicit requests, described in section 3.7, and suppress implicit requests when creating visualizations or views comprised of usage information.

Our evaluations of the accuracy of our behavioral heuristics described in sections 3.6.1 and 3.7.1 and our evaluations of the efficiency of our preprocessing and data integration techniques described in sections 3.8.1 and 3.9.1 show that we have created a more efficient and more effective approach for web usage mining.

The unsupervised and semi-supervised classification techniques described in chapter 4 allowed evaluators to seed semi-supervised clustering algorithms with sessions indicative of particular user types. Then usage modeling techniques described in chapter 5, when integrated into our framework, provided evaluators with the ability to guide the process of grouping users based on a combination of several different, manually selected characteristics, then visualize the behavioral patterns of these groups through graphical representations of the subset of usage information to provide answers that existing frameworks and methodologies could not.

Our software framework in chapter 6 automated these processes in a way that allowed evaluators to utilize these techniques and contributions in practice. Through formative

evaluations, we have shown that our framework has strengthened and become more useful over time.

For our future work, we intend to extend our user modeling techniques further to include the notion of a *Stepwise Comparative Model Graph (SCMG)*, iterative, interactive graphical representations of the pages, transitions and transition probabilities between requests found within multiple groups of sessions. Once modularized, we intend to provide the next version of our framework to the evaluators for further formative evaluations.

We also intend to perform a case study, outlined in section 7.2, that would analyze the hierarchically structured sessionized access log to help evaluators better determine the effectiveness of the web site's tutorials. In doing so, we believe that our methodologies may help evaluators better determine the qualities of effective tutorials.

7.1 Stepwise Comparative Model Graphs (SCMGs)

Stepwise Comparative Model Graphs (SCMG) are interactive visualizations allowing evaluators to view the aggregate set of actions existing in the session group it represents for any given step. For example, if evaluators obtained three groups of sessions known to have been created by novice, intermediate and advanced users, then viewed the information as a Stepwise Comparative Model Graph, they would be presented with a visualization similar to the one found in figure 7.1. The evaluators could then interact with the visualization by stepping forward and backward through the actions existing in each session group, where the first step would display the unionized set of entry pages for all sessions within the group.

The technique for generating Stepwise Comparative Model Graphs described in section 7.1.1 combines various techniques found in the areas of content mining and user evolution modeling, and uses the notion of productions and goals employed by cognitive architectures when creating and using cognitive models. Specifically, the technique allows evaluators to isolate information by time, group users based on their patterns of behavior, then select productions from a list of dynamically generated episodes found within the set of sessions. The technique then creates interactive visualizations that provide both a generalized overview

Figure 7.1: Stepwise Comparative Model Graph Example

of group-specific usage information through graphs similar to CBMGs and functionality allowing evaluators to iterate stepwise through each groups' site traversal independently.

7.1.1 Generating Stepwise Comparative Model Graph

The Stepwise Comparative Model Graph generation technique, shown in figure 7.2, shares several similarities with the profile generation and behavioral clustering technique, such as manually selected group representatives, clustering using the SCFC algorithm, and generating profiles from the clusters. However, profile generation and behavioral clustering technique requires that the clustering and profile generation be generated manually once the representative sessions are selected, and allows for only one representative per group.

The Stepwise Comparative Model Graph generation technique provides evaluators with the ability to isolate sessions based on the date and time they were created, then presents Customer Behavior Model Graphs (CBMGs) of visitors found within the specified time period and allows for multiple representatives to be selected for each user archetype of interest. The framework allows the evaluators to then select goal states, or pages of interest, and sessions that do not contain any goal states are removed from the set. The set of sessions is then analyzed and the most common subsequences of requests found within the sessions are returned to the evaluator. The evaluator is then provided with the ability to name

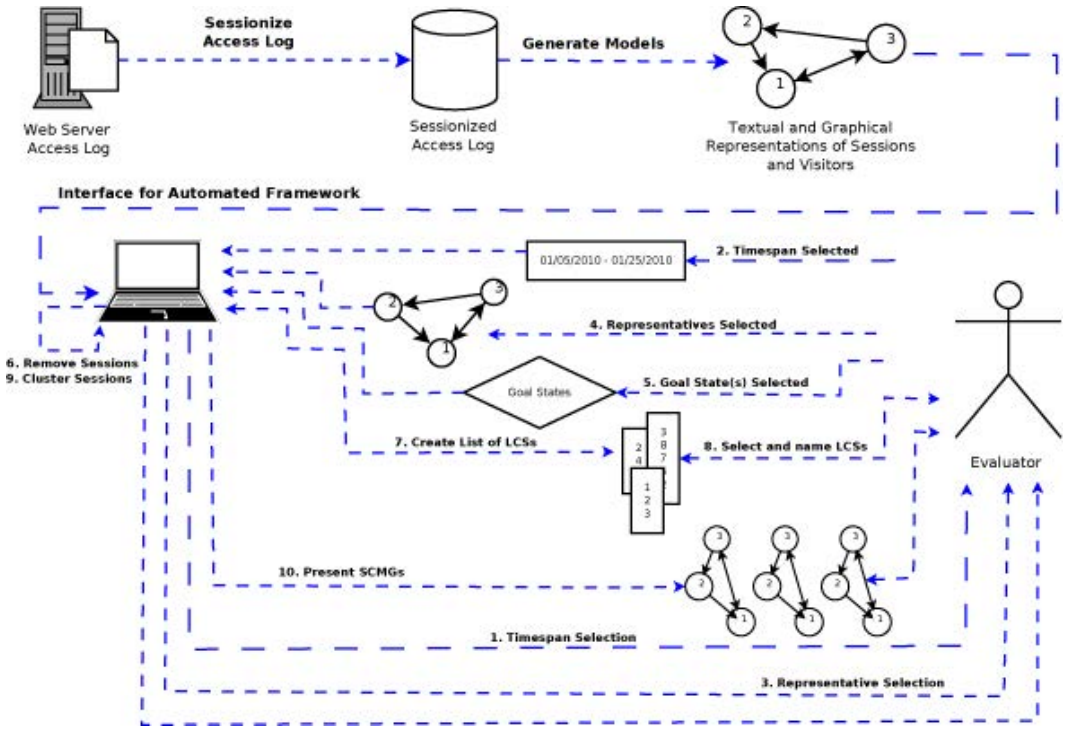

Figure 7.2: Generating Stepwise Comparative Model Graph

and save any and all available subsequences as productions. The framework merges the evaluator selected CBMGs into a single model per group, then seeds the SCFC algorithm with the merged models. The automated framework then creates Stepwise Comparative Model Graphs (SCMGs) of the clusters generated by SCFC, and the SCMGs are modified to present the previously selected productions and goals.

The Stepwise Comparative Model Graph generation technique provides a dynamic environment that allows evaluators to select user archetypes, cluster based on the usage behavior exhibited by the selected archetypes, specify goals, select productions based on a list of common subsequences of requests, and analyze the subsequent visualization.

7.2 Proposed Case Study: CryptoDB - Effectiveness of Tutorials

For this proposed case study, we wish to analyze how web users interact with the strategies workspace found within CrytpoDB.org. CryptoDB integrates whole genome sequence and

annotation along with experimental data and environmental isolate sequences provided by community researchers. The database includes supplemental bioinformatics analyses and a web interface for data-mining. CryptoDB can be considered a fully functioning web application that interfaces massive data stores to assist web users in querying against and discovering information. The strategies workspace is a module within the web application that allows web users to manipulate, share, and save complex stratagem. Strategies consist of steps where the web user is allowed to unionize, intersect, or subtract results from multiple queries in order to target specific sets of information to use within their research.

One of the issues the developers for CryptoDB have is determining whether or not the application is useful to the web users. That is, the developers do not have the means to determine whether using the strategies workspace allowed the web users to obtain their results, or reach their goals, more effectively, efficiently, and with few errors. While the EuPathDB group, the organization charged with developing and maintaining CryptoDB, holds yearly workshops that allow them to ask the web users questions directly, there is no clear way to determine the effectiveness of the strategies workspace with audit trails, or access logs, alone.

Another issue the developers have is determining whether or not the novice web users progress to intermediate and advanced users. They also face the issue of properly typing users based solely on the users' navigational patterns and behavior. In the past, researchers have provided sessionized access logs to the developers of CrytpoDB to have them manually type small sets of user sessions based on their behavior. With these annotated data sets, clustering algorithms were able to group the remaining sessions fairly accurately and derive results that allowed the developers to answer particular questions directly. One of the questions the developers posed was whether or not a recent revision to the web application increased the transition frequencies of three different types of users, novice, intermediate, and advanced, of certain paths within the web application. The assumption being that if the transition

frequency increased, more users were traversing the paths the developers wanted them to take, and thus the revision could be considered effective.

There are several issues with this approach, but the most notable is that effectiveness was determined based on what the developers thought the users should be doing. In educating oneself in the area of cognitive user models, it is easy to see that effectiveness should be defined based on the users' ability to reach their goals efficiently, and with as little error as possible. It is also easy to see that the evolution of the user should also happen quickly and easily. In hindsight, it seems commonsensical to define effectiveness based on the users' ability to reach their goals, and the users' ability to progress and evolve quickly. With the notion of productions and goals, we could easily redefine user types based on the productions found within user behavior as users attempt to achieve their goals as well as the errors involved when attempting to reach the goals themselves.

The proposal involves two separate studies. The first is to determine the effectiveness of tutorials and derive productions and errors involved with reaching a goal, and the second is to create a cognitive user model by typing the users and then creating CBMGs and TSA-SEGs from the sessionized access log. The CBMG Predictive User Model can be used to make predictions of how well a user type would do on a different system.

For the first part of the study, we will have two sets of users who are familiar with the domain, biology students preferably, to perform actions within the strategies workspace to meet a particular goal. This will be a think-aloud study where the web users will explain what they are thinking at every step while we capture video and audio of the procedure. The first set of users will perform a task to reach a desired goal, while the second set of users will watch a tutorial on how to use the strategies workspace before attempting to reach a desired goal. We will analyze the audio and video, as well as the session information, from each study to capture the staying time and path traversals of each user from each group. We will also annotate the sessions by assigning production names to the episodes which they correspond.

In other words, if we determined that production A consists of traversing from page X to page Y to page Z, we can annotate this episode as production A if found within a session. This is contingent upon the fact that each production will have a unique subpath. If, however, separate productions share the same subpath then we will have to rethink our approach for mapping productions to subpaths. We will then analyze the tutorial to determine which productions are shown to the viewer explicitly. Next, we will analyze the sessions created by the users to extract average staying time per page. Given this information we will then compare the sessions created by the first group to the sessions created by the second group to determine if watching the tutorial helped the second set of users reach the goal. The effectiveness of a tutorial could thus be measured by time, how much more quickly a user can reach a goal after watching a tutorial, by error rate, the percentage decrease in error rate after watching a tutorial, and degree of production existence, the number of productions shown in the tutorial. The proposed equations for determining the effectiveness of a tutorial are shown below.

avg_{gr_a,go_x} = average time spent by all users in group a to reach a goal x

$error_{gr_a,go_x}$ = average error rate by all users in group a to reach goal x

$prod_{gr_a,go_x}$ = average number of productions by all users in group a to reach goal x

$E_{time} = avg_{gr_a,go_x}/avg_{gr_b,go_x}$ = the percentage increase or decrease of average time spent by all users within two groups to reach goal x.

$E_{error} = error_{gr_a,go_x}/error_{gr_b,go_x}$ = the percentage increase or decrease of average number of errors shown by all users within two groups while reaching goal x

$E_{prod} = prod_{gr_a,go_x}/prod_{gr_b,go_x}$ = the percentage increase or decrease of average number of productions shown by all users within two groups while reaching goal x.

$OE = E_{time} + E_{error} + E_{prod}/3$ = Overall Effectiveness of a tutorial. A result above 1.0 is more effective by a percentage $OE - 1.0$, and a result below 1.0 is less effective by a percentage $OE - 1.0$.

Next, we would intersect the list of productions generated by observing the two groups reaching a goal with the list of productions explicitly shown with the tutorial to generate a new list. This list of productions would be those used by the users to reach a goal that were not found within the tutorial. We could then determine the effectiveness of a tutorial and generate a list of productions that may be added to the tutorial to increase it's effectiveness.

The second part of our study is to type the users based on their behavior. For example, an advanced user may be one who utilized all the correct productions to reach the goal, whereas a novice user may be one who utilized several incorrect productions and did not reach the goal. This will have to be a supervised process where the site administrators determine what is novice behavior and what is advanced behavior. In supervising this process, we will generate distinct, typed groups of users. We will then merge the sessions of each group of users to form a PG-CBMG per group. The PG-CBMGs, as defined in the previous section, could be used to help the evaluators answer specific questions about the user interface before making significant changes. The PG-CBMGs could then be used to test the user interface once the changes are made.

First, we will evaluate our manual process by using information obtained through the yearly EuPathDB workshops and information provided to us by the administrators of EuPathDB.org to complete the first five steps outlined in figure 4.1. Given this information, we will create programs to complete the remainder of the manual process and gauge the quality

of the suggestions derived by the manual process based on the administrators' collective opinions. In doing so, we intend to strengthen our methodology by factoring these opinions into the automated framework before finalizing and using the application for the second part of our evaluation.

Second, we will briefly train the administrators to use the application surrounding the automated framework and they will be provided with the information necessary to complete steps two and four of the procedure outlined in section 7.1.1. We will then determine how closely related the suggestions generated by both processes are based again on the collective opinion of the administrators. We hope to show that we can determine the overall effectiveness of an application and provide suggestions to increase the effectiveness of an application with nothing more than usage data. However, if the results aren't comparable and time permits, we will use the administrators' collective opinions to further strengthen the process and loop through this part of the evaluation until the process generates viable results.

Bibliography

[1] Log files - apache http server. http://httpd.apache.org/ docs/2.0/logs.html.

[2] Log Files - Apache HTTP Server. http://httpd.apache.org/docs/2.0/logs.html.

[3] Web Characterization Terminology - Definitions Sheet. http://www.w3.org/1999/05/WCA-terms/.

[4] Web characterization terminology - definitions sheet. http://www.w3.org/1999/05/WCA-terms/.

[5] Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 3–10, New York, NY, USA, 2006. ACM.

[6] Robert St Amant and Thomas E. Horton. Model-based evaluation of cell phone menu interaction. In *Proceedings of the CHI04 Conference on Human Factors in Computer Systems*, pages 343–350. ACM, 2004.

[7] Apache. Apache commons collections framework. http://commons.apache.org/ collections/, 2011.

[8] Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. Webwatcher: A learning apprentice for the world wide web. pages 6–12. AAAI Press, 1995.

[9] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *Proceedings of 19th International Conference on Machine Learning (ICML)*, pages 19–26, July 2002.

[10] Sugato Basu and Raymond J. Mooney. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *In Proceedings of the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 42–49, 2003.

[11] Roberto J. Bayardo and Google Inc. Detecting near-duplicates for web crawling. In *Proceedings of the Sixteenth International World Wide Web Conference*.

[12] Helene Pigot Belkacem Chikhaoui. Evaluation of a contextual assistant interface using cognitive models. In *Proceedings of the 5th International Conference on Human Computer Interaction*, pages 36–43. ACM, 2008.

[13] Bettina Berendt. Detail and context in web usage mining: coarsening and visualizing sequences. In *WebKDD 2001 - Mining Web Log Data Across All Customer Touch Points*, pages 1–24. Springer-Verlag, 2002.

[14] Krishna Bharat and Monika R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 104–111, New York, NY, USA, 1998. ACM.

[15] Daniel Billsus and Michael J. Pazzani. A hybrid user model for news story classification. In *Proceedings of the Seventh International Conference on User Modeling*, 1999.

[16] Mike Bostock and Jeff Heer. Protovis: A graphical approach to visualization. Accessed Online, 2010. `http://vis.stanford.edu/protovis/`.

[17] Sergey Brin. Extracting patterns and relations from the world wide web. Technical Report 1999-65, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0119.

[18] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[19] Michael D. Byrne. ACT-r/PM and menu selection: applying a cognitive architecture to HCI. *International Journal of Human Computer Studies*, 55(1):41–84, 2001.

[20] Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Model-based clustering and visualization of navigation patterns on a web site. *Data Min. Knowl. Discov.*, 7:399–424, October 2003.

[21] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 307–318, New York, NY, USA, 1998. ACM.

[22] Soumen Chakrabarti, Byron E. Dom, David Gibson, Jon Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Mining the link structure of the world wide web. *IEEE COMPUTER*, 32:60–67, 1999.

[23] Ming-Syan Chen, Jong Soo Park, and Philip S. Yu. Efficient data mining for path traversal patterns. *IEEE Trans. on Knowl. and Data Eng.*, 10(2):209–221, 1998.

[24] Apache Commons. Uniform resource identifier (uri): Generic syntax. Accessed Online, 2010. `http://labs.apache.org/webarch/uri/rfc/rfc3986.html`.

[25] Joshue Connor. User testing: How to involve users in technical web development cycles as a natural evolution in the creation of inclusive technologies and accessible content.

In Klaus Miesenberger, Joachim Klaus, Wolfgang Zagler, and Arthur Karshmer, editors, *Computers Helping People with Special Needs*, volume 5105 of *Lecture Notes in Computer Science*, pages 258–263. Springer Berlin / Heidelberg, 2008.

[26] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1:5–32, 1999.

[27] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1:5–32, 1999.

[28] J. Cowie and W. Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, 1996.

[29] D. Crow and B. Smith. Db_habits: comparing minimal knowledge and knowledge based approaches to pattern recognition in the domain of user-computer interactions. *Neural Networks and Pattern Recognition in Human-Computer Interaction*, pages 39–61, 1992.

[30] Arindam Das and Wolfgang Stuerzlinger. A cognitive simulation model for novice text entry on cell phone keypads. In *Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!*, ECCE '07, pages 141–147, New York, NY, USA, 2007. ACM.

[31] Andrea de Lucia, Michele Risi, Genoveffa Tortora, and Giuseppe Scanniello. Clustering algorithms and latent semantic indexing to identify similar pages in web applications. In *WSE '07: Proceedings of the 2007 9th IEEE International Workshop on Web Site Evolution*, pages 65–72, Washington, DC, USA, 2007. IEEE Computer Society.

[32] Laurent Destailleur. Awstats - free log file analyzer for advanced statistics. Accessed Online, 2010. http://awstats.sourceforge.net/.

[33] Laurent Destailleur. Awstats - free log file analyzer for advanced statistics. Accessed Online, 2010. http://awstats.sourceforge.net.

[34] Frei Software Development. Webalizer xtended. Accessed Online, 2010. http://www.patrickfrei.ch/webalizer/.

[35] Thomas Draier and Patrick Gallinari. Characterizing sequences of user actions for access logs analysis. In *UM '01: Proceedings of the 8th International Conference on User Modeling 2001*, pages 228–230, London, UK, 2001. Springer-Verlag.

[36] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization. *ACM Trans. Internet Technol.*, 3(1):1–27, 2003.

[37] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization. *ACM Trans. Internet Technol.*, 3(1):1–27, 2003.

[38] Michael Eisen. Cluster analysis and visualization software. Accessed Online, 2010. http://rana.lbl.gov/eisen/?page_id=7.

[39] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz - Open Source Graph Drawing Tools. *Graph Drawing*, pages 483–484, 2001.

[40] Brian Lavoie et al. Web characterization terminology and definitions sheet. Accessed Online, 2010. http://www.w3.org/1999/05/WCA-terms/.

[41] Ronen Feldman and Ido Dagan. Knowledge discovery in textual databases (kdt). In *In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95*, pages 112–117. AAAI Press, 1995.

[42] Ronen Feldman, Moshe Fresko, Yakkov Kinar, Yehuda Lindell, Orly Liphstat, Martin Rajman, Yonatan Schler, and Oren Zamir. Text mining at the term level. In *PKDD '98: Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 65–73, London, UK, 1998. Springer-Verlag.

[43] Hichem Frigui and Raghu Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):450–465, 1999.

[44] Pallavi Chakraborty G. Premkumar, Alfred T. Ho. E-government evolution: an evaluation of local online services. *International Journal of Electronic Business*, 4(2):177–190, 2006.

[45] Chris Tilt Gene Lynch, Susan Palmiter. The max model: A standard web site user model. *5th Conference on Human Factors and the Web*, 1999.

[46] Google. Google analytics. Accessed Online, 2011. http://www.google.com/analytics/.

[47] Google. Guava: Google core libraries for java 1.5+. http://code.google.com/ p/guava-libraries/, 2011.

[48] Michael Wagner Heather A. Johnson. Analysis of web access logs for surveillance of influenza. *Medinfo*, 11:1202–1206, 2004.

[49] Brian Helfrich and James A. Landay. Quip: Quantitative user interface profiling. Unpublished manuscript, 1999.

[50] Sebastien Helie. The clarion cognitive architecture: A tutorial, 2006.

[51] Philippe Le Hgaret. W3c document object model, 2010. `http://www.w3.org/DOM/`.

[52] Jason I. Hong, Jeffrey Heer, Sarah Waterson, and James A. Landay.

[53] William H. Hsu, Joseph Lancaster, Martin S. R, and Paradesi Tim Weninger. Structural link analysis from user profiles and friends networks: A feature construction approach. 2008.

[54] Scott E. Hudson, Bonnie E. John, Keith Knudsen, and Michael D. Byrne. A tool for creating predictive performance models from user interface demonstrations. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 93–102, New York, NY, USA, 1999. ACM.

[55] S.; Fong S.; Hussain, T.; Asghar. A hierarchical cluster based preprocessing methodology for web usage mining. In *Advanced Information Management and Service (IMS)*, pages 472–477, Fac. of Eng. & Appl. Sci., Muhammad Ali Jinnah Univ. (MAJU), Islamabad, Pakistan, 2010.

[56] Google Inc. Publications by googlers, Nov 2010. `http://research.google.com/pubs/papers.html`.

[57] Melody Y. Ivory and Marti A Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4):470–516, 2001.

[58] Xin Jin, Yanzan Zhou, and Bamshad Mobasher. Task-Oriented Web User Modeling for Recommendation. volume 3538, pages 109–118. 2005.

[59] Bonnie E. John and David E. Kieras. The goms family of user interface analysis techniques: comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351, 1996.

[60] J.M. Jolion, P. Meer, and S. Bataouche. Robust clustering with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):791–802, 1991.

[61] Anupam Joshi, Karuna Joshi, and Raghu Krishnapuram. On mining web access logs. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 63–69, 2000.

[62] Henry Kautz, Bart Selman, and Mehul Shah. The hidden web. *AI Magazine*, 18:27–36, 1997.

[63] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[64] Aleksey Kolupaev and Juriy Ogijenko. Captchas: Humans vs. bots. *IEEE Security and Privacy*, 6:68–70, 2008.

[65] Raymond Kosala and Hendrik Blockeel. Web mining research: A survey. *SIGKDD Explorations*, 2:1–15, 2000.

[66] R. Krishnapuram and C. P. Freg. Fitting an unknown number of lines and planes to image data through compatible cluster merging. In *Pattern Recognition*, volume 25, pages 385–400, 1992.

[67] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. *Computer Networks*, 31(11-16):1481 – 1493, 1999.

[68] Christian Kurz, Helmut Hlavacs, and Gabriele Kotsis. Workload generation by modelling user behavior in an isp subnet. 2008.

[69] TAUSCHER L. and GREENBERG S. How people revisit web pages: empirical findings and implications for the design of history systems. *International Journal of Human-Computer Studies*, 47, 1997.

[70] Nicolas Labroche, Marie-Jeanne Lesot, and Lionel Yaffi. A new web usage mining and visualization tool. *Tools with Artificial Intelligence, IEEE International Conference on*, 1:321–328, 2007.

[71] Zhigang Li, Ming-Tan Sun, Margaret Dunham, and Yongqiao Xiao. Improving the web site's effectiveness by considering each page's temporal information. In *Advances in Web-Age Information Management*, volume 2762 of *Lecture Notes in Computer Science*, pages 47–54. Springer Berlin / Heidelberg, 2003.

[72] Alentum Software Ltd. Weblog expert - powerful log analyzer. Accessed Online, 2010. `http://www.weblogexpert.com/`.

[73] Sanjay Madria, Sourav Bhowmick, W. Ng, and E. Lim. Research issues in web data mining. In Mukesh Mohania and A Tjoa, editors, *DataWarehousing and Knowledge Discovery*, volume 1676 of *Lecture Notes in Computer Science*, pages 805–805. Springer Berlin / Heidelberg, 1999.

[74] L. Martin. Usability analysis and visualization of web 2.0 applications. In *10th International Symposium on Web Site Evolution*.

[75] Daniel A. Menascé, Virgilio A. F. Almeida, Rodrigo Fonseca, and Marco A. Mendes. A methodology for workload characterization of e-commerce sites. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 119–128, New York, NY, USA, 1999. ACM.

[76] Daniel A. Menasce and A. F. Almeida Virgilio. *Scaling for E Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[77] Daniel A. Menasce and A. F. Almeida Virgilio. *Scaling for E Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[78] Un Yong Nahm and Raymond J. Mooney. A mutually beneficial integration of data mining and information extraction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 627–632. AAAI Press, 2000.

[79] Olfa Nasraoui. Mining web access logs using relational competitive fuzzy clustering. In *Proceedings of the Eight International Fuzzy Systems Association World Congress*, 1999.

[80] Olfa Nasraoui. R.:an evolutionary approach to mining robust multiresolution web profiles and context sensitive url associations. *International Journal of Computational Intelligence and Applications*, 2:339–348, 2002.

[81] Olfa Nasraoui and Hichem Frigui. Extracting web user profiles using relational competitive fuzzy clustering. In *International Journal on Artificial Intelligence Tools*, volume 9, pages 509–526, 2000.

[82] David Nicholas, Paul Huntington, and Anthony Watkinson. Scholarly journal usage: the results of deep log analysis. *Journal of Documentation*, 61(2):248–280, 2005.

[83] Robert W. Proctor Niels A. Taatgen, Addie Johnson and Kim-Phuong L. Vu. *The Handbook of Human Factors in Web Design.*, chapter 25. 2004.

[84] Robert W. Proctor Niels A. Taatgen, Addie Johnson and Kim-Phuong L. Vu. Improving the web site's effectiveness by considering each page's temporal information. pages 47–54. Springer Berlin / Heidelberg, 2004.

[85] UGA Division of External Affairs. University of georgia - division of external affairs, Nov 2010. http://www.externalaffairs.uga.edu/ea/.

[86] Elizabeth Leon Olfa Nasraoui and Raghu Krishnapuram. Unsupervised niche clustering: Discovering an unknown number of clusters in noisy data sets. *Evolutionary Computation in Data Mining*, pages 157–188, 2005.

[87] Oracle. Innodb startup options and system variables. http://dev.mysql.com/doc/refman/5.5/en/innodb-parameters.html, 2011.

[88] Oracle. Mysql version 5.1 reference manual. http://dev.mysql.com/doc/ refman/5.1/en/, 2011.

[89] Helena Ahonen Oskari, Helena Ahonen, Oskari Heinonen, Mika Klemettinen, and A. Inkeri Verkamo. Applying data mining techniques for descriptive phrase extraction in digital document collections. In *In Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries*, pages 2–11, 1998.

[90] InnoBase Oy. Innodb. http://www.innodb.com/, 2011.

[91] Alfred T. Ho Pallavi Chakraborty G. Premkumar. E-government evolution: an evaluation of local online services. *International Journal of Electronic Business*, 4:177–190, 2006.

[92] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Hua Zhu. Mining access patterns efficiently from web logs. In *PADKK '00: Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, pages 396–407, London, UK, 2000. Springer-Verlag.

[93] Nicolas Poggi, Toni Moreno, Josep Berral, Ricard Gavald, and Jordi Torres. Web customer modeling for automated session prioritization on high traffic sites. In Cristina

Conati, Kathleen McCoy, and Georgios Paliouras, editors, *User Modeling 2007*, volume 4511 of *Lecture Notes in Computer Science*, pages 450–454. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-73078-1_63.

[94] Matthias Rauterberg. Amme: an automatic mental model evaluation to analyse user behaviour traced in a finite, discrete state space. *Ergonomics*, Nov 1993.

[95] Frank E. Ritter, Frank E. Ritter, Andrew R. Freed, Andrew R. Freed, Onida L. Haskett, and Onida L. Haskett. Discovering user information needs: The case of univeristy department web sites.

[96] van Rooy D. St. Amant R. Ritter, F. E. and K. Simpson. Providing user models direct access to interfaces: An exploratory study of a simple interface with implications for hri and hci. In *IEEE Transactions on Systems, Man, and Cybernetics: Part A, Systems and Humans*, volume 36, pages 592–601, 2006.

[97] Salvatore Sanfilippo. Visitors, a fast web log analyzer. Accessed Online, 2010. `http://www.hping.org/visitors/`.

[98] Salvatore Sanfilippo. Visitors, a fast web log analyzer. Accessed Online, 2010. http://www.hping.org/visitors.

[99] Luas Sarmento, Alexander Kehlenbeck, Euganio Oliveira, and Lyle Ungar. Efficient clustering of web-derived data sets. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 5632 of *Lecture Notes in Computer Science*, pages 398–412. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-03070-3_30.

[100] G. Scanniello, D. Distante, and M. Risi. Using semantic clustering to enhance the navigation structure of web sites. In *10th International Symposium on Website Evolution*.

[101] Alan Schwartz. Markov model. Accessed Online, 2010. http://araw.mede.uic.edu/~alansz/tools/markov.html.

[102] Ahmed Seffah, Mohammad Donyaee, Rex Kline, and Harkirat Padda. Usability measurement and metrics: A consolidated model. *Software Quality Journal*, 14:159–178, 2006.

[103] Shubhashis Sengupta. Characterizing web workloads - a transaction-oriented view. In Samir Das and Sajal Das, editors, *Distributed Computing - IWDC 2003*, volume 2918 of *Lecture Notes in Computer Science*, pages 833–833. Springer Berlin / Heidelberg, 2003.

[104] Shubhashis Sengupta. Characterizing web workloads - a transaction-oriented view. In Samir Das and Sajal Das, editors, *Distributed Computing - IWDC 2003*, volume 2918 of *Lecture Notes in Computer Science*, pages 833–833. Springer Berlin / Heidelberg, 2003.

[105] Craig Silverstein, Sergey Brin, Rajeev Motwani, and Jeff Ullman. Scalable techniques for mining causal structures. In *Data Mining and Knowledge Discovery*, pages 594–605, 1998.

[106] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explor. Newsl.*, 1(2):12–23, 2000.

[107] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explor. Newsl.*, 1(2):12–23, 2000.

[108] Kelly Storm and Eileen Kraemer. Web site evolution: Usability evaluation using time-series analysis selected episode graphs. In *WSE '09: Proceedings of the Ninth IEEE International Symposium on Web Site Evolution*, pages 27–38, 2009.

[109] Rick Tarleton. Center for tropical and emerging global diseases, Nov 2010. `http://www.ctegd.uga.edu/`.

[110] Stephen Turner. Analog. http://www.analog.cx/, 2011.

[111] Tusker. Regular expression benchmarks. http://www.tusker.org/regex/20100713.html, 2011.

[112] Wikipedia. Application programming interface. Accessed Online, 2010. `http://en.wikipedia.org/wiki/API`.

[113] Wikipedia. Arpanet. Accessed Online, 2010. `http://en.wikipedia.org/wiki/Arpanet`.

[114] Wikipedia. Bayesian probability. Accessed Online, 2010. `http://en.wikipedia.org/wiki/Bayesian_probability`.

[115] Wikipedia. Cognitive architecture. Accessed Online, 2010. `http://en.wikipedia.org/wiki/Cognitive_architecture`.

[116] Wikipedia. Cognitive model. Accessed Online, 2010. `http://en.wikipedia.org/wiki/Cognitive_model`.

[117] Wikipedia. Data flow diagram. Accessed Online, 2010. `http://en.wikipedia.org/wiki/Data_flow_diagram`.

[118] Wikipedia. Document object model. Accessed Online, 2010. `http://en.wikipedia.org/wiki/Document_Object_Model`.

[119] Wikipedia. Ecmascript. Accessed Online, 2010. http://en.wikipedia.org/wiki/ECMAscript.

[120] Wikipedia. Gui. Accessed Online, 2010. http://en.wikipedia.org/wiki/Graphical_user_interface.

[121] Wikipedia. Hidden markov model. Accessed Online, 2010. http://en.wikipedia.org/wiki/Hidden_Markov_model.

[122] Wikipedia. Javascript. Accessed Online, 2010. http://en.wikipedia.org/wiki/JavaScript.

[123] Wikipedia. Markov propery. Accessed Online, 2010. http://en.wikipedia.org/wiki/Markov_property.

[124] Wikipedia. Naive bayes classifer. Accessed Online, 2010. http://en.wikipedia.org/wiki/Naive_Bayes_classifier.

[125] Wikipedia. Naive bayes classifer. Accessed Online, 2010. http://en.wikipedia.org/wiki/Naive_Bayes_classifier.

[126] Wikipedia. Personalization. Accessed Online, 2010. http://en.wikipedia.org/wiki/Personalization.

[127] Wikipedia. Remote procedure call. Accessed Online, 2010. http://en.wikipedia.org/wiki/Remote_procedure_call.

[128] Wikipedia. Usability. Accessed Online, 2010. http://en.wikipedia.org/wiki/Usability.

[129] Wikipedia. Web mining. Accessed Online, 2010. http://en.wikipedia.org/wiki/Web_mining.

[130] Wikipedia. Ajax (asynchronous javascript and xml). Accessed Online, 2011. http://en.wikipedia.org/wiki/Ajax_(programming).

[131] Wikipedia. Bounce rate. http://en.wikipedia.org/ wiki/Bounce_rate, 2011.

[132] Wikipedia. Heuristic. Accessed Online, 2011. http://en.wikipedia.org/wiki/Heuristic.

[133] K.-L. Wu, P. S. Yu, and A. Ballman. Speedtracer: a web usage mining and analysis tool. *IBM Syst. J.*, 37(1):89–105, 1997.

[134] Yitong Wang Zhenglu Yang and Masaru Kitsuregawa. An effective system for mining web log. *Lecture Notes in Computer Science*, pages 40–52, 2006.

[135] Shi Zhou, Ingemar Cox, and Vaclav Petricek. Characterising web site link structure. *Web Site Evolution, IEEE International Workshop on*, 0:73–80, 2007.

Appendix A

List of Terms

- **Advanced Research Projects Agency Network (ARPANET):** ARPANET was created by a small research team at the head of the Massachusetts Institute of Technology and the Defense Advanced Research Projects Agency (DARPA) of the United States Department of Defense, was the world's first operational packet switching network, and one of the networks that came to compose the global Internet [113].

- **Application Programming Interface (API):** An application programming interface is an interface implemented by a software program that enables it to interact with other software. It facilitates interaction between different software programs similar to the way the user interface facilitates interaction between humans and computers [112].

- **Bayesian Probability:** is an interpretation of the concept of probability and lies within the category of evidential probabilities [114]. To evaluate the probability of an event, the Bayesian probabilist specifies a prior probability, which is updated based on the creation and insertion of new relevant data.

- **Cognitive Architecture:** Cognitive architectures are blueprints for intelligent agents that support computational processes attempt to simulate or mimic certain cognitive systems, such as a person [115]. Cognitive architectures form a subset of general agent architectures.

- **Cognitive Model:** A cognitive model is an approximation to cognitive processes for the purposes of comprehension and prediction [116]. Cognitive models are typically focused on singular cognitive processes, how multiple processes interact, and predictive analysis of behavior given a simple task or utility.

- **Data Flow Diagrams (DFD):** A data flow diagram is a graphical representation of the "flow" of data through an information system [117]. Elements within a DFD originate from external data sources and arrive at internal states where they either move to other existing internal states or to an external data source. It differs from the flowchart as it shows the data flow instead of the control flow of the program.

- **Document Object Model (DOM):** The Document Object Model is a cross-platform and language-neutral convention for representing and interacting with objects in HTML, XHTML and XML documents [51]. The DOM's elements may be accessed and manipulated within the syntax of the programming language in use, such as JavaScript. The public interface of a DOM is specified in its Application Programming Interface [118].

- **ECMAScript:** ECMAScript is the scripting language standardized by Ecma International in the ECMA-262 specification and ISO/IEC 16262. The language is typically used for client-side scripting run by the web browser, using one of several languages such as JavaScript, JScript, and ActionScript [119].

- **Episode:** An episode is a subset of the requests found within a stored session. In can also be thought of as a subset of related user clicks that occur within a user session [40].

- **Graphical User Interface (GUI):** A graphical user interface is a type of user interface that allows users to interact with programs in more ways than typing such as selecting icons on a computer screen [120].

- **Hidden Markov Model (HMM):** A hidden Markov model is a statistical Markov model in which the system being modeled is assumed to be a Markov process with hidden states [121]. An HMM can be considered as the simplest dynamic Bayesian

network.

- **Hierarchical Unsupervised Niche Clustering (H-UNC) [86]:** Nasraouis H-UNC algorithm takes a set of sessions and groups them according to the largest discovered value of fitness for a series of sessions clustered based upon dissimilarity and weight. It uses a genetic algorithm to cut two random sessions in half and combine them to create two new child nodes. If the fitness of the cluster created by grouping the remaining nodes to either of the newly created child nodes then the child replaces the parent and the algorithm continues on until some predefined threshold has been met. Being an unsupervised algorithm, this entire process is repeated until, again, predefined thresholds are met.

- **Information Extraction (IE):** a type of information retrieval technique focused on automatically extracted structured information from unstructured digital documents [28].

- **JavaScript:** JavaScript is an implementation of the ECMAScript language standard and is typically used to enable programmatic access to computational objects within a host environment [122]. It can be characterized as a prototype-based object-oriented scripting language that is dynamic, weakly typed and has first-class functions. JavaScript is primarily used in the form of client-side JavaScript, implemented as part of a web browser in order to provide enhanced user interfaces and dynamic web sites.

- **Naive Bayes classifier:** A simple probabilistic classifier based on applying Bayes' theorem with strong independence assumptions [125].

- **Markov Model:** A model that inherently contains the Markov property. The Markov model is a way to represent a changing set of states over time, where there is a known probability or rate of transition from one state to another [101].

- **Markov Property:** A process has the Markov property if the conditional probability distribution of future states of the process depend upon the present state [123].

- **Profile:** A table showing the frequency of occurrence of each URL within the collection of sessions [80].

- **Remote Procedure Call (RPC):** A remote procedure call is a synchronous or asynchronous message that allows a computer program to execute a subroutine or procedure in another address space, usually on another computer, without the programmer explicitly defining the action. [127]

- **Request:** A message describing an atomic operation to be carried out in the context of a specified resource [40]. For example, a user 'requests' information such as a web page through a web browser.

- **Session/Visit:** A session is a semi-permanent interactive information interchange between a computer and user. A session is set up or established at a certain point in time, and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically stateful, where information of the requests made by a user within a session are stored within a web server access log [40].

- **Session similarity:** The level of similarity between two sessions. A brief description of the logic behind the SCFC and H-UNC algorithm should also be provided for the readers benefit. Although the H-UNC was not used on the dataset used for the experiments, it will still be explained to, at the very least, understand our intent [11].

- **Supervised algorithm:** Any algorithm which is seeded or guided by a human to help determine the outcome or results.

- **Supervised Competitive Fuzzy Clustering (SCFC):** Supervised competitive fuzzy clustering takes a specified number of sessions from a human user and groups the remaining number of sessions within a given dataset according to their similarity to the N specified sessions. Similarity is determined by calculating the pair-wise syntactic and session level similarity of each representative session within the N number of specified sessions to the remaining sessions [108].

- **Syntactic similarity:** The level of similarity between two URLs based on their structure [79]. For example, the URL /one/two/three is more similar to /one/two/four than it is to /one/three/two since syntactic similarity is computed first by splitting up the URL by directory then iteratively comparing the directory names by position.

- **Transaction-Oriented View (TOV):** A graphical representation of web workloads presented in a transaction-oriented manner. Instead of viewing actions independently, TOVs group related actions as transactions to create a broader view of the system and thus help isolate issues more quickly. [103]

- **Unsupervised algorithm:** Any algorithm started and guided automatically to determine the outcome or results.

- **Uniform Resource Identifier (URI):** A Uniform Resource Identifier is a standardized sequence of characters used to identify resources. URIs have a flexible, generic syntax and process for reference resolution. The syntax consists of a grammar that is a superset of all valid URIs, which helps explain the structure's flexible nature. [24]

- **Web Server:** A web server is a computer program that services content, such as web pages, using the Hypertext Transfer Protocol (HTTP), over the World Wide Web. The term web server can also refer to the computer or virtual machine running the program [40].

- **Web Server Access Log:** A web server access log is a log file automatically created and maintained by a server of activity performed by it [40]. A typical example is a web server log which maintains a history of page requests. Information about the request, including client IP address, request date/time, page requested, HTTP code, bytes served, user agent, and referrer are typically added. These data can be combined into a single file, or separated into distinct logs, such as an access log, error log, or referrer log.