# INTERACTIVE VISUAL ANALYSIS OF WEB LOG DATA

SRINIDHI KANNAPPADY

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

APRIL 2007

*Your file Votre référence*
*ISBN: 978-0-494-28952-5*
*Our file Notre référence*
*ISBN: 978-0-494-28952-5*

# ABSTRACT

INTERACTIVE VISUAL ANALYSIS OF WEB LOG DATA

SRINIDHI KANNAPPADY

With growing number of applications using web based transactions and services, development of tools and techniques for analyzing large website usage data for enabling deeper insights into usage trends and hidden patterns is a major requirement today. We propose a solution approach for interactive visual analysis of website usage, which integrates usage modeling and visual rendering. Our approach not only makes the approach scalable to large data by reducing the time for visual rendering, but also enables easier interaction for subsequent analysis by visually presenting responses to queries in the global context of historic usage behavior. As the first step, we apply a fuzzy clustering technique to web log data to obtain a usage model and image it through a two-view display showing a point cloud rendering of clustered sessions and the website page hierarchy. Since web usage data is high-dimensional and non-Euclidean, we combine two dimensionality reduction techniques, namely Multidimensional Scaling and Sammon mapping for transforming the usage session data into displayable primitives. To demonstrate the effectiveness of our approach, we have developed a prototype system and performed experiments using large datasets which supports (1) clickstream visualization to identify macro level trends in real time, (2) decision on when web site restructuring is needed based on a proposed *cost* model while interactively rearranging nodes in the hierarchical display, and (3) visual depiction of noise for intuitively deciding upon when to carry out the expensive process of reclustering data. We also illustrate the effectiveness of the proposed approach using some benchmark datasets.

# Acknowledgements

I am very grateful for the advice and support of my supervisors, Dr. Sudhir P. Mudur and Dr. Nematollaah Shiri. Thanks to Dr. Mudur for his enthusiasm, inspiration, and great efforts to explain things clearly and simply. Throughout my thesis period, he provided encouragement, sound advice, good teaching, and lots of good ideas. I would have been lost without him. Thanks to Dr. Shiri for many insightful thoughts during the development of the ideas in this thesis and also for constructive comments and suggestions throughout the development of the thesis. I further extend a special thanks to them for giving me opportunities to present papers, which we co-authored, at international conferences, which was an experience in itself.

I also wish to thank all my colleagues in the graphics and visualization lab and the database labs at Concordia, for sharing their ideas, and for their valuable suggestions. A special thanks to Bhushan S. Suryavanshi for his contribution in the development of the RFSC algorithm used in this research.

I would also like to thank the CSE department at Concordia for providing the web log records used as part of our experiments.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Web usage analysis of large and popular websites can provide vital information about online business transactions that can be used by business administrators to improve the services provided through their websites. As the World Wide Web continues to grow in popularity, better tools for understanding web usage data are needed. For analysis purposes, raw web usage data can be organized in many different ways. For example, usage data could be organized into a collection of sessions, ordered/unordered in time, where each session is defined as a set of pages accessed by a user in a continuous fashion without a long break. Another organization would be that of a stream or a time sequence of page clicks, known as clickstream data. Data mining and visualization techniques are among the most prominent techniques being researched for analysis of web usage data. Data mining techniques, for example, are used to extract web usage profiles from web logs. Profiles refer to information about the preferences of individuals for a given web site. The navigational behavior and preferences are learnt typically through analysis of the user access logs recorded by the web server. While users cannot be identified from anonymous web server logs, usage can be anonymously tracked through the server logs. In our work too, the emphasis is on 'usage' profiles rather than 'user' profiles. Moreover, it is these anonymous users that every company wishes to attract more. The collection of usage profiles reflects the semantics of the historic user access dataset at a particular point of time. Content creators or web designers can then use this information to identify which pages are used more often by which user groups, how long a page is viewed, and in what order they are accessed. Profiles can also be used for path prediction, pre-fetching of pages, improving the structure of websites, and in a nutshell improving the user's experience of interacting with the website.

# 1.1 Visual Analysis of Data

Information overload is considered as one of the fundamental human-computer interaction problems today. In an effort to cope with this problem, more and more users are turning to data visualization. Data visualization is the process of "representing data as a visual image" [Latham 1995] in which an image is created using a combination of graphics primitives, say points, lines, triangles, etc. along with attributes like color, texture, shades, etc. to represent different measured quantities [Tufte 1983]. Visual data analysis aims at integrating the human in the data exploration process, enabling enhanced use of a human's perceptual abilities for analyzing large data sets which is commonly encountered in today's computing environments. The basic idea of visual data analysis is to suitably transform and present the data in a visual form, allowing the human to get insight into the data, draw conclusions, and directly interact with the data. Visual data mining techniques have proven to be of high value in exploratory data analysis and they also have a high potential for exploring large databases [Keim 2002]. Visual analysis of the dataset or of a suitably transformed version of it combined with intuitive interactive control to examine the image can help the user gain insights, i.e., say, detecting customer purchase patterns, web usage patterns, etc. This potential to "see" patterns attracts decision makers to use data visualization as a tool to better understand the dataset and thus make better decisions. Data visualization has helped users in a broad range of industries around the world to make critical business decisions by allowing them to eliminate the information overload [Lathman 1995].

## 1.2 Characteristics of Web Log data

There are a number of distinct characteristics of web usage data, in our context, that make the development of appropriate tools and techniques for data mining and/or visual analysis of data rather difficult.

1) Web log data can be enormous in volume. Never before in history has data been generated at such high volumes and rates as it is today. Exploring and analyzing the vast volumes of data becomes increasingly difficult. For example, the web log data of our department at Concordia University had over half a million user clicks during just the summer of 2005 from May to August.

2) The typical dimensionality of this data is excessively large. Again, consider the example of our department's web site; it has over 10,000 web pages[1]. A session is a binary vector of this large dimensionality, each bit corresponding to a unique web page. Most mining and visual analysis techniques do not scale well to such large datasets and dimensions.

3) Web log data is inherently sparse and fuzzy. In any single session, depending on the purpose, a user typically visits only a small subset of the pages on the website. Also a user may browse the same page for different purposes. Each time the user accesses the

[1] This is not unusual; in addition to administration and service web pages, our web site hosts pages for faculty, individual courses, teaching assistants, graduate students, publications, seminars, etc.

site, he or she may have different browsing goals. Furthermore, the same user in the same session may have different goals and interests at different times.

4) Web log data is non-Euclidian and does not admit simple transformations into Euclidean space without distortion of interrelationships inherent in the dataset.

While it is possible to put these data into tables and to analyze the data with statistical software or data mining applications, there is often a need for visual analysis to enable researchers to interpret the meaning of the available data. While visual data analysis is especially useful when little is known about the raw data and the exploration goals are vague, development of scalable and interactive techniques remains a major challenge. In particular, it is hard, if not impossible, to develop truly effective visualization techniques by directly rendering the voluminous raw web log data with a large number of dimensions, as discernability and understandability decrease with the amount of data that is displayed. We show that the approach proposed in this thesis of integrating usage modeling with visual rendering is a promising solution that addresses the above problem.

# 1.3 Contributions

The main contributions of this research are the following:

1) A new approach has been proposed for supporting interactive visual analysis of very large web log data by closely coupling the processes of visual rendering and usage modeling. Usage modeling enables us to make the visual rendering process scalable to very large datasets. We have developed a working prototype implementation which applies the relational fuzzy subtractive clustering (RFSC)

to obtain the usage model from historic web log data in the form of usage profiles and then transforms the modeled usage data into a point cloud rendering. Each cloud depicts a user group with a distinct profile. As each new session data is gathered, it is appropriately added in real time to the point cloud, using the incremental RFSC algorithm and a suitable mapping technique, in such a manner as to visually reflect its relationships with all the clusters in the usage model. The prototype uses a two-view display for interactive analysis, one for displaying the point cloud and other for displaying the hierarchical website layout. For analysis purposes, the point cloud view provides the context by visually depicting historic usage profiles, and the hierarchy display shows the detail in terms of URLs and supports page and structure specific interactions. The effectiveness of this approach is demonstrated with help of a number of experiments conducted on large web log data of over half a million user clicks.

2) Since web usage data is high dimensional and non-Euclidean, we have devised a method for mapping usage group centers into 3D Euclidean space by combining and suitably adapting two dimensionality reduction techniques, Multidimensional scaling (MDS) and Sammon mapping (SM). This combined method, while preserving the similarity relationships in the usage data to a great extent, provides a good mapping of usage profiles into 3D points and we show this by a comparative study with other competing methods [Kannappady et al. 2006a]. We have made clever use of the results of the fuzzy clustering technique to enable us to image all the user sessions into a point cloud display.

3) Using this two-view display, we demonstrate a number of techniques with specific goals for interactive visual analysis. These include:

   - *Helping a web administrator decide on remodeling of web log data*: The usage model changes over time. So, at appropriate times, we need to rebuild usage profiles. This is an expensive process. While some of the new user sessions when incrementally added get correctly incorporated into an existing model, other new user sessions would usually be classified as noise. Our imaging method images the latter sessions as noise in the point cloud domain. Whenever the web administrator visually sees the amount of noise as unacceptable, he/she can invoke the computational process for reclustering the web log data [Kannappady et al. 2006a].
   - *Visualizing usage behavior in real-time for discovering current macro-level trends*: User click stream data is fed into the interactive system in real time, active sessions are updated, incrementally added, mapped and imaged in a suitably highlighted fashion. Clouds with significant highlights clearly show the profiles into which new users are attracted [Kannappady et al. 2006b].
   - *Web site restructure analysis*: Another advantage of visualizing the web page hierarchy displayed along with the point cloud model is that it enables visual responses to "if-then" type of questions. Say, for example, the question is, "If I move around some pages/nodes in the current web structure, then what impact will it have on different user groups; which users have to click more/less links to reach their pages of interest?" Interactively editing these changes in the hierarchy view window and visually seeing the impact in the

point cloud view window can substantially ease decision making on when and how to remodel the web site structure.

## 1.4 Outline of Thesis

The rest of the thesis is organized as follows:

In Chapter 2, we provide an overview of the relevant literature in visualizing web usage data and visualizing clickstream data. In Chapter 3, we provide the required background by first briefly describing the clustering algorithm we have used for mining the web usage data. This is followed by a discussion on applicable dimensionality reduction techniques. Next we present our approach for mapping high dimensional non-Euclidean data into 3D space. We also describe some details from the comparative study of different dimensionality reduction techniques for deciding on the specific method(s) to use for positioning in 3D space, the cluster centers to best preserve the similarity relationships amongst them. In Chapter 4, we discuss our algorithms for mapping all other sessions (non cluster centers) from historic usage data into a point cloud model and for incorporating new session data gathered in real time. We also illustrate its application through experiments on very large web site log data. In Chapter 5, we discuss a number of illustrative examples of interactive visual analysis using our approach. We conclude our work in Chapter 6 with our observations on the research results achieved so far in this proposed approach and a list of possible directions for further development.

# Chapter 2

# Related Work

Two visualization aspects of the web usage data namely, visualizing the mined web usage data and visualizing the clickstream data are of primary concern in our research work. In this chapter, we provide an overview of the relevant literature in these two aspects of web visualization.

## 2.1 Web Log Data Organization

All the pages of a website accessed by different users are logged by the web server in the form of a sequence of pages (URLs) that are accessed. Log files produced on the web servers are text files with a row for each HTTP transaction. Typically each row consists of IP-address of the client who made the request to the server, the time when the request was made and the URL of the page that was requested. For analysis purposes, web log data can be organized in a number of ways:

a) A stream of web page clicks, essentially a time sequence of URLs.

b) A collection of user sessions. (A session is typically the URLs of pages visited by a user from the moment the user enters a website to the moment the same user leaves it. In our case, a user is assumed to have left the website, if there is no activity by the user for 45 minutes or more. )

c) A sequence of user sessions, ordered according to the session end time.

d) A sequence of incrementally updated sessions; as each new page is clicked by a user, her/his active session is updated to reflect this page visit and the updated session are added to the sequence.

## 2.2 Visualization of web usage data

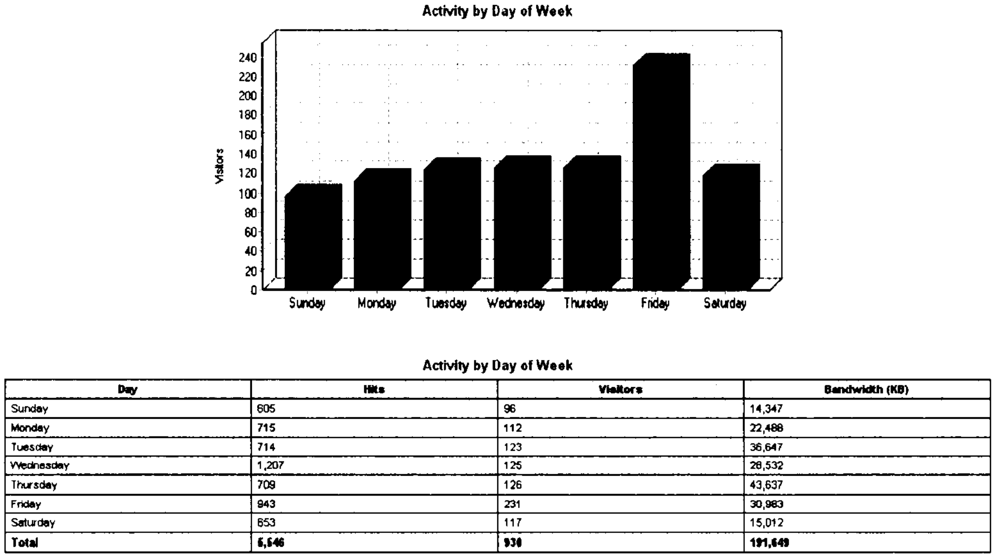
The navigational behavior and preferences of different users are learnt usually through analysis of such user access logs, typically in the form of "usage profiles". We emphasize on 'usage' profiles rather than 'user' profiles because users cannot be identified from anonymous web server usage logs [Cooley 1999]. On the other hand, usage can be traced through the server logs, which are basically user access history datasets. Moreover, it is these anonymous users that every company wishes to attract and get involved with its website when they first visit the site. The collection of usage profiles reflects the semantics of the historic user access dataset at a particular point of time, and can be used in many different applications. One major application is their use by content creators or web designers to identify which pages are used more often, how long a page is viewed, and in what order they are accessed. While data mining primarily focuses on the use of algorithmic techniques, data visualization relies on human visual perception capabilities to understand and get insight into large and complex datasets. Visual representations of the dataset or of a suitably transformed version of it combined with intuitive interactive control to examine the image can help the user gain insights, i.e., detecting customer purchase patterns, web usage patterns, etc. Below we discuss a number of earlier efforts made in visualizing the web usage data which are reported in the literature.

It is important to distinguish between visualization of website structure and web usage. There is considerably more work on the former. However, since our focus here is on web usage, and not on visualizing the website structure, for interested readers, we shall only provide pointers to web structure visualization techniques. Since the web structure is more like a graph, results from the entire field of research devoted to the general problem of visualizing graph structures become applicable. Further, web site structures have certain properties, e.g., hierarchical organization of links/pages, which make visualizing them slightly easier than visualizing a general graph. The first-level goal of early web visualization work was to help users navigate more effectively by visually representing the non-linear information access structure. One of the earliest solutions is based on the *cone tree* visualization technique [Herman and Marshall 2000] for a medium size hierarchy (of about 100 nodes) [Chi 1994]. Cone trees are hierarchies laid out uniformly in three dimensions. Andrews described a landscape approach for small websites [Andrews 1995], and Munzner used a 3D hyperbolic browser to map fairly large websites [Munzner 1998]. Inxight Software's (www.inxight.com) SiteLens used the hyperbolic tree technique to visualize web site structure, while NicheWorks [Wills 1997] used an angular layout similar to disk trees [Chi 2002].

In comparison web usage visualization has received less attention. Unfortunately, it is hard, if not impossible, to visualize sparse voluminous data of a large number of dimensions of numerous items in a workable manner, as comprehension decreases with the amount of data displayed [Herman et al. 2000]. Several approaches have been applied to reduce the amount of data, such as binding, filtering and hierarchy of documents or usage patterns, [Chi 1994, Chen 1999]. However, the resulting abstract views are often too complex to interpret or compare.

There are a few commercial and research systems available which use other visualization techniques in order to envision the log file contents. However, the most popular technique which has been employed in these packages is a bar graph. WebLog Expert [http://www.weblogexpert.com] is a powerful commercial access log analyzer. It provides information about the site's visitors, activity statistics and accessed files. Fig. 1 shows the log file dataset and bar graphs generated by Weblog Expert. The 2D bar graph in Fig. 1 shows activities such as number of hits, number of visitors, bandwidth, etc., grouped into days of the week.

Although traditional graph visualization is easy to understand [Waterson 2002], it is not scalable. Besides it makes it difficult to visualize additional metadata, such as usage profiles, page view times, trends in user interests, page relevancy or order of access, etc.



Activity by Day of Week

| Day | Hits | Visitors | Bandwidth (KB) |
|---|---|---|---|
| Sunday | 605 | 96 | 14,347 |
| Monday | 715 | 112 | 22,488 |
| Tuesday | 714 | 123 | 36,647 |
| Wednesday | 1,207 | 125 | 28,532 |
| Thursday | 709 | 126 | 43,637 |
| Friday | 943 | 231 | 30,983 |
| Saturday | 653 | 117 | 15,012 |
| **Total** | **5,546** | **930** | **191,649** |

**Figure 1: Using 2D bar graphs to visualize log file dataset [http://www.weblogexpert.com]**

[Stephen 2000] used a site map to show the page usage in the web site; the hierarchic structure of the website is visualized by applying a real world tree metaphor. Fig. 2 shows an example of this kind of visualization. Each slice of disk tree represents one week's worth of content, usage and structure changes on web site. This technique works well for relatively small sites. However, the disadvantage of this method is that for large sites, display size limitations make it impossible to show every page in the web site. Also it is not possible to include other parameters present in the log files in this visualization.

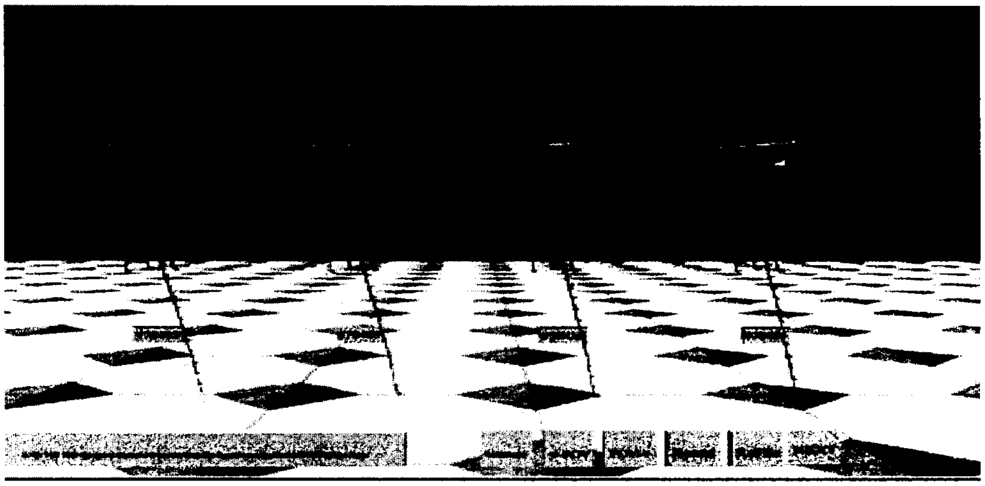


**Figure 2: Time tube visualization [Stephen 2000]**

# 2.3 Visualization of clickstream data

Since the business world has been evolving towards increased use of e-commerce, analyzing the data of clients that visit a company's website is becoming a necessity in order to remain competitive. This analysis can be used to discover two things for the company, the first being an analysis of a user's clickstream while using a website to reveal usage patterns, which in turn gives a heightened understanding of customer behavior. Secondly, it can be used to improve customer satisfaction. On a Web site, clickstream analysis (sometimes called as clickstream analytics) is the process of collecting, analyzing, and reporting aggregate data about which pages are visited and in what order,  as a result of the succession of mouse clicks a visitor makes (that is, the clickstream). Because a large volume of data can be gathered through clickstream analysis, many e-businesses rely on pre-programmed applications to help interpret the data and generate reports on specific areas of interest. This use of the analysis creates a usage profile that aids in understanding the types of people that visit a company's website. As discussed in [Van 2005], clickstream analysis can be used to predict whether a customer is likely to purchase a product from an e-commerce website. Clickstream analysis can also be used to improve customer satisfaction with the website and through interacting with the company itself. Both of these generate a significant business advantage. Since the clickstream data for any website is generally huge, visualization aids become necessary to provide an insight into the hidden patterns. Below, we review a number of relevant previous research reports on clickstream visualization.

WebQuilt [Hong and Landay 2001] is a tool that uses a proxy server to log the user's clickstream. The closest visualization work to that of WebQuilt is QUIP [Helfrich and Landay 1999], which is essentially a logging and visualization environment for Java applications. WebQuilt builds on QUIP by extending the logging and visualization to the web domain. WebQuilt uses directed graphs to construct a visualization of the user's browsing path. The thickness and the color of the arrows indicate the user's browsing behavior. The thicker arrows indicate a more heavily traversed path, while the darker arrows mean that more time is spent [Fig. 3].

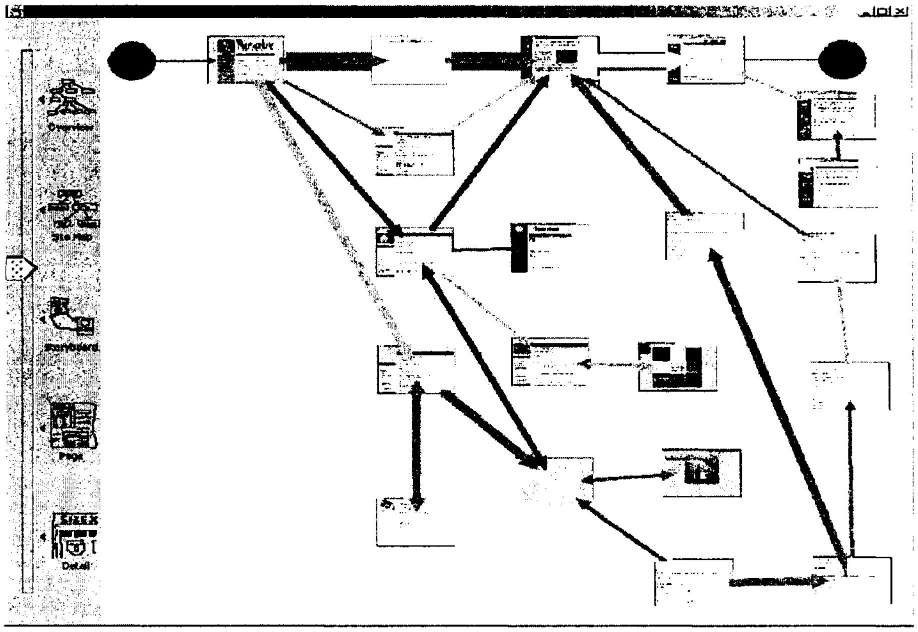


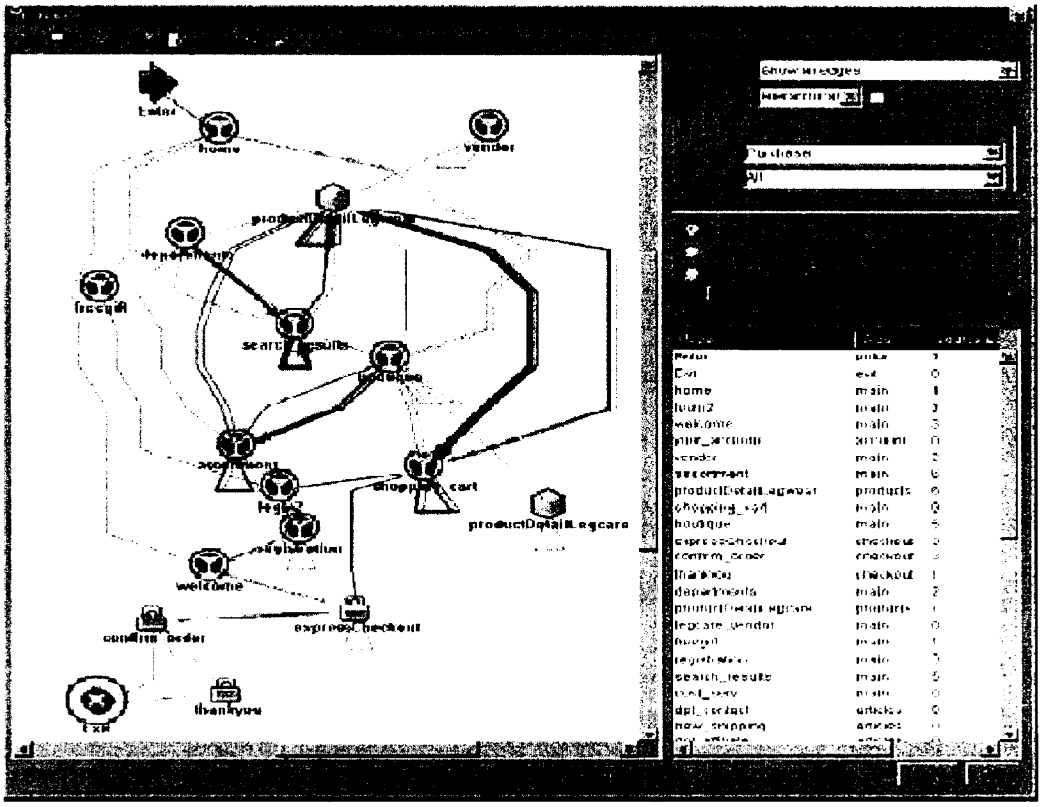
**Figure 3: Representation of WebQuilt tool visualizing the clickstream data [Hong and Landay 2001]**

Another system that is similar to WebQuilt's logging software is Etgen and Cantor's Web-Event Logging Technique (WET) [Etgen and Cantor 1999]. WET is an automated usability testing technique that works by modifying every page on the server. It can automatically and remotely track user interactions. WET takes advantage of the event handling capabilities built into the Netscape and Microsoft browsers.

ClickViz [Brainerd and Becker 2001] is a tool for analyzing the behavior of user of a website. This is done using a graph structure in which each node represents a web page in the website, labeled with the name of the web page, and an edge connects two nodes, indicating a set of transitions between the connected web pages. Each edge in the graph is colored differently depending on the customer segments. The width of the edge is proportional to the number of transitions in the set. As the data becomes incomprehensible, they filter out the edges based on weight or based on the current node selection. Fig. 4 shows a sample from ClickViz output and the corresponding hierarchical layout.

VISVIP [Cugini and Scholtz 1999] is another tool that visualizes the user path recorded by the web log, by laying out a 2D graph of the website using enhanced force-directed algorithm. Like ClickViz, VISVIP represents web pages as nodes in the graph and the links between pages as edges. Nodes are colored by type. The user can select which user paths to display once the graph of the website is obtained. Every user is assigned a unique color. Fig. 5 shows a sample screen shot of a display produced using this tool.

A few of the visualization tools use 3D or multidimensional graphics, which can incorporate more features in one graph. Examples of such tools include Parallel Coordinate [Inselberg and Dimsdale 1999] and Scalable Framework [Lopez et al., 2001].



| [illegible] | [illegible] | [illegible] |
|---|---|---|
| Return | [illegible] | [illegible] |
| Exit | exit | 0 |
| home | main | 1 |
| [illegible] | main | [illegible] |
| welcome | main | 3 |
| [illegible] | [illegible] | 0 |
| vendor | main | [illegible] |
| [illegible] | main | [illegible] |
| productDetail_[illegible] | products | [illegible] |
| shopping_cart | main | 0 |
| boutique | main | [illegible] |
| express_checkout | checkout | [illegible] |
| confirm_order | checkout | [illegible] |
| thankyou | checkout | 1 |
| departments | main | 2 |
| productDetail[illegible] | products | [illegible] |
| [illegible] | main | 0 |
| [illegible] | main | 1 |
| [illegible] | main | [illegible] |
| search_results | main | 5 |
| [illegible] | main | 0 |
| dpt_[illegible] | articles | 0 |
| now_shipping | articles | [illegible] |

**Figure 4: ClickViz window showing the hierarchical layout [Brainerd and Becker 2001]**

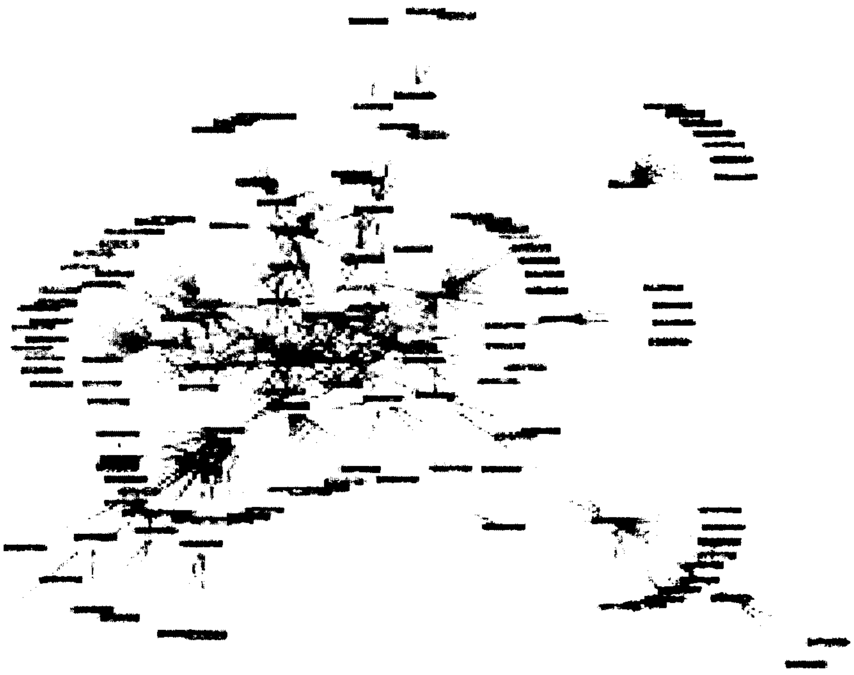


**Figure 5: 2D web site layout using VISIP [Cugini and Scholtz 1999]**

## 2.4 Comparison with our approach

It is important to note that we find it difficult to visualize how the visualization techniques used in all these tools can scale up to handle clickstream data consisting of millions of records. Further, none of them keep any visual record of usage history, thus making it difficult to gage patterns and trends. Our work differs from all the above mainly in the following aspects: (1) we provide a two-view display which shows the website hierarchy structure and provides detailed, page level control, along with a point cloud visual of historic usage model which provides the global context. (2) we closely couple our visualization technique with a data mining technique that discovers usage patterns in the form of usage profiles and images user sessions (historic as well as recent ones) in the form of a point cloud; this enables us to analyze the "validity" of a historic usage model by looking at the amount of noise in the point cloud rendering or to animate active users' click data by overlaying it on a point cloud rendering of clustered historical usage data or to provide interactive support for website restructure analysis by looking at the changes in "click-path" costs for different user groups.

# Chapter 3

# Background & Notations

This chapter briefly presents the background techniques on which we have developed our new approach and the notations used in our work. We begin with the web log mining technique used in our approach to produce usage profiles in the form of clusters. We then study basic dimensionality reduction and visualization techniques needed for understanding our method of mapping cluster centers into 3D Euclidean space. Lastly we describe the methods we have adapted for use in our approach and the corresponding comparative performance study of different dimensionality reduction techniques applied to the dataset used in all our experiments.

## 3.1 Web Usage Mining

In this section, we review some basic concepts in web usage mining, and provide an overview of relational fuzzy subtractive clustering algorithm (RFSC) [Suryavanshi et al. 2005a], which we have used for clustering web usage data and to obtain historic usage profiles. The access log file recorded by a web server is at a very fine granularity. Cleaning is done to remove log entries for image files and other such components within a web page, details of which are explained in Appendix A. While there are a number of clustering techniques proposed for fuzzy clustering of web log data, in our work we have opted to use RFSC [Suryavanshi et al. 2005a], because of its clear advantages for this kind of application, namely efficiency, scalability to handle large data sets, and ability to handle noise in web usage data.

Clustering is the process of partitioning of a collection of objects into disjoint subsets or clusters such that objects in the same cluster have some common properties that distinguish them from objects in other clusters. Fuzzy clustering is a generalization of this process, where the clusters are not necessarily subsets of the collection, but instead they are fuzzy subsets, based on the notion of fuzzy sets introduced by [Zadeh 1965]. Apart from fuzzy clustering there are many other fields where fuzzy set theory can be applied, e.g., mechanical subsystems [Barr et al. 1996, Hyniova et al. 2001], Image processing and retrieval [Lapanja et al. 1997, Kannappady et al. 2006c], Pattern recognition [Ferguson and Dunlop 2002], etc. In fuzzy clustering, each object is assigned a number in the range [0, 1] with respect to every cluster, called as *grade of membership.* Objects which are similar to each other are identified by having high memberships in the same cluster. "Hard" or crisp clustering algorithms assign each object to a single cluster that is using the two distinct membership values of 0 and 1. In many situations, say, web usage data, this "all or none" or "black or white" membership restriction is not realistic as very often there may not be sharp boundaries between clusters and many objects may have characteristics of different classes with varying degrees. In such situations, it is more natural to assign each object a set of memberships, one for each cluster or class, in turn making class boundaries not hard but rather fuzzy. The main advantage of fuzzy clustering over hard clustering is that it yields more detailed information about the underlying structure of the data. We next focus on describing the Relational Fuzzy Subtractive Clustering (RFSC) algorithm.

## 3.1.1 Relational Fuzzy Subtractive Clustering

RFSC is based on the subtractive clustering algorithm proposed in [Chiu 1994], a technique widely used in fuzzy systems modeling. [Yager and Filev 1992] proposed a simple and effective algorithm, called the mountain method, for estimating the number and initial location of cluster centers. Their method is based on gridding the data space and computing a potential value for each grid point based on its distances to the actual data points; a grid point with many data points nearby will have a high potential value. The grid point with the highest potential value is chosen as the first cluster center. The key idea in this method is that once the first cluster center is chosen, the potential of all grid points is reduced according to their distance from the cluster center. Grid points near the first cluster center will have greatly reduced potential. The next cluster center is then placed at the grid point with the highest remaining potential value. This procedure of acquiring new cluster center and reducing the potential of surrounding grid points repeats until the potential of all grid points falls below a threshold. Although this method is simple and effective, the computation cost grows exponentially as the dimension of the problem increases. For example, a clustering problem with 4 variables and every dimension having a resolution of 10 grid lines would result in 104 grid points that must be evaluated. [Chiu 1994] proposed an extension of the mountain method, called subtractive clustering, in which each data point, as opposed to a grid point, is considered as a potential cluster center. The main advantage of this method is that it eliminates the need to specify a grid resolution, in which tradeoffs between accuracy and computational complexity must be considered. RFSC follows the same idea of considering each data object as a potential cluster center. It uses a potential function for relational data which is derived on the same lines as the one used in [Chiu 1994].

RFSC algorithm:

The potential $P_i$ of any object $x_i$ is calculated using the function:

$$P_i = \sum_{j=1}^{N_U} e^{-\alpha R_{ij}^2}, \text{ where } \alpha = 4/\gamma^2$$

where $R_{ij}$ is the dissimilarity between objects $x_i$ and $x_j$, $N_U$ is the number of objects to be clustered, and $\gamma$ is essentially the neighborhood calculated from the relational matrix R. The notion of *neighborhood-dissimilarity* ($\gamma_i$) of each object $x_i$ from every other object is defined as the median of dissimilarity values of $x_i$ to all other objects. The neighborhood-dissimilarity value $\gamma$ for the entire dataset is defined as the median of all $\gamma_i$'s, for $1 \leq i \leq N_U$. Further, the RFSC algorithm imposes a restriction of having normalized dissimilarity values, i.e., the relational matrix R is required to be a dissimilarity matrix such that $0 \leq R_{ij} \leq 1$. Therefore, $\gamma$ will always be a value in the range [0, 1]. This is a heuristic that was shown through experiments to work fine for many datasets.

The object with the highest potential $P_1^*$ is selected as the first cluster center. Next, the potential of each object is reduced proportional to the degree of similarity with this previous cluster center. Thus, there is larger subtraction in potential of objects that are closer to this cluster center compared to those which are farther away. After this subtractive step, the object $x_t$ with the next highest potential $P_t$ is selected as the next candidate cluster center. Now to decide whether this candidate cluster center can be accepted as an actual cluster center or should be rejected, two threshold values, called *accept ratio* (denoted as $\bar{\in}$) and *reject ratio* ($\underline{\in}$) are used where $0 < \underline{\in}$, $\bar{\in} < 1$, and $\underline{\in} < \bar{\in}$. If $P_t > \bar{\in} P_1$*, then $x_t$ is selected as the next cluster center, and this is followed by the subtractive step described above. If $P_t < \underline{\in} P_1$*, then $x_t$ is rejected, and the clustering algorithm terminates. If the potential $P_t$ lies between $\bar{\in} P_1$* and $\underline{\in} P_1$*, then it is said that the potential has fallen in the gray region. In this case, a check is performed to see if the object provides a good trade-off between having a sufficient potential and being sufficiently far from existing cluster centers. In such cases, it is selected as the next cluster center. This process of subtraction and selection continues until $P_t < \underline{\in} P_1$*, which is the termination condition. After finding the cluster centers, say C in number, membership values of different $x_j$ with respect to each cluster $c_i$ are calculated using the following formula:

$$u_{ij} = e^{-\alpha R^2_{c_{i,j}}}, \ i = [1..C] \text{ and } j = [1..N_U]$$

where $Rc_{i,j}$ is the dissimilarity of the $i^{th}$ cluster center $x_{ci}$ with the $j^{th}$ session $x_j$. When $x_j = x_{ci}$, we have that $Rc_{i,j} = 0$ and the membership value of $u_{ij} = 1$.

While there are a number of fuzzy clustering algorithms available, we have chosen the RFSC algorithm for the following reasons:

1) **RFSC does not require any user specified parameters**. In comparison, many of the other well known clustering algorithms [Corsini et al., 2004], user input parameter is a very critical factor that determines the quality of the clustering. When the dataset is small, it may be possible to estimate the number of clusters C. However, in case the dataset is huge, as is often the case with web log records, it is not clear how to estimate C correctly. In such cases, it is thus difficult to guarantee that the clustering results would manifest the true structure in the data.

2) **RFSC works well on large datasets**. The web log data recorded by the web server in our computer science department for a period of four months consisted of over half a million user clicks. Most of the clustering algorithms like ARCA [Corsini et al., 2004] etc. are inadequate to handle such large data. RFSC has shown itself to be a more effective method. We could obtain the clustering result in an hour and half.

3) **RFSC is relatively less sensitive to noise**. While most fuzzy C-means algorithms [Bezdek, 1982] impose the condition $\sum_{i=1}^{C} u_{ij} = 1$, RFSC does not. This effectively makes RFSC algorithm less sensitive to noise. Noise sessions are easily identified since their membership values will always lie on the asymptote of each of the clusters.

For our experiments, we have used the above mentioned web log data of our department. Data cleaning (preprocessing) was done to this log file to remove the log entries for image files and other such components within a web page. The dataset of half a million user clicks after cleaning is organized into 10,000 sessions. The RFSC algorithm is then applied to these 10,000 sessions. It produced 34 clusters. Then, the membership value of each session is calculated with every cluster center.

For visualizing usage data, we would need to apply a suitable dimensionality reduction technique to the raw session data. Given the high dimensionality and volume of this data, dimensionality reduction can be computationally prohibitive, as it would involve Eigen value analysis using a matrix of the size (10,000 x 10,000) iteratively. In particular, convergence can be a major problem as there are many near equal dissimilarity values in the dataset. Fortunately, RFSC provides us with a much smaller number (34) of cluster centers for this dataset. Therefore, we first map these cluster centers into 3D (explained later in section 3.3) and then use the fuzzy membership values to render the rest of the sessions (explained in Chapter 4). While this yields a computationally simple procedure, use of membership values for mapping helps ensure that the important relationships in the usage data with respect to user profiles are retained.

# 3.2 Dimensionality Reduction

In this section, we review a number of known techniques for dimensionality reduction. Dimensionality reduction techniques allow a dataset of high dimensional nature to be explored by projecting the data into low dimensional space such as the 2D plane or 3D space. Dimensionality reduction techniques have been applied in many fields, including psychology, cartography, machine learning, and information visualization, to name a few. We distinguish two major types of dimensionality reduction methods: linear and non-linear. Examples of linear methods include Principal Component Analysis (PCA) [Hotelling 1933] and projection pursuit [Friedman 1987], and for non-linear methods they include Multidimensional Scaling (MDS) [Kruskal 1964], Sammon Mapping (SM) Sammon 1969], triangulation method [Lee et al. 1977], and Self-Organization Map (SOM) [Kohonen 1989] etc.

Principle component analysis (PCA) is a linear dimensionality reduction method. PCA transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Projection pursuit tries to show the best visual representation that reveals most of the non-normally distributed structure of the data set. A neural network implementation of this method is introduced in [Fyfe and Baddeley 1995]. Linear dimensionality reduction techniques have a number of disadvantages. For example, PCA cannot take into account nonlinear structures and projection pursuit cannot project the nonlinear structures onto a low-dimensional display if the dataset has many dimensions and is highly nonlinear.

Several approaches have been proposed to project non-linear, high-dimensional structures onto a low-dimensional space. The most common methods allocate each individual data point onto a lower dimensional display and then optimize the display so that the distances between the points are as close as possible to the original distances. These methods differ in the selection of the objective function they optimize and the optimization strategy used.

Multidimensional scaling (MDS) is an established non-linear dimensionality reduction method which uses proximities among data points to produce a representation of the dataset [Shepard 1962]. The representation consists of a geometric configuration of the points on a map where each point corresponds to one of the data items. MDS is widely used in behavioral, economic, and social sciences to analyze the pair wise proximities of the data points (e.g., similarity of brands in a market survey). MDS is discussed in greater detail later in section 3.3.1. Another nonlinear dimensionality reduction method is Sammon mapping (SM). SM is closely related to MDS. It tries to optimize an objective function in order to preserve the relative pair wise distance between data points. Details on Sammon mapping are provided in Section 3.3.2.

Non-linear visualization methods are computationally very intensive for large data sets. The triangulation method can be used to reduce the computational complexity. An important property of the triangulation method is that the distances to its nearest two neighbors can be preserved exactly when inserting a new data item. Using the triangulation method, data items will be projected onto a map one by one and the nearest neighbor distances can always be preserved, that is, the generated map is based on a subset of distances in the original space. The projection process is, thus, substantially faster than other nonlinear visualization methods. However, the mapping result generated by the triangulation method may not be as accurate as the map by other non-linear visualization methods since the projection preserves only part of the distance.

# 3.3 Multidimensional Scaling

From a slightly more technical point of view, for a set of observed distances between every pair of $N$ items, multidimensional scaling methods aim to find a visual representation of the items in lower dimensional space in such a way that the resulting distances among items match the original distances as closely as possible. The metric version of MDS aims to find configurations of the data items where the resulting distances are as close as possible to the original distances of data items. Non-metric MDS methods try to keep the rank orders of the distances among data items as close as possible to the original rank orders. In our work, we will consider only metric MDS, in order to preserve the resulting distances as close as possible to the original distances, rather than preserving just the rank order of distances.

As mentioned earlier, the web usage data we have used in our experiments contains around 10,000 user sessions. Since it is computationally prohibitive and unstable to try and map this large dimension data in 3D, we first find the 3D space configuration for the cluster centers only, obtained from the output of the RFSC algorithm. The section below describes the procedure we followed for plotting these cluster centers in 3D while preserving the distance relationship among the points to reflect the original distance relationship, as much as possible.

## 3.3.1 Metric MDS

Metric MDS begins with an n× n distance matrix D with elements $d_{ij,}$ where 1≤ i,j ≤ n.

The objective of metric MDS is to find a configuration of points in p-dimensional space (p=3, in our case) from the distances between the data points such that the coordinates of the *n* points along the p dimensions yield an Euclidean distance matrix whose elements are as close as possible to the elements of D. Using the metric MDS, we obtain a positional configuration in 3D space for the cluster centers. Below we describe the steps of this process.

1. Start with the distance matrix D with elements $d_{ij}$.
2. Define matrix $A = -0.5d_{ij}^{2}$ . (1)
3. Obtain matrix B using the formula:

$$B = (a_{ij} - a_{i\bullet} - a_{\bullet j} + a_{\bullet\bullet}), \qquad (2)$$

where, $a_{i\bullet} = \frac{1}{n} \times \sum_{j=1}^{n} a_{ij}$

$a_{\bullet j} = \frac{1}{n} \times \sum_{j=1}^{n} a_{ij}$

$a_{\bullet\bullet} = \frac{1}{n^{2}} \times \sum_{j=1}^{n} a_{ij}$

4. Find the Eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ and the associated Eigenvectors $\gamma_1, \gamma_2, \ldots, \gamma_p$, where the Eigenvectors are normalized so that $\gamma_i^T \gamma_i = \lambda_i$

5. Choose a suitable number of dimensions p (in our case, p = 3)

6. The coordinates of the *n* points in the Euclidean space are given by:

$$x_{ij} = \gamma_{ij} \lambda_j^{1/2} \tag{3}$$

for i = 1,..., n and j = 1,..., p.

Using this algorithm, we get coordinates for the positioning of the 34 cluster centers in 3D as shown in Fig. 6.

As shown in [Johnson and Wichern 1998], since it is not possible to perfectly represent the original distances in a given lower dimensional space, a numerical measure is needed to indicate the closeness. This measure is called a *stress* function, defined in section 3.3.2. The stress value obtained from mapping these cluster centers using metric MDS was 0.423. This high stress value implies that fidelity to the original distance relationship is poor. This is due to low dimensional projection. To minimize this loss in fidelity, we suggest that we use these results from MDS as our initial choice for further refinement using Sammon Mapping, which is described in 3.3.2.

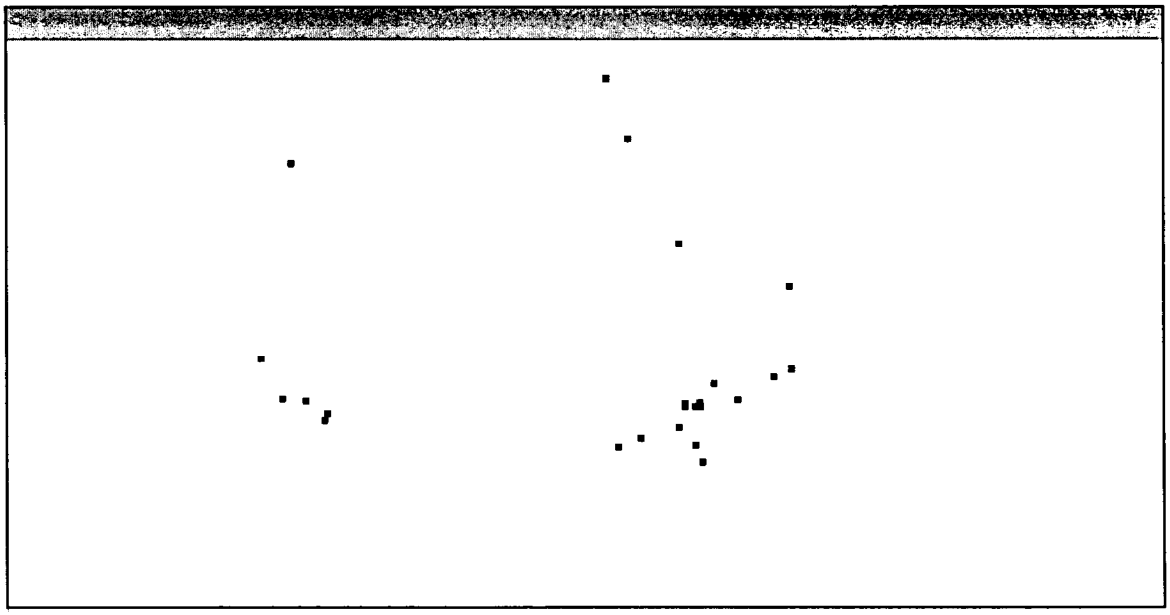


**Figure 6: Positions of cluster centers after MDS**

### 3.3.2 Sammon Mapping

Sammon Mapping (SM) is an unsupervised, nonlinear method that tries to preserve relative distances among the data items to be positioned in the space [Sammon 1969]. Here "unsupervised" means no targeted information or outcome to predict. To preserve the inherent distance relationships, the algorithm that generates a Sammon map employs a nonlinear transformation of the observed distances among data items when mapping data items from a high-dimensional space onto a low-dimensional space. Let $d^*_{ij}$ denote the distance (usually Euclidean distance) between two different data items $i$ and $j$ in the original dimensional space, and $d_{ij}$ denote the distance in the target projected space. Then the error function of SM is defined as follows:

$$E = \frac{1}{\sum_{i=1}^{n}\sum_{j=i+1}^{n} d^*_{ij}} \sum_{i=1}^{n}\sum_{j=i+1}^{n} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}} \qquad (4)$$

Here, smaller the error value *E*, better is the map we get. However, in practice, we are often unlikely to obtain perfect maps especially when the dataset is large and in high-dimensional space. Therefore, approximate preservation is what we can expect when projecting high-dimensional data onto a low-dimensional space. The SM algorithm starts with coordinate values obtained from MDS method as the input and works iteratively as follows:

Let *E(m)* be the mapping error after the *m*th iteration, i.e.,

$$E(m) = (1/c)\sum_{i<j}^{n}[d_{ij}^{*} - d_{ij}(m)]^2 / d_{ij}^{*} \qquad (5)$$

where $d_{ij}^{*}$ **is** the original distance matrix,

$$c = \sum_{i<j}^{n} d_{ij}^{*}\text{, and}$$

$$d_{ij} = \sqrt{\sum_{k=1}^{p}[y_{ik}(m) - y_{jk}(m)]^2}$$

The new *d*-space configuration at iteration step *m*+1 is given by:

$$y_{pq}(m+1) = y_{pq}(m) - (MF) \times \Delta_{pq}(m) \qquad (6)$$

where

$$\Delta_{pq}(m) = \frac{\partial E(m)}{\partial y_{pq}(m)} \Bigg/ \left| \frac{\partial^2 E(m)}{\partial y_{pq}(m)^2} \right|$$

and MF is the "magic factor" which has been determined empirically to be about 0.3 or 0.4. The partial derivatives are given by the following two formulae:

$$\frac{\partial E(m)}{\partial y_{pq}} = \frac{-2}{c}\sum_{j=1}^{n}\left[\frac{d_{pj}^{*} - d_{pj}}{d_{pj}d_{pj}^{*}}\right](y_{pq} - y_{jq}) \qquad (7)$$

$$\frac{\partial^2 E}{\partial {y_{pq}}^2} = \frac{-2}{c}\sum_{j=1}^{n}\frac{1}{{d_{pj}}^* d_{pj}}\left[\left({d_{pj}}^* - d_{pj}\right) - \frac{(y_{pq} - y_{jq})^2}{d_{pj}}\left(1 + \frac{{d_{pj}}^* - d_{pj}}{d_{pj}}\right)\right] \qquad (8)$$

This iterative process terminates when the Sammon stress value $E$ cannot be decreased anymore.

If the stress is zero, then the resulting pair wise distances are exactly the same as the pair wise distances in the original distance matrix. While, the stress value may not always reach zero, the iterative process is still useful as it yields a lower stress value, and as long as a certain amount of distortion is tolerable. [Kruskal 1964] provides a guideline for indicating tolerance corresponding to different stress values, as shown in Table 1.

**Table 1: Relationship between stress values and tolerance**

| Stress | Goodness of fit |
|---|---|
| 0.3 | Poor |
| 0.2 | Fair |
| 0.1 | Good |
| 0.025 | Excellent |
| 0.0 | Perfect |

## 3.3.3 Modified Increment Formula

It has been observed that if the distance values of any two points in the *d*-space are identical or near identical, then the Sammon stress $E$ will far exceed the value 1, which is not desirable. This is due to the fact that in computing $\Delta_{pq}(m)$, we have $d_{pj}$ in the denominator and when this becomes near zero, we have a divide by near zero condition in equations (7) and (8) and it naturally leads to an incorrect increment. In the original version also [Sammon 1969], it has been explicitly noted that the stress value can explode beyond 1 and when this problem is detected, it has to be suitably corrected programmatically. In our case, to overcome this problem, we modified the formula for increment computation by scaling the terms using $d_{pj}$, essentially to avoid the divide by near zero situations. The modified formula for increment is given below:

$$\Delta_{pq}(m) = \frac{\sum_{j=1}^{n}\left[\frac{d_{pj}^{*}-d_{pj}}{d_{pj}^{*}}\right](y_{pq}-y_{jq})}{\sum_{j=1}^{n}\frac{1}{d_{pj}^{*}}\left[\left(d_{pj}^{*}-d_{pj}\right)-(y_{pq}-y_{jq})^{2}\left(1+d_{pj}^{*}-d_{pj}\right)\right]}$$

We tested the SM algorithm with this modification using benchmark datasets such as *iris* and *wine* (http://www.ics.uci.edu/_mlearn/MLRepository.html) and confirmed that the results produced after this modification are the same as the results obtained using the original SM algorithm.

Fig. 7 shows the plotting of the cluster centers (same 2D view as in Fig. 6) of our experimental data using the SM algorithm with the modified increment formula.

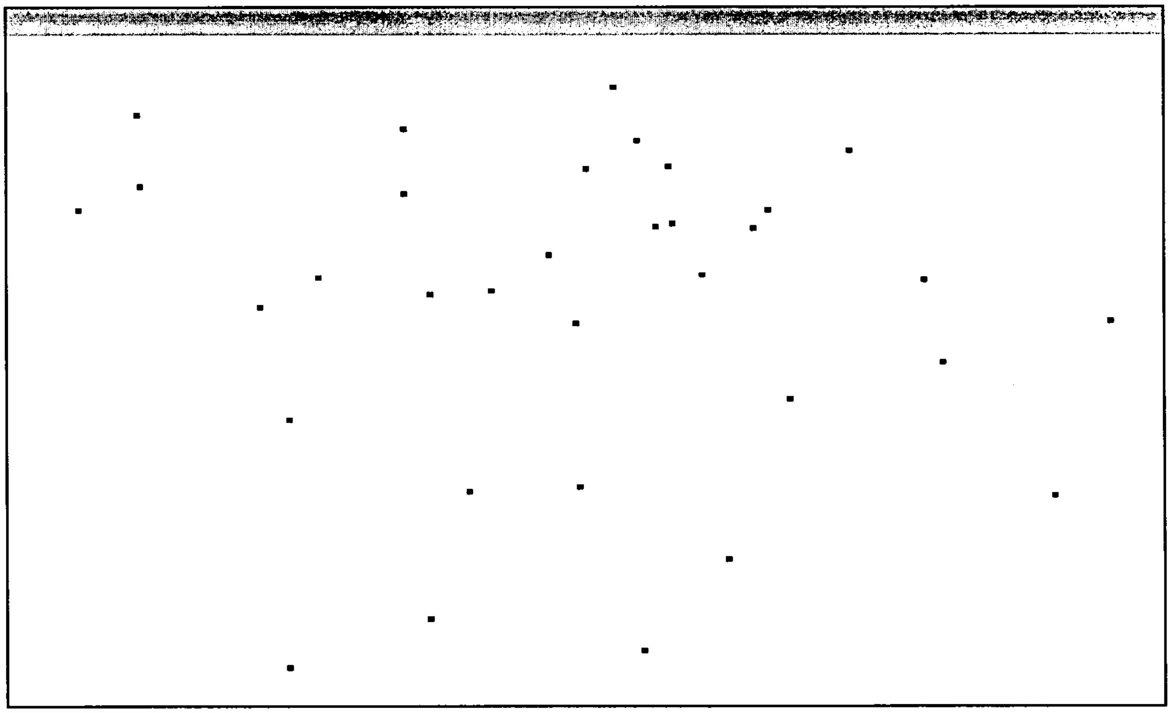


Figure 7: Positions of cluster centers after modified SM

The best stress value $E$ obtained to plot those 34 cluster centers using Sammon mapping method with this modification was 0.107, which is good, according to Table 1.

## 3.4 Comparative Study

Before making our specific choice of MDS to be coupled with SM, we implemented and compared different choices of dimensionality reduction techniques (both linear and non-linear). We also implemented and conducted experiments to compare different combinations of dimensionality reduction methods. Our experiments showed that the combination of metric MDS and SM method gives us the lowest stress value amongst all these methods. Fig. 8 plots the stress value of different dimensionality reduction methods (Stress value represented in percentage) for our example data set of 34 clusters.

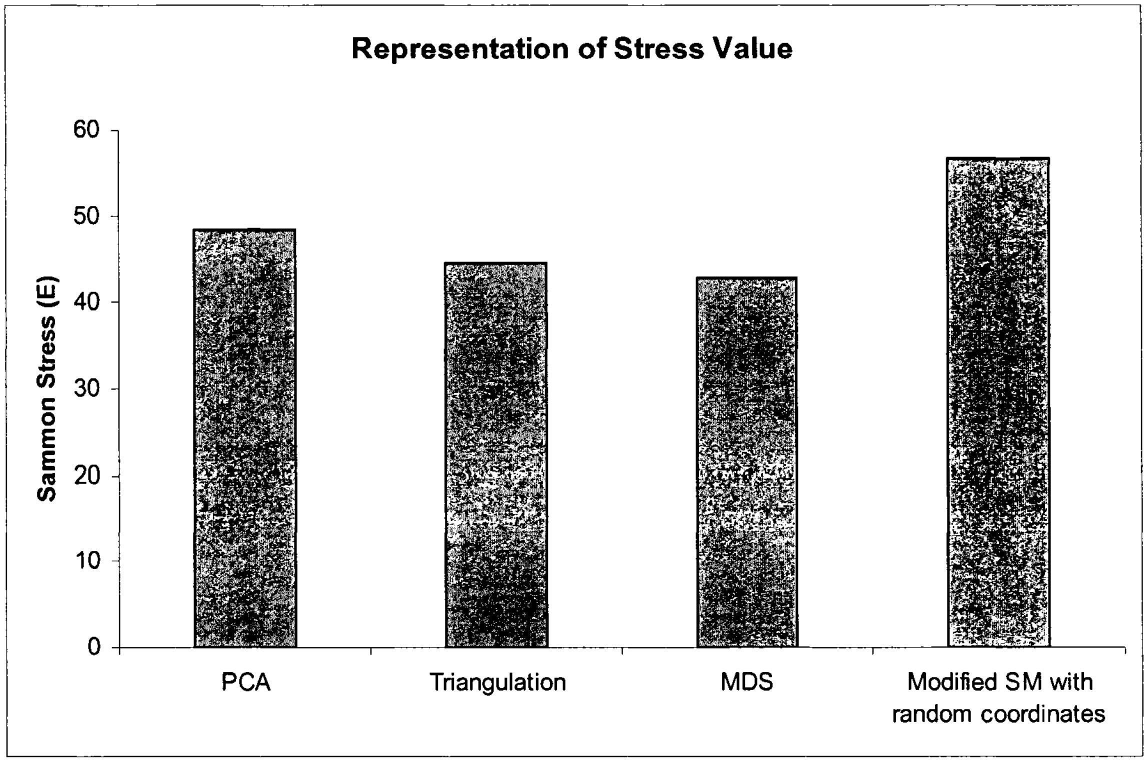


**Figure 8: Stress value comparison of different dimensionality reduction methods**

Table 2 represents the stress values obtained from different combinations of dimensionality reduction methods over *n* iterations with $S_I$ representing the initial stress value and $S_F$ representing the final stress value. For this usage dataset with over 10, 000 sessions, the time taken for combination of the metric MDS and SM method was around 40 minutes whereas the SM combined with the triangulation method took around 35 minutes, and the SM with initial random coordinate values took 57 minutes to find the optimum coordinate positions for the cluster centers in 3D space. Although, the time (40 minutes) taken for combination of SM and the metric MDS method is slightly greater than the time (35 minutes) taken by the combination of triangulation and SM. The former method has a better stress value (0.1007) compared to the latter (0.1436), yielding a layout that better preserves the original similarity relationships and hence considered as our preferred method. In addition, we also repeated these experiments for standard benchmark datasets like IRIS and Wine, as well as a much larger web log usage dataset with 64,529 sessions having 46 cluster centers. We found that for all these datasets as well, the combination of metric MDS and SM has comparatively lower stress value.

**Table 2: Stress value comparison of different combination of methods**

| Method | Iris | | Wine | | 10,000 Sessions | | 64,529 Sessions | |
|---|---|---|---|---|---|---|---|---|
| | $S_I$ | $S_F$ | $S_I$ | $S_F$ | $S_I$ | $S_F$ | $S_I$ | $S_F$ |
| **SM with initial coordinates chosen randomly** | 0.0742 | 0.0103 | 0.0970 | 0.0503 | 0.56782 | 0.19037 | 0.56782 | 0.21137 |
| **SM with initial coordinates chosen by triangulation method** | 0.0340 | 0.0078 | 0.01695 | 0.00680 | 0.44637 | 0.143623 | 0.449 | 0.15562 |
| **SM with initial coordinates chosen by MDS method** | 0.001354 | 0.001354 | 0.001752 | 0.001752 | 0.42839 | 0.1007 | 0.43539 | 0.11 |

# 3.5 Summary

We have combined metric MDS with Sammon Mapping method with a modification to the increment step formula to contain stress value within 1. This combined dimensionality reduction technique yields coordinate values of the cluster center in 3D space, so as to better preserve the original similarity relationships in the data.

1) Conventional Sammon Mapping starts with a choice of random starting points but

according to [Ripley 1996]; this algorithm becomes a poor minimiser by doing so. So, instead, we have used metric MDS algorithm for obtaining the initial points. The reason for selecting metric MDS over other dimensionality reduction method is two fold:

- The metric MDS algorithm is computationally fast, simple and efficient as compared to most of the dimensionality reduction techniques. This can also be seen in our experiments.
- Lower stress value, i.e., a better map of the cluster centers, is obtained by combining SM with metric MDS and this combination overall takes nearly the same time to compute the positions of cluster centers in 3D than other combinations.

2) We observed that if any two points in the projected space have identical positions, then the Sammon stress $E$ will go beyond 1, which is not desirable. To overcome this problem, we tailored the increment formula used in the iterative step of Sammon Mapping algorithm so that the increment does not have divide by near zero problem and the stress value $E$ does not go beyond 1.

# Chapter 4

# Point Cloud Rendering of Fuzzy Clusters

In the previous chapter, we showed how the cluster centers obtained from the RFSC clustering technique are mapped into 3D space, while preserving their similarity relationships to a very good extent. The next step is to use this mapping for imaging the usage data. We have chosen a simple 3D point cloud visual representation for reflecting the web usage patterns discovered by RFSC. In this chapter, we will explain how the remaining user sessions are mapped and displayed with respect to these cluster centers to create a point cloud rendering of all the web usage data. Keeping in mind the scalability requirements we feel that a point cloud rendering, though very simple, is quite adequate, given the huge volume of web usage data, its sparseness, the inherent fuzziness and noise, and the need for dynamic update to handle clickstream data in real time. More specifically, our choice of point cloud rendering in 3D was based on the following observations.

**Choice of 3D over 2D**: The earlier experiments with RFSC carried out on web usage data of different sizes varying from 5,000 to 64,000 sessions had shown that the number of cluster centers returned by RFSC ranged from 20 to 130 [Suryavanshi et al. 2006a]. The reason for choosing 3D space over 2D was because the added dimension of "depth" in 3D could be used to obtain positioning of the cluster centers to better reflect the distance relationships among them. With the current trends in graphics hardware which enable interactive rendering of large 3D objects, it becomes possible to use a simple metaphor of navigating in space and looking around a collection of objects (clouds, in our case) to visually explore the dataset and gain more insight.

**Fuzziness in usage data:** The point cloud can easily handle the fuzziness captured by the clustering technique and this fuzziness can be visually depicted with considerable fidelity.

**Scalability of the Visual Mapping Technique:** Mapping of sessions to represent them as particles in 3D space is a fairly simple computation and the intensity can be varied to reflect the closeness of association with a cluster. Large volumes of data can be handled efficiently, and more importantly, without undue computational overhead.

**Close Integration with Clustering Method:** It was desired to have a simple method integral to clustering, so that one could show the users' navigational behavior in real time to help the web administrator get insight into current trends and interests.

From MDS and the SM methods, we already have the 3D positions of cluster centers. Since RFSC provides every session in the dataset with membership values within each cluster center, we make use of these membership values to assign 3D positions to all other sessions. The method is simple, and is described below.

# 4.1 Mapping Individual Sessions

Every session (other than cluster center and noise, as identified by RFSC) in the dataset is classified into one of following three categories:

i) The first category consists of those sessions having "large" affinity towards only one cluster center. The sessions that belong to this category will have one high membership value, and all other membership values are much lower.

ii) In the second category, sessions will have high affinity towards two cluster centers and much lower membership values towards all other cluster centers.

iii) All other sessions excluding noise sessions are categorized as the third category.

For each session, cluster centers are arranged in the order of their membership values, say, $C_1$, $C_2$, $C_3$, etc. Let $m_1$ be the membership in $C_1$, $m_2$ be the membership in $C_2$, and so on.

Let $a$ be the average distance between clusters i.e.,

$$\mathrm{a} = \frac{\sum^{i>j} d_{ij}}{\frac{n(n-1)}{2}}$$

, where $\mathrm{d_{ij}}$ is the distance between cluster centers $\mathrm{C_i}$ and $\mathrm{C_j}$ and $n$ the number of cluster centers, and let $R$ denote any random 3D vector. In what follows, we present the steps of our rendering algorithm for each category of sessions.

Steps for rendering sessions that belong to the first category:

1) We consider Polar coordinates, i.e., $(\mathrm{r}, \Theta, \Phi)$. Take radius $r = 0.3a\ (1 - m_1)$.

The values $\Theta$ and $\Phi$ are chosen randomly to account for the fuzziness.

2) Then we convert these spherical coordinates to Cartesian coordinates, which gives a position (dx, dy, dz) in 3D space relative to the position of $C_1$.

3) These points are assigned full intensity value in the 3D point cloud model as they

belong specifically to only one cluster center.

Steps for rendering sessions that belong to the second category:

1) The vector difference, $C_2 - C_1$ is multiplied by $(1 - m_1)$ to get a new vector, say $P$.

2) Cross product of $C_2 - C_1$ with random vector $R$ is calculated, resulting in a vector $N$.

3) Vector $N$ is multiplied with $0.5a\ (1\text{-}m_2)$.

4) The desired coordinates are then obtained by adding the vectors $C_1$, $P$, and $N$.

5) Lastly, assign intensity values reduced in proportion to the distance from the cluster centre.

Steps for rendering sessions that belong to the third category:

1) The vector difference, $C_2 - C_1$ is multiplied by $(1 - m_1)$ to get a new vector say $P$.

2) Cross product of $C_2 - C_1$ with a random vector $R$ is calculated, say $N$.

3) Vector $N$ is multiplied with $0.5a\ (1 - m_2)$.

4) The desired coordinates are then obtained by adding the vectors $C_1$, $P$, and $N$.

5) Steps 1 to 4 are repeated for the first and third cluster centers.

6) The weighted average of the two points (step 4) is calculated to obtain the final position of the session.

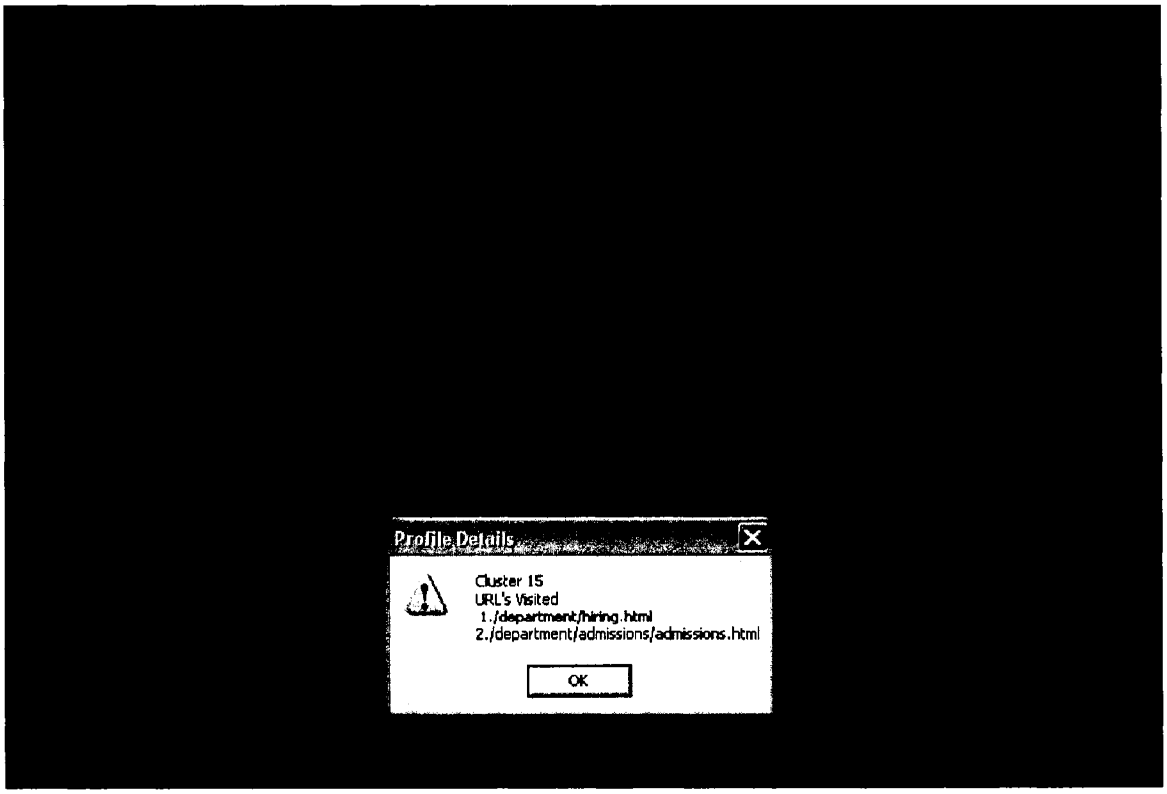
7) Assign intensity values reduced in proportion to the summed distances from the cluster centers.

The above procedure yields a computationally efficient method for assigning 3D positions and color intensity to sessions. Use of dominant membership values results in preserving the inherent relationships much better.

**Figure 9: Point cloud image of fuzzy clustering of web usage sessions**

We have developed a prototype of the proposed technique for visualization of the points, using the C language and OpenGL. Fig. 9 shows a screenshot of the running prototype, and Fig. 10 shows a different view which was obtained by viewing the point cloud model from a different view point and view direction. Fig. 11 illustrates information displayed for a cluster clicked on by the user. This illustration helps provide additional information regarding a cluster like, cluster ID, the list of URL's visited by the cluster, etc. In all these figures, each point represents a user session.

Figure 10: A different view of the point cloud representation of fuzzy clustering of web usage sessions



Cluster 15
URL's Visited
1./department/hiring.html
2./department/admissions/admissions.html

Figure 11: Illustration of a screenshot showing usage profile details when clicking on a cluster

## 4.2 Dual Window Visualization

Visualization exploration is often aided through the use of multiple views. This enables the user to observe the visualization through different forms and to navigate the visualization via different methods [Roberts et al., 2000]. The multiple window visualization has the following advantages:

1. To overcome misinterpretations and provide additional insight of the data.

2. To aid the scientific exploration tasks of relating, coupling and to support the 'drilling down' of information,

3. To provide alternative viewpoints by expressing different user-interpretations of the same information.

To use multiple views effectively, these different modes of views should be, among other things, easily created, automatically coupled to other views, and dynamically manipulated [Erbacher et al., 2000].

In our approach, for interactive analysis, we too have adopted a two-view display coupling the point cloud image (left window) with the website hierarchy (right window), programmatically derived the log records in the form of directory structure. For this, the web log data is used to extract the unique URL's visited and then organized into a directory structure (in 2D space) using XML, and displayed using C++.NET programming language. The directory strucure is simple, easy to use and understand, and is also easily navigable. The directory structure also helps display large amount of data in a single window through the familiar processes of scrolling, collapsing, and expanding operations on subdirectories.

Interactions are permitted in both windows, and the effects are simultaneously made visible in both. For example, one can click on a user profile in the left window and then see the web pages making up that profile in the right window and vice versa. Similarly, as we shall see later, one can interactively move around the nodes in the directory structure and visually see the global impact (in terms of page-clicks costs) this could have on user groups in the left window.

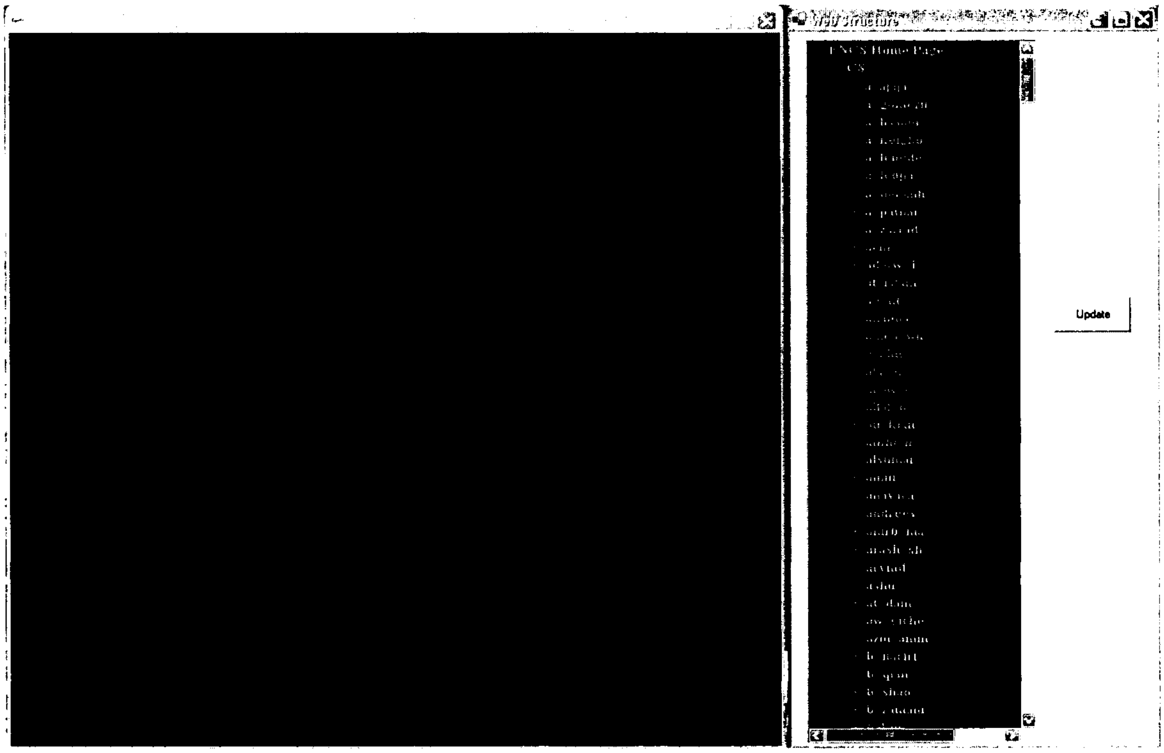


**Figure 12: Screenshot of dual visualization**

Fig. 12 shows a screenshot of the point cloud model as well as the corresponding directory structure. The directory structure we built provides more meaning to the point cloud model by expanding and highlighting the nodes (pages) visited by the cluster center of the clicked cluster. This helps users to get better insight into the popularity of any page with respect to different clusters. Fig. 13 illustrates a screenshot of expanding and highlighting the pages visited in a particular cluster. By clicking on a particular cluster

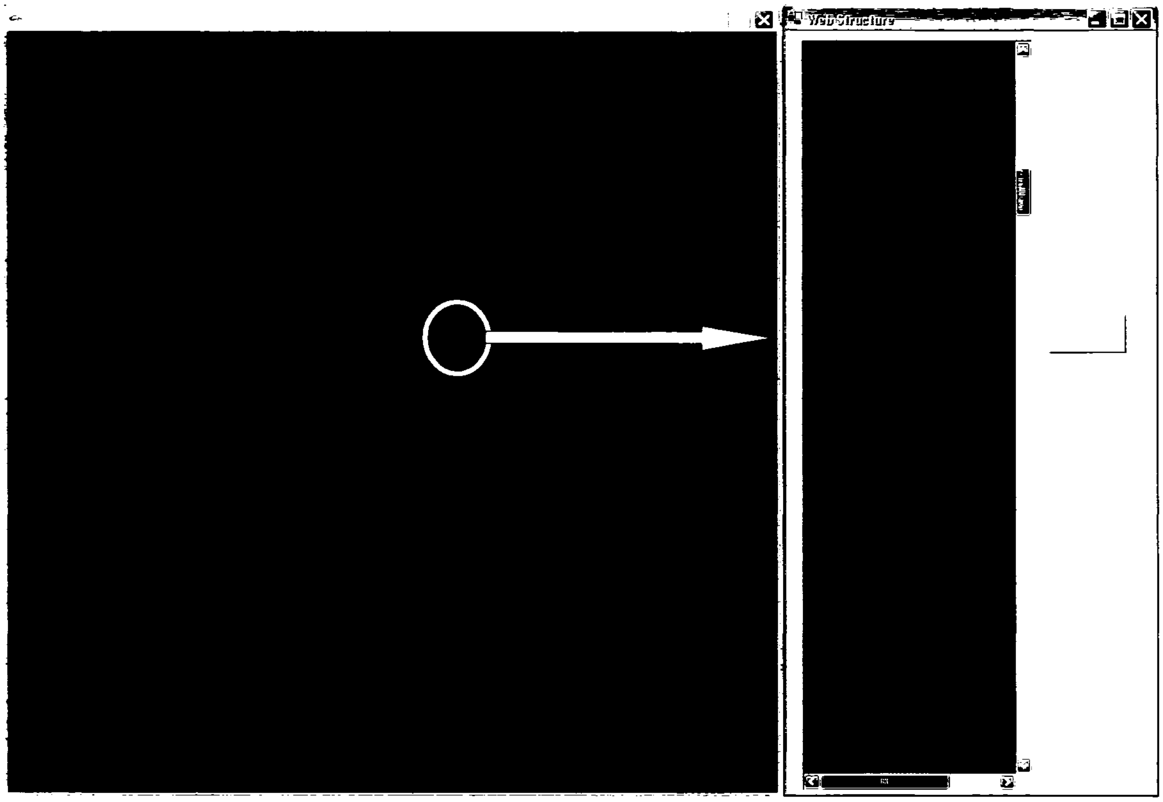


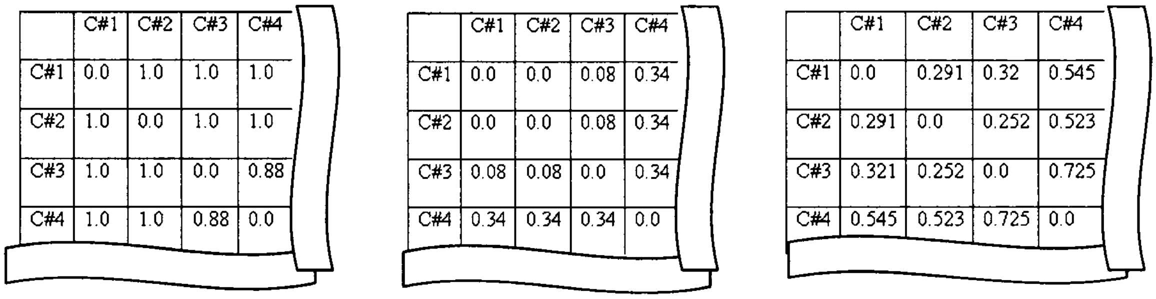
Figure 13: Screenshot illustration of highlighting the pages visited by a particular user group

Figure 14: Screenshot representing popularity of a page clicked

the directory structure expands itself and shows in red the pages visited in that particular cluster. Fig. 14 is a screenshot of the system showing the popularity of the page "comp335" in different sessions. As the user clicks on the page in the directory structure, the sessions that accessed the page are highlighted in red in the point cloud model.

## 4.3 Visualizing Very Large Datasets

Below, we report the results of our experiments with a much larger dataset using the visualization method proposed. Our results shown in Chapter 3 have indicated that the combination of metric MDS and the SM method scales well even for large dataset. The user access log for this particular experiment was recorded during the winter of 2005 from January to May. There were more than 1 million clicks recorded during this period. The total number of user sessions obtained was 64,529 and the number of cluster centers identified by RFSC was 46. Dissimilarity values between cluster centers are extracted from the relational data matrix and used as the input to the dimensionality reduction technique (partially shown in left table in Fig. 15).



| | C#1 | C#2 | C#3 | C#4 |
|---|---|---|---|---|
| C#1 | 0.0 | 1.0 | 1.0 | 1.0 |
| C#2 | 1.0 | 0.0 | 1.0 | 1.0 |
| C#3 | 1.0 | 1.0 | 0.0 | 0.88 |
| C#4 | 1.0 | 1.0 | 0.88 | 0.0 |

| | C#1 | C#2 | C#3 | C#4 |
|---|---|---|---|---|
| C#1 | 0.0 | 0.0 | 0.08 | 0.34 |
| C#2 | 0.0 | 0.0 | 0.08 | 0.34 |
| C#3 | 0.08 | 0.08 | 0.0 | 0.34 |
| C#4 | 0.34 | 0.34 | 0.34 | 0.0 |

| | C#1 | C#2 | C#3 | C#4 |
|---|---|---|---|---|
| C#1 | 0.0 | 0.291 | 0.32 | 0.545 |
| C#2 | 0.291 | 0.0 | 0.252 | 0.523 |
| C#3 | 0.321 | 0.252 | 0.0 | 0.725 |
| C#4 | 0.545 | 0.523 | 0.725 | 0.0 |

**Figure 15: Dissimilarity values: original (left), after MDS (middle), after Sammon Mapping (right)**

As we saw in Table 2 in Chapter 3, after applying metric MDS to this data we obtained Sammon stress value of 0.4349 which is shown in Fig. 16. The Sammon stress value after the Sammon Mapping was 0.11. Fig. 17 shows the cluster centers plotted in 3D (same 2D view as in Fig. 16) after the Sammon Mapping method is applied to it.

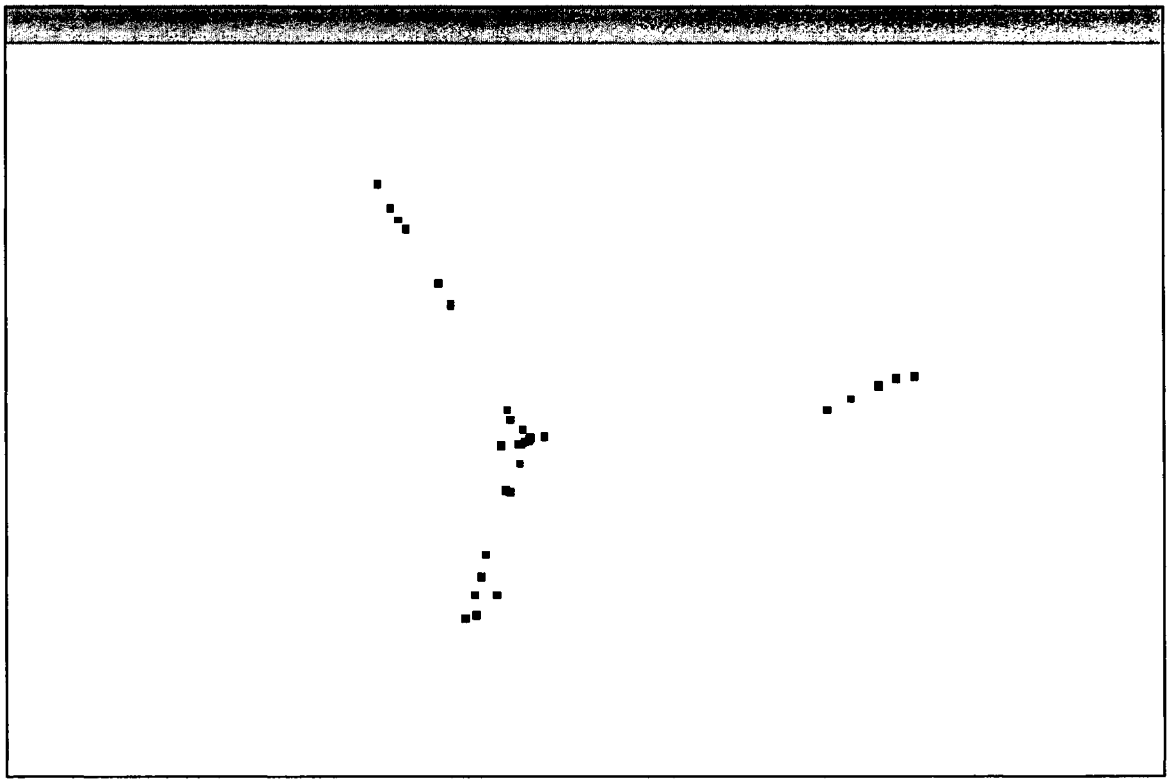


Figure 16: Positions of cluster centers after MDS

Fig. 18 shows a point cloud visual representing this usage data. The procedure described previously in Section 4.1 is used to plot the non cluster center sessions, around 64,483 sessions. Mapping of historic usage data is a preprocessing operation. While the time taken for obtaining the 3D positions for 46 cluster center was around 1 hour, the time taken to assign 3D positions to all other sessions is less than 1 second.

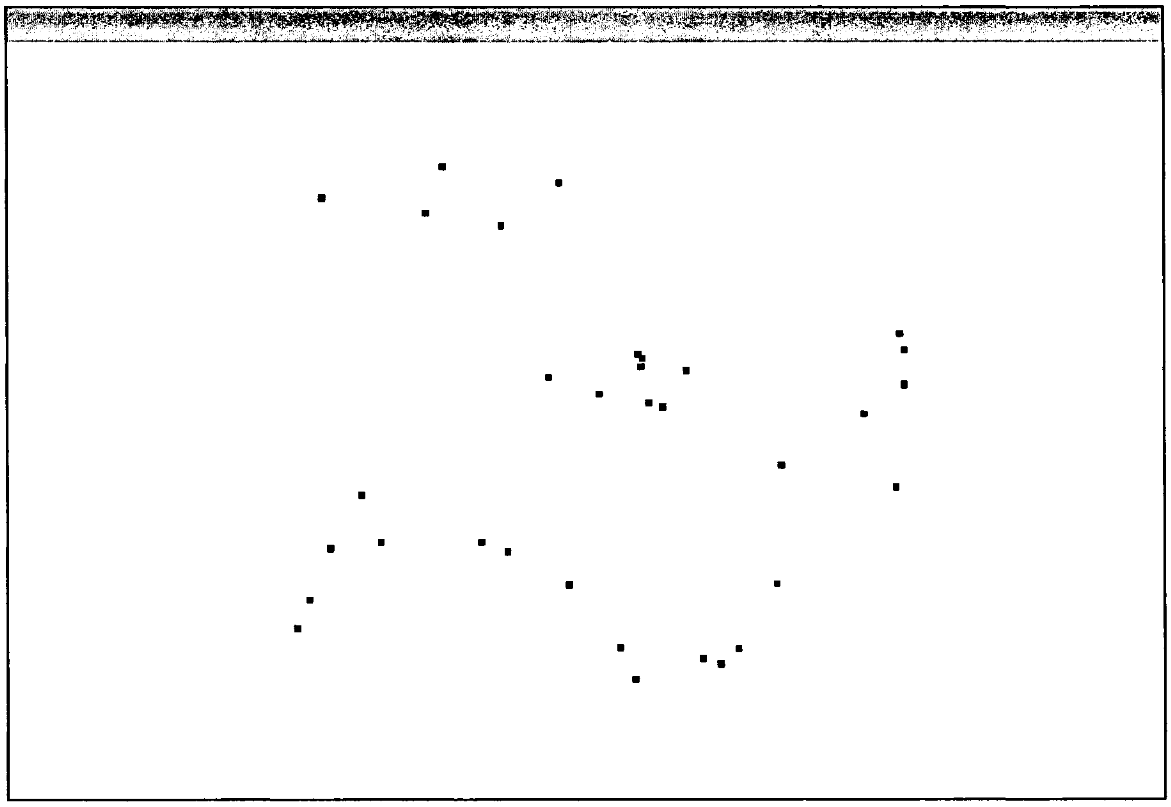


Figure 17: Position of cluster centers after modified SM

Figure 18: Point cloud image of fuzzy clustering of web usage profiles

## 4.4 Updating Point Cloud with New Sessions

Although the clusters obtained from usage data manifest the interests and trends among the users accessing the site at the time the clustering operation was applied, the interests and needs of users change dynamically over time. New logs are continuously created and added as new users continue to access the website. We thus require a usage model update (cluster maintenance) scheme by which these changing trends and patterns can be captured without having to frequently apply this relatively expensive operation of reclustering of large volume of old and new data together.

Cluster maintenance is not a complete alternative to reclustering. By its very definition, cluster maintenance tries to incorporate newly arrived data into the existing model, while maintaining the profiles created earlier from the original set of data. This at best can result in a close approximation to clustering of the complete data, including both old and new sessions. Clearly, reclustering will always give more accurate results and is required to be done periodically. Cluster maintenance however enables us to continuously adapt to the dynamic and changing environment in a much less expensive manner in terms of computation times and resources, and which also allows subsequent maintenance even after reclustering. Thus a "balanced" combination of full data clustering and cluster maintenance is ideally suited for dynamic environments. For this purpose, as the main component for our usage profile maintenance scheme, we make use of incremental RFSC algorithm [Suryavanshi et al. 2005b], which is an extension of RFSC algorithm already described in Chapter 3.

The incremental RFSC algorithm is as follows:

1. Calculate the potential $P_{new}$ of the new object $x_{new}$ to be inserted.

2. Raise the potentials of all existing cluster prototypes;

   Di, new = dissimilarity between $x_{new}$ and $i^{th}$ cluster center

   $P_i = P_i + e^{-\alpha Di^2, new}$

3. $d_{min}$ = minimum of the dissimilarity values between $x_{new}$ and all the previously found cluster centers.

```
        for i=1 to C do
            if ($P_{new}$ > Pi) then
                if ($P_{new}$ > $\bar{\in}$P1), then //case (iii)
                    I += $d_{min}$
                else if ($P_{new} \geq \underline{\in}$P1) then
                        if ($d_{min}/\gamma$) + ($P_{new}$/P1) $\geq$ 1, then //case (iii)
                        $I$ += $d_{min}$
                        end if
                 end if
            else
                $P_i = P_i + e^{-\alpha Di^2, new}$
        If ($P_{new}$ < 0), then goto step 5; //case (i) -- no further comparisons required
            end if
        end for
4.  if (($P_{new}$ >0) and (Case (iii) is not true))
      //check if $x_{new}$ can become a new cluster center
        if ($P_{new} \geq \underline{\in}$P1), then
            if ($d_{min}/\gamma$) + ($P_{new}/P_1$) 1, then
                //case (ii) is true; $x_{new}$ is the new cluster center
                   C = C+1; //increment cluster count
                   PC = $P_{new}$;
              Calculate the membership of NU old objects with respect to this new cluster;
            end if
        end if
    end if

5. Increment Nu by 1 as a new object is being added;
   Calculate membership of $x_{new}$ with the C clusters.
End algorithm;
```

When a new session is added, this algorithm either makes it a new cluster center or assigns fuzzy membership values to existing clusters. As new clusters are discovered, it becomes essential to add the new clusters into the point cloud without changing the position of the existing clusters, to avoid any visual confusion to the viewer. We have devised a method for plotting new clusters without having to run the MDS and SM methods for the whole data again. For this, we first obtain an initial coordinate value by calculating the distances between the new cluster center and the existing cluster centers. Then we use the SM method with the modification described in chapter 3 to decrease the error in distance between the newly found cluster center and the existing cluster centers. When the Sammon stress exceeds a pre-defined threshold, we need to perform the MDS and SM computations for the entire data set. We have experimented by removing the sessions belonging to a cluster and found that Incremental RFSC does add that new cluster and the visualization method assigns a new 3D position to the cluster center, sufficiently distinguishable from the rest. If the newly arrived session is a non cluster session then we need to add them to the point cloud model. In order to render a new session into the point cloud, we first compute its distances to the cluster centers and membership values using the equations given in chapter 2. Using these membership values, we classify this session into one of the three categories or as noise, and assign 3D coordinate values. For a noise session, we assign 3D coordinates by randomly distributing them in the 3D space containing all the cluster centers.

For experimental purposes, out of the 34 cluster centers, we plotted 25 cluster centers and the remaining 9 were added incrementally as it was found by the incremental RFSC. The Sammon stress value for adding these cluster centers varied from 10% for the 26th cluster center to 16% for the 34th cluster center. The initial 26 cluster centers are presented in Fig. 19 in red and the remaining 9 cluster centers are shown in green.

## 4.5 Discussion

We make the following observations from the above explained visualization method:

a) We have plotted the 10,000 sessions in 3D space by making use of the fuzzy membership values obtained from the RFSC algorithm we used. The time taken to obtain the 3D position for the non cluster center sessions was less than a minute. This is much better compared to applying dimensionality reduction technique to all the 10,000 sessions because dimensionality reduction for such large data sets is computationally prohibitive, as it involves Eigen value analysis using a matrix of this size. In particular,

**Figure 19: Incremental addition of cluster centers**

convergence can be a major problem as there are many near equal dissimilarity values in such datasets.

b) As new clusters are found by the incremental RFSC algorithm, we efficiently find the position of only the newly found clusters with respect to already mapped clusters. This helps users to avoid confusion in understanding the model as new clusters are obtained. In the next chapter, we shall describe how this method of mapping new clusters into the existing point cloud model is used for animation of clickstream data so as to enable users to get better insight into the most recent trends and interests in real time.

c) The two-view visualization also helps gain insight into details of the pages visited in a particular cluster, or to be more precise, the pages visited in the sessions in a particular cluster. Also, by highlighting the sessions that accessed a particular page, it helps to visually represent the popularity of a page. Another important purpose of the two-view visualization is to provide visual responses to the user for "if-then" types of queries made with respect to changes in the page hierarchy layout. We elaborate on this point later in Chapter 5.

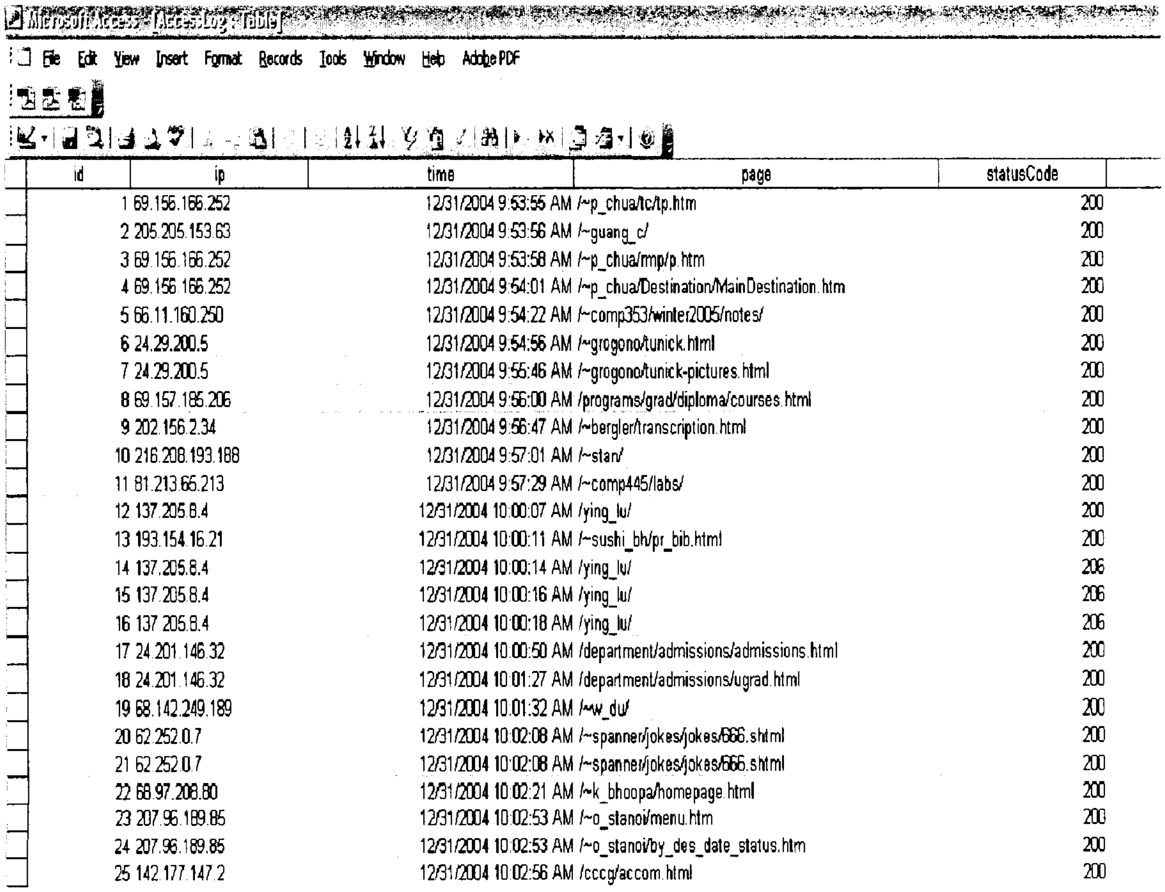
# Chapter 5

# Visual Analysis Examples

In the last two chapters, we described our methods for efficiently mapping and displaying web log data on 3D space using a two-view display. In this chapter, we will describe a few specific examples of visual analysis techniques that are of importance to web administrators. These techniques make effective use of the global context provided by the visual presentation of the historic web usage along with the page hierarchy structure of the website.

## 5.1 Clickstream Data Animation

As discussed in Chapter 2, clickstream data is the sequence of page clicks logged by the web server as different users move from page to page and click on items within a web site. This information is stored in the log file by the web server. Web site designers can use clickstream data to improve users' experiences with a site. Fig. 20 shows a small sample of typical clickstream data from our department's web server. The clickstream data includes fields such as, the IP address which can be used to identify the different visitors of the site, the time indicating when the request for a particular page was made, the URL of the web page visited, status code indicating whether the request was successful or not, etc. Of those, currently our interest is the IP address, the time and the page visited. Analysis of all the recent pages visited can provide insight into current and most recent trends and interests of users visiting the website. We have devised an elegant technique for visual analysis of such trends and interests. As page clicks stream in, active sessions are updated to reflect new pages visited. Any changed active session is mapped into the same 3D space which shows historic usage and a highlight is added to the cloud that is affected by this changed session. Over time, these animated highlights provide adequate visual cues for identifying currently "hot" user profiles. For example, if a specific user profile is "hot" and has most of the pages (products) of current interest to users, this will result in the corresponding cloud being the one that has the most highlights.



| id | ip | time | page | statusCode |
|---|---|---|---|---|
| 1 | 69.156.166.252 | 12/31/2004 9:53:55 AM | /~p_chua/tc/tp.htm | 200 |
| 2 | 205.205.153.63 | 12/31/2004 9:53:56 AM | /~guang_c/ | 200 |
| 3 | 69.156.166.252 | 12/31/2004 9:53:58 AM | /~p_chua/mp/p.htm | 200 |
| 4 | 69.156.166.252 | 12/31/2004 9:54:01 AM | /~p_chua/Destination/MainDestination.htm | 200 |
| 5 | 66.11.160.250 | 12/31/2004 9:54:22 AM | /~comp353/winter2005/notes/ | 200 |
| 6 | 24.29.200.5 | 12/31/2004 9:54:56 AM | /~grogono/tunick.html | 200 |
| 7 | 24.29.200.5 | 12/31/2004 9:55:46 AM | /~grogono/tunick-pictures.html | 200 |
| 8 | 69.157.185.206 | 12/31/2004 9:56:00 AM | /programs/grad/diploma/courses.html | 200 |
| 9 | 202.156.2.34 | 12/31/2004 9:56:47 AM | /~bergler/transcription.html | 200 |
| 10 | 216.208.193.188 | 12/31/2004 9:57:01 AM | /~stan/ | 200 |
| 11 | 81.213.66.213 | 12/31/2004 9:57:29 AM | /~comp445/labs/ | 200 |
| 12 | 137.205.8.4 | 12/31/2004 10:00:07 AM | /ying_lu/ | 200 |
| 13 | 193.154.16.21 | 12/31/2004 10:00:11 AM | /~sushi_bh/pr_bib.html | 200 |
| 14 | 137.205.8.4 | 12/31/2004 10:00:14 AM | /ying_lu/ | 206 |
| 15 | 137.205.8.4 | 12/31/2004 10:00:16 AM | /ying_lu/ | 206 |
| 16 | 137.205.8.4 | 12/31/2004 10:00:18 AM | /ying_lu/ | 206 |
| 17 | 24.201.146.32 | 12/31/2004 10:00:50 AM | /department/admissions/admissions.html | 200 |
| 18 | 24.201.146.32 | 12/31/2004 10:01:27 AM | /department/admissions/ugrad.html | 200 |
| 19 | 68.142.249.189 | 12/31/2004 10:01:32 AM | /~w_du/ | 200 |
| 20 | 62.252.0.7 | 12/31/2004 10:02:08 AM | /~spanner/jokes/jokes/666.shtml | 200 |
| 21 | 62.252.0.7 | 12/31/2004 10:02:08 AM | /~spanner/jokes/jokes/666.shtml | 200 |
| 22 | 68.97.208.80 | 12/31/2004 10:02:21 AM | /~k_bhoopa/homepage.html | 200 |
| 23 | 207.96.189.85 | 12/31/2004 10:02:53 AM | /~o_stanoi/menu.htm | 200 |
| 24 | 207.96.189.85 | 12/31/2004 10:02:53 AM | /~o_stanoi/by_des_date_status.htm | 200 |
| 25 | 142.177.147.2 | 12/31/2004 10:02:56 AM | /cccg/accom.html | 200 |

**Figure 20: Clickstream data recorded from a web server**

In order to test out the above technique, we carried out our experiments with the same web log data. To simulate clickstream data in real time, we partitioned the web log data into two sequences, the first sequence containing all but the last 5000 records and the second containing the last 5000 click records. The latter sequence is treated as the clickstream and used as the input for clickstream animation. The sequence of clickstream data can be viewed conceptually as a sequence of active_session and URL list pairs. For example if the stream consisted of the following clicks (<$s_1$, $u_{11}$>, <$s_1$, $u_{12}$>, <$s_2$, $u_{21}$>, <$s_1$, $u_{13}$>, <$s_3$, $u_{13}$>, <$s_2$, $u_{22}$>, ... then the sequence is of the form:

{($s_1$, <$u_{11}$>)}, {($s_1$, <$u_{11}$, $u_{12}$>)}, {($s_2$, <$u_{21}$>)}, {($s_1$, <$u_{11}$, $u_{12}$, $u_{13}$>)}, {($s_3$, <$u_{31}$>)}, {($s_2$, <$u_{21}$, $u_{22}$>)}, and so on.

As new click data is obtained, an active session may change its list of URLs. In order to map this changed session into the 3D space, we need to calculate the membership value with the cluster centers. However, in order to calculate the membership value we first need to calculate the similarity of this changed active session with every cluster center.

To determine similarity of any two sessions $s_k$ and $s_l$, two measures are used. The first measure is cosine which does not consider the site structure, and is defined as follows:

$$S_{1,kl} = \sum_{i=1}^{M} s_{ki} s_{li} \Bigg/ \sqrt{\left( \sum_{i=1}^{M} s_{ki} \sum_{i=1}^{M} s_{li} \right)}$$

The second similarity measure, defined below, incorporates syntactic URL similarity:

$$S_{2,kl} = \sum_{i=1}^{M} \sum_{j=1}^{M} s_{ki} s_{lj} S_u(i,j) \Bigg/ \left( \sum_{i=1}^{M} s_{ki} \sum_{j=1}^{M} s_{lj} \right)$$

The similarity between any pair of sessions $s_k$ and $s_l$ is a value in [0, 1], defined as:

$$S_{kl} = max(S_{1,kl}, S_{2,kl}) .$$

Once we have the similarity, the membership value is calculated as explained previously in Chapter 2. Using these membership values, we categorize the newly changed active session by following the procedure explained earlier in Section 4.1. Instead of plotting the newly arrived session, we keep a count for every cluster center which indicates the number of newly arrived sessions belonging to it. The display is updated every *n* units of time. The brightness of a cluster is directly proportional to the number of active sessions belonging to that cluster. Fig. 21 shows the popularity of three specific clusters (encircled) relative to the remaining clusters at that particular point of time. For comparison Fig. 22 shows the same point cloud without the highlights resulting from clickstream data animation.

With the immense growth of the Internet usage, websites are being developed in an uncontrolled and ad-hoc manner. Thus, a critical task for a website administrator is to use enumerable metrics in order to identify substructures of the site that are objectively popular. Identifying hot spots is one way to achieve the above mentioned goal. Hot spots are the part of the website that receives significant traffic. Identifying these parts is important for website administrators to decide on performing local structural changes to improve services and increase users' satisfactions. Clearly, by animating the dynamic clickstream data, one can visually identify current hot spots in the website.

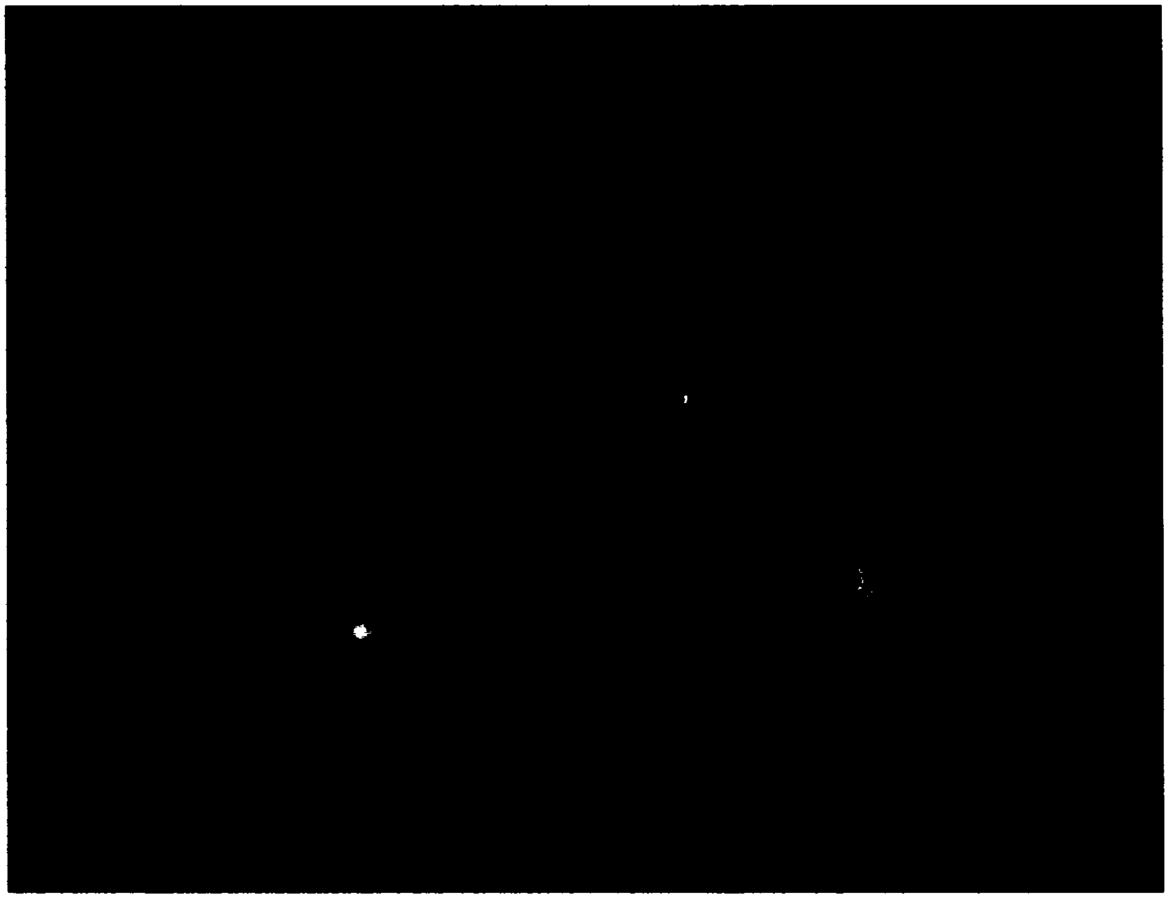


Figure 21: Representation of clickstream visualization

## 5.2 Analyzing the Impact of Website Layout Changes

As the interests of web users' change over time, a web site must change itself accordingly to best serve its users. In other words, web sites should be adaptive. An adaptive web site has been defined as a web site that semi-automatically improves its organization and presentation by learning from visitor access patterns [Perkowitz and Etzioni 1997]. To be more specific, the aim is to make a web site that provides its users the information they want with less clicks. This reduces the effort on the user's side. By analyzing the usage of

**Figure 22: Initial point cloud model without highlights**

a web site and the structure of the web site, modifications to the web site structure are found to accommodate changes in access patterns of its users. These modifications will be suggested to the web administrator for consideration and implementation. The general idea of rearranging is to cut down the number of intermediate index pages a user has to go through. To achieve this, the frequently accessed pages or link to such pages could be placed closer to the home page, while pages that are accessed least frequently could be moved lower in the web hierarchy.

We have formulated a simple model which helps to analyze the impact of page rearrangement in the structure. Our model implies that transition to the next page is dependent only on the current page. This assumption correctly captures the situation whereby a user selects a link on the current page to go to the next in the browsing sequence. However, the model does not take into account for external random jumps by entering a fresh URL into the browser window. Our model has the following components:

1) Popularity of a page (URL): Popularity of each page is computed by finding the ratio of the sum of the page weights across sessions to the total number of sessions in that cluster. That is,

Popularity $P$ of page $j$ in cluster $i$ is defined as the number of sessions belonging to cluster $i$ that have visited $j$ divided by the total number of sessions belonging to cluster $i$.

2) Cost factor: The cost factor of accessing a web page $M$ is based on the level at which the page is placed, the number of links it includes and its popularity value. The formulation of cost factor is given as follows:

$$\text{Cost of accessing page } M = \sum_{i=1}^{n} P \times l \times \log(M_i)$$

where, n is the number of links present in *M,*

$P$ is the popularity of page *M, and*

$l$ is the level of page *M in the web hierarchy,* from the root

This equation is based on the fact that access to a desired web page in a web site involves two things: the level at which the page is located from the root (assuming that user begins from the root), and the number of links the page contains. Once the cost of every page is determined, we can calculate the average cost of a cluster.

Using this access cost, the visualization helps the web administrator to decide which pages can be rearranged in the web site, described as follows:

The directory structure provides an option to rearrange the pages within the structure. As the pages are updated, the point cloud model changes its color depending on the new average cost of the cluster calculated dynamically which represents the effect of moving the pages. We compare the average cost after modification (final cost) with the initial average cost. Every time the final average cost of cluster goes above the initial value, the cluster is shown in red color indicating that the user group associated with this cluster is negatively affected by the move. On the other hand, if the average cost falls below the initial value, then the cluster is shown in green indicating that the user group associated is positively affected by the change. Additionally, a graph is also displayed in another window showing the initial and the final average cost of rearranging the pages.

Fig. 23 and Fig. 24 show the effect of rearranging some of the web pages. The impact can be seen in the point cloud model as well as the bar chart which gives the exact cost of moving the pages. By looking at the point cloud model, the web administrator can obtain an overall idea of the impact on different user groups accessing the web site. That is, the administrator can identify the number of clusters which are going to be affected. For example, in Fig. 23 we can observe that the particular move for this example is not affordable because majority of the clusters are affected negatively. Whereas in Fig. 24, the move is very good as it affects majority of the clusters positively. So, using this visualization, web administrator can make cost-effective decisions of rearranging the pages of the web site efficiently.

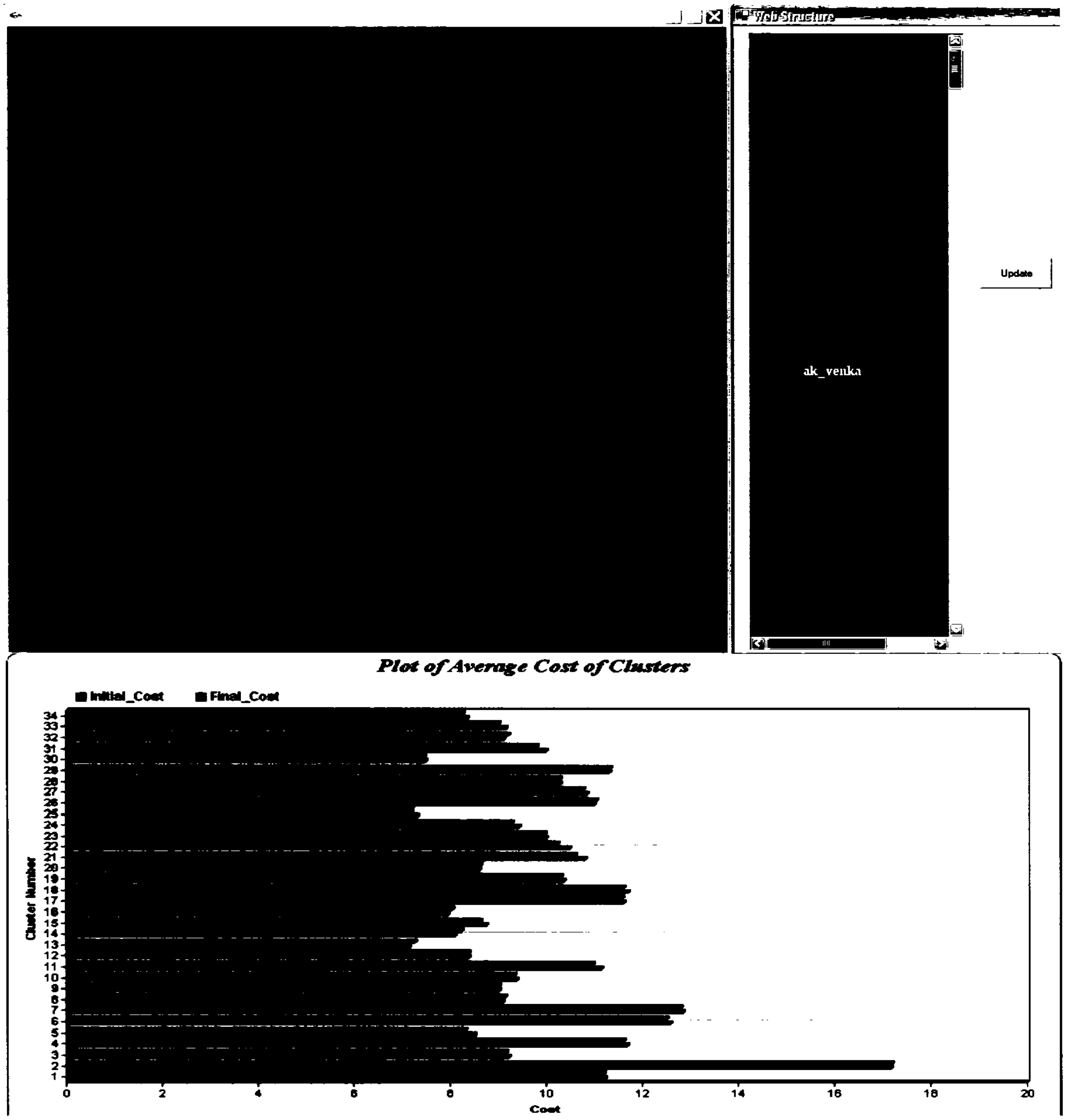


Figure 23: Screenshot representing negative impact of moving pages

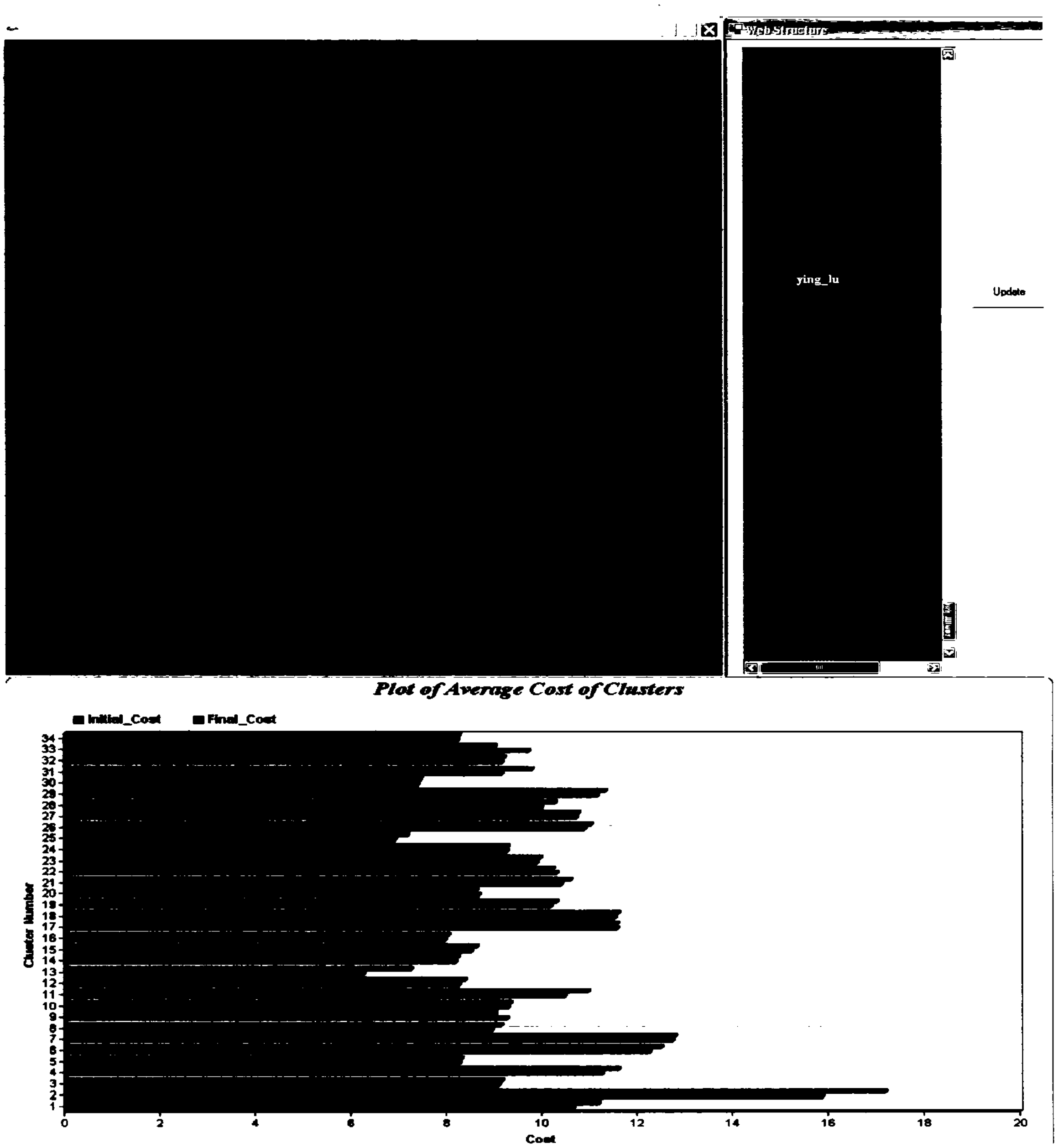


**Figure 24: Screenshot representing positive impact of moving pages**

## 5.3 Analyzing Noise in Usage Model

Every usage model has some noise; usage data is inherently noisy in nature. This is because the users browse websites for different purposes, sometimes even without specific purpose. The RFSC membership function we used is Gaussian in nature and the membership is calculated for every cluster using this function. With respect to individual clusters, the objects along the asymptotes of this Gaussian function are noise for that cluster. Noise objects with respect to the entire dataset are therefore essentially defined as objects which lie along the asymptotes of the Gaussian function for all clusters. An example of a noise session is given below in Table 3.

**Table 3: Example of a Noise session**

| Profile # | Usage Profile | Description |
|---|---|---|
| Noise | /~ni_wang/openglspace/index.htm - 0.031<br>/~ni_wang/openglspace/welcome.html - 0.030<br>/db/db/main.html - 0.022<br>/~tktran/welcome.html - 0.019<br>/~lzhang/ - 0.018 | This is noise profile consisting of sessions with low membership to all the clusters. We see that it's a mix of unrelated pages with top five pages having very low popularity. |

As new sessions arrive and get incorporated into the usage model in an incremental fashion as described in Chapter 4, we categorize them into either new cluster center, or a new non cluster center, or a noise session. For a noise session, we assign 3D coordinates by randomly distributing them in the 3D space containing all the cluster centers. As the number of noise sessions increases, the point cloud model becomes fuzzier indicating the need for reclustering.

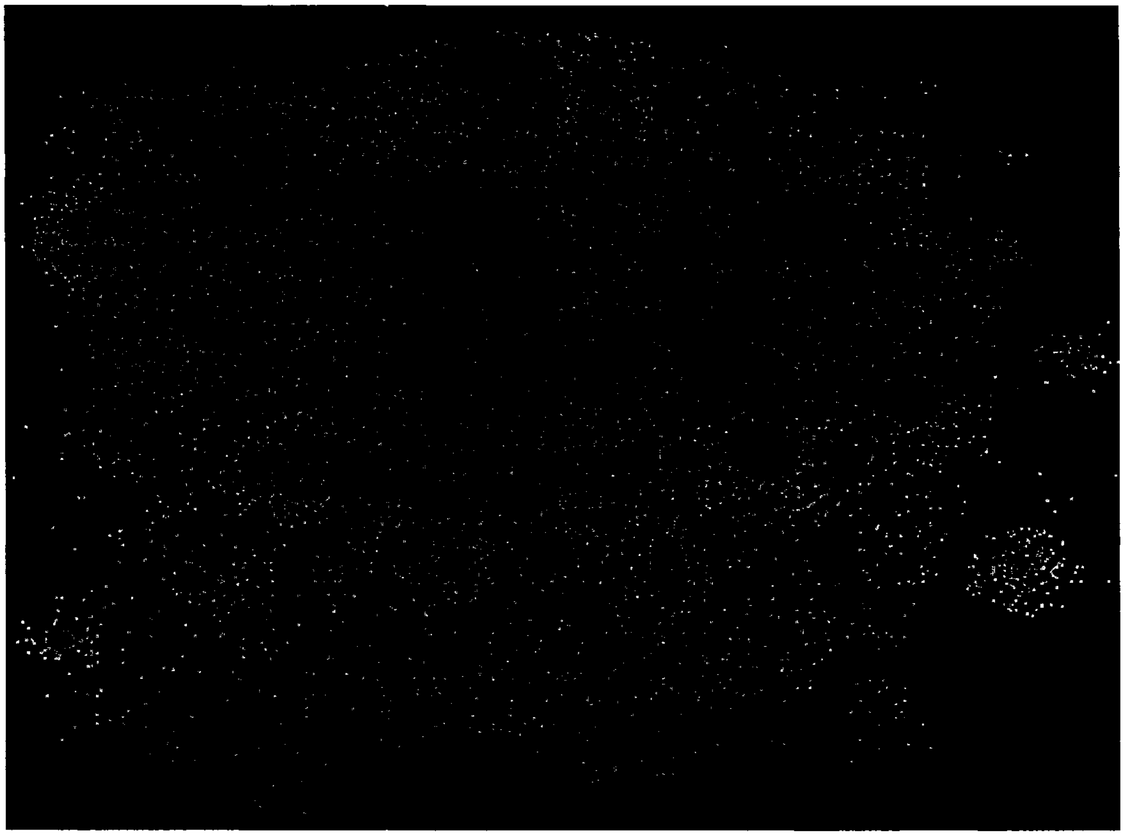


Figure 25: View of Point Cloud with 15% (simulated) noise

Clustering of web usage data is an expensive process and must be delayed as much as possible. There is no universally accepted formula for making a decision on when to recluster the data. Decision of reclustering is subjective and is solely dependent on the administrator. Displaying noise sessions in the 3D point cloud model helps the administrator to visually make a decision on when to recluster the web log data.

Fig. 25 and Fig. 26 show two views of the same dataset with simulated noise sessions added for illustrational purposes. It is clear from the figure that the cluster pattern is lost when large number of noise sessions is added to the point cloud model.

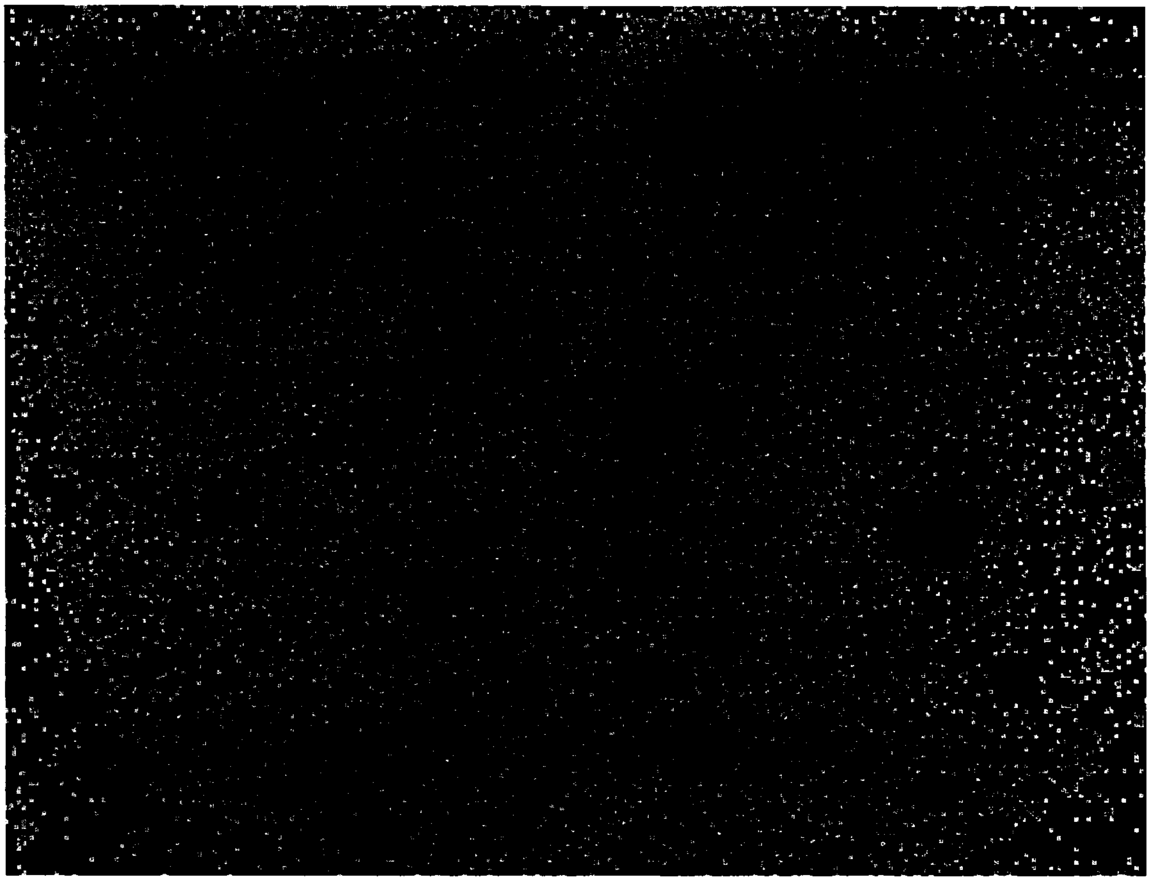


**Figure 26: Zoomed-in view of point cloud with 40% (simulated) noise**

# 5.4 Discussion

We make the following observations from the above applications of visualization:

1. By identifying the hot spots in the web log data using the technique of clickstream animation, the web administrator can consider to rearrange the pages that are hot spots. Also, he/she can better understand the usage patterns at any given point of time.

2. By calculating the cost of rearranging the web pages and visualizing it in point cloud model, the web administrator can assess the affect from users' perspective. For example, the web administrator can change the structure by editing one or more links, and the system would react by illustrating its impact on the point cloud model.

3. Plotting of noise sessions in point cloud model helps the web administrator to make a decision on when to perform reclustering. This is very important because the web log data is very large and the time needed to cluster this data is enormous. So it is desired to postpone the reclustering as much as possible.

# Chapter 6

# Conclusions and Future Work

As the Internet matures as a major medium for doing business, companies are trying to move beyond the basics of business transactions into a deeper level of understanding of users' interaction with their web site. At any given website, the flow of users' visits represents a valuable source of information for web administrators. However, the identification and interpretation of web usage patterns is not an easy task. The sheer volume and complexity of the browsing data captured in the website server logs makes understanding users a difficult and time-consuming task. Visualization techniques can aid in the process of understanding the interactions between users of a web site and the web site itself.

In this thesis we have proposed a new approach for visualizing web usage by integrating the usage modeling and rendering processes. Fuzzy clustering technique is first applied to the historic log data organized into sessions. The results obtained in the form of fuzzy cluster centers along with the membership value of every session with the cluster centers are imaged in the form of a simple point cloud. As new user data is gathered, the incremental fuzzy clustering algorithm is used to update the point cloud rendering. A major challenge is to have a rendering process that scales to large volumes of high dimensionality data, such as web usage data. We have devised an efficient method for mapping usage data into 3D space. This is achieved by first mapping the cluster centers in a preprocess using suitable dimensionality reduction techniques followed by assigning

3D positions for all other sessions through simple computations that maintain their behavior with respect to cluster centers as reflected in the membership values. This clever use of membership values considerably reduces the computational effort required for mapping such large volume data. Once the coordinates are assigned, web usage data is rendered as clouds of points, with each cloud representative of a fuzzy cluster. Based on a number of experiments with different dimensionality techniques, we have chosen a combination of metric Multidimensional Scaling and a Sammon Mapping as giving us the mapping that best preserves the similarity present amongst the cluster centers. The main advantage of integrating the fuzzy clustering algorithms into the rendering of web log data is that we contain the computational complexity. While this has the computational simplicity of the triangulation method, the use of distinct membership value differences for rendering yields visuals that preserve the fidelity of original relationships among the entities much better.

We have also shown that a new session can be incrementally added to the visualization quite easily while maintaining the fidelity of its relationship to the usage model, thus enabling the web administrator to decide on the fly as to when the web usage data needs to be reclustered. As the browsing behavior of users changes dynamically, we need to show the changes so that the administrator can identify the most recent trends in user interests. For this, we have devised the technique of clickstream data animation which will show "hot spots in the point cloud", if present in the newly arriving data. We believe that identifying these parts facilitates website administrators on deciding how to perform local structural changes optimally.

From a user interaction aspect, we have implemented the two-view display showing the usage model in one window for providing the global context to any queries, and the detail of the page hierarchy in another window using the directory structure format. The ability to interact with either window and see the impact simultaneously in both makes interaction easier by providing the context + detail view at all times. For example, as the user clicks on the desired cluster, the directory structure expands and displays the pages visited in that particular cluster. Similarly, the impact of any change in the page hierarchy layout can be seen in the usage model. For this, we have proposed a cost evaluation method to calculate the average cost factor for the user group associated with the cluster, as a function of the popularity of the pages visited in that cluster. The main advantage of our cost model is that it helps the administrator to interactively assess the impact of moving a particular page or a group of pages in the web site on different groups of users. Through appropriate experiments on large real-life datasets as described in Chapters 4 and 5 we have demonstrated that our proposed approach works well and enables interactive visual analysis. We are not aware of any other work that uses the historic usage model for providing the global context for visual analysis of web usage and can deal with such large data with large dimensions.

There are many different avenues to explore this work further. A few of these are listed below.

- We would like to try this visualization on commercial web site usage data for example, Amazon®, eBay®, etc., and we would like to see users behavior using our visualization method.
- In our work, we have not considered the amount of time each user has spent on a particular page. Time information can be very useful to identify the popularity of

a page, in highlighting the hot spots of dynamic users or in detecting noise/transition/content pages.

- Also the formula we proposed for calculating the cost factor to identify the hot spots is rather simplistic. It should be possible to reformulate it to take into consideration other factors such as time needed for manipulating the mouse through the hierarchy.
- For visualizing the pages accessed by the users, we used directory structure as our visualization. We would like to see how other visualization methods like cone tree [Herman and Marshall 2000] or 3D hyperbolic graphs [Munzner 1998] can be used more effectively for interacting with the detailed structure.

# Bibliography

**[Andrews, 1995]** Andrews, K. *Visualizing cyberspace: information visualization in the harmony internet browser*, 1st IEEE Symposium on Information Visualization (Info Vis-95), IEEE Computer Society, Silver Spring, pp. 90-96, 1995.

**[Barr et al., 1996]** Barr, A. J. and Ray, J. L. *Control of an Active Suspension Using Fuzzy Logic,* Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, Vol. 1, IEEE, pp. 42-48, 1996.

**[Bezdek 1981]** Bezdek, J.C. *Pattern recognition with fuzzy objective function algorithms, Plenum*, New York, 1981.

**[Brainerd and Becker, 2001]** Brainerd, J., Becker, B. *Case Study: E-commerce Clickstream Visualization*, In Proc. of the IEEE Symp. On Information Visualization, pp. 153-156, 2001.

**[Chen, 1999]** Chen, C. *Information Visualization and Virtual Environments*, Springer-Verlag, ISBN 1-85233-136-4, 1999.

**[Chi, 1994]** Chi, Ed H. *WebSpace Visualizations*, Proc. 2nd Int'l World Wide Web Consortium (W3C), Oct. 1994; IEEE Internet Computing, Vol. 6, Issue 2, pp.64-71, 1994.

**[Chi, 2002]** Chi, Ed H. *Improving Web Usability Through Visualization*, IEEE Internet Computing, pp. 64-71, 2002.

**[Chiu, 1994]** Chiu, S.L. *Fuzzy model identification based on cluster estimation*, J. of Intelligent and Fuzzy Systems, 2(3), 1994.

**[Cooley et al., 1999]** Cooley, R., Mobasher, B., Srivastava, J. *Data Preparation for Mining World Wide Web Browsing Patterns*, J. of Knowledge and Information Systems, 1, pp. 1-27, 1999.

**[Cooley, 2000]** Cooley, R. *Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data*, PhD thesis, University of Minnesota, 2000.

**[Corsini et al. 2004]** Corsini, P., Lazzerini, B., Marcelloni, F. *A new fuzzy relational clustering algorithm based on fuzzy C-means algorithm*, Soft Computing, Springer-Verlag, 2004

**[Cox and Cox, 2001]** Cox, T.F. Cox, M.A.A., Multidimensional Scaling, 2nd Edition, Chapman & Hall/CRC, 2001.

**[Cugini and Scholtz, 1999]** Cugini, J., Scholtz, J. *VISVIP: 3D Visualization of Paths through Web Sites*, In Proceedings of International Workshop on Web-Based Information Visualization (WebVis'99). Florence, Italy. pp. 259-263, 1999.

**[Erbacher et al., 2000]** Erbacher, R.F., Chen, P.C., Roberts, J.C. *Multiple view and multiform visualization*, In Proc. SPIE Vol. 3960, Visual Data Exploration and Analysis VII, pp. 176-185, 2000.

**[Etgen and Cantor, 1999]** Etgen, M., Cantor, J. *What Does Getting WET (Web Event-Logging Tool) Mean for Web Usability?,* In Proceedings of Fifth Human Factors and the

Web Conference, 1999.

**[Ferguson and Dunlop, 2002]** S. Ferguson and G.R. Dunlop, *Grasp Recognition From Myoeletric Signal, Australian Conference on Robotics and Automation,* Auckland – Australia, 2002.

**[Friedman, 1987]** Friedman, J.H. *Exploratory projection pursuit*, J. of the American Statistical Association, 82, pp. 249-266, 1987.

**[Fu et. al., 2001]** Fu, Y., Creado, M., Ju, C. *Reorganizing Web Sites Based on User Access Patterns*, In Proc. CIKM'01 Tenth International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, pp. 583-585, 2001.

**[Fyfe and Baddeley, 1995]** Fyfe, C., Baddeley, R.J. *Finding compact and sparse distributed representations of visual images*, Network: Computation in Neural Systems, 6(3), pp.333-344, Aug. 1995.

**[Helfrich and Landay, 1999]** Helfrich, B., Landay, J.A. *QUIP: Quantitative User Interface Profiling*, 1999. http://home.earthlink.net/~bhelfrich/quip/

**[Herman and Marshall, 2000]** I. Herman, G. Melançon, Marshall, M.S. *Graph Visualization and Navigation in Information visualization: a Survey*, IEEE Trans. Visualization and Computer Graphics 6(1), pp. 24-43, 2000.

**[Hong and Landay, 2001]** Hong, J.I., Landay, J.A. *WebQuilt: A Framework for Capturing and Visualizing the Web Experience*, In Proc. 10th Int'l World Wide Web Conference, Hong Kong, China, pp. 717-724, 2001.

**[Hotelling, 1933]** Hotelling, H. *Analysis of a complex of statistical variables into principal components*, J. of Educational Psychology, 24, pp. 417-441, 1933.

**[Hyniova et al., 2001]** Hyniova K., Stribrsky A., Honcu J. *Fuzzy control of mechanical vibrating systems*, Proc. Int. Carpathian Control Conference, Krakow, pp 393–398, 2001.

**[Inselberg and Dimsdale, 1999]** Inselberg, A., Dimsdale, B. *Parallel coordinates: A tool for visualizing multi-dimensional geometry,* In Proc. Visualization 90, San Francisco, CA, USA, pp. 361-370, 1999.

**[Kannappady et al., 2006a]** Kannappady, S., Mudur, S.P., Shiri, N. *Visualization of Web Usage Patterns*, In Proc.10th Int'l Database Engineering & Applications Symposium (IDEAS), New Delhi, India, pp.220-227, 2006.

**[Kannappady et al., 2006b]** Kannappady, S., Mudur, S.P., Shiri, N. *Clickstream Visualization Based on Usage Patterns*, In Proc. 5th Indian Conf. on Computer Vision, Graphics and Image Processing (ICVGIP), LNCS, Springer, Madurai, India, pp.339-351, 2006.

**[Kannappady et al., 2006c]** Kannappady, S., Demirli, K., Mudur, S.P., Shiri, N. *Application of Fuzzy Edge Detection for Fast Object-based Image Retrieval*, In Proc. NAFIPS, Montreal, Canada, 2006.

**[Keim, 2002]** Keim, D.A. *Information Visualization and Visual Data Mining*, IEEE Transactions on Visualization and Computer Graphics, Vol. 8, No. 1, 2002.

**[Kohonen, 1989]** Kohonen, T., Self-Organization and Associative Memory, (3rd ed) Berlin: Springer, 1989.

**[Kruskal, 1964]** Kruskal, J.B., *Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis*, Psychometrika, 29(1), 1-27, 1964.

**[Lapanja et al., 1997]** Lapanja, I., Mraz, M., Zimic, N., Virant, J. *Edge detector: towards the solid ground of an image retrieval system*, In Proc. Intelligent Information Systems, IIS '97. pp. 371 - 375, 1997.

**[Latham, 1995]** Latham, R., The Dictionary of Computer Graphics and Virtual Reality, (2nd ed) New York: Springer, 1995.

**[Lee, 1977]** Lee, R.C.T., Slagle, J.R., Blum, H. *A triangulation methods for the sequential mapping of points from N-space to two-space*, IEEE Transactions on Computers, 26, 288-292, 1977.

**[Lopez et al., 2001]** Lopez, N., Kreuseler, Schumann, H. *A scalable framework for information visualization*, Trans. on Visualization and Computer Graphics, 2001.

**[Mobasher et al., 2001]** Mobasher, B., Dai, H., Luo, T., Nakagawa, M. *Improving the Effectiveness of Collaborative Filtering on Anonymous Web Usage Data*, In Proc. of ITWP'01, Seattle, August 2001.

**[Mobasher, 2004]** Mobasher, B. Web Usage Mining and Personalization, Practical Handbook of Internet Computing, Munindar P. Singh (ed.), CRC Press, 2004.

**[Morrison et al., 2003]** Morrison, A., Ross, G., Chalmers, M., *Fast multidimensional scaling through sampling, springs and interpolation, Information Visualization*, 2(1), pp. 68-77, 2003.

**[Munzner, 1998]** Munzner, T. *Drawing Large Graphs with H3Viewer and Site Manager*, Proc. Graph Drawing 98, Springer-Verlag, New York, pp. 384-393, 1998.

**[Nasraoui et al., 2002]** Nasraoui, O., Krishnapuram, R., Joshi, A., Kamdar, T. *Automatic Web User Profiling and Personalization using Robust Fuzzy Relational Clustering*, in E-commerce and Intelligent Methods Ed., Springer-Verlag, 2002.

**[Parker, 1995]** Parker, S.G., Johnson, C.R. *SCIRun: A scientific programming environment for computational steering*, In Proc. ACM IEEE Supercomputing Conf., pp. 1419-1439, 1995.

**[Perkowitz and Etzioni, 1997]** Perkowitz, M., Etzioni, O. *Adaptive web sites: An ai challenge*. In Proc. Int. Joint Conf. on AI (IJCAI), pp 16-23, 1997.

**[Ripley, 1996]** B. D. Ripley, Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge, U.K., 1996.

**[Roberts et al., 2000]** Jonathan C. Roberts, Rob Knight, Mark Gibbins and Nimesh Patel, *Multiple Window Visualization on the Web using VRML and the EAI*, In Proc.VR-SIG Conference, Robin Hollands (Editor). pp. 149-157, 2000.

**[Sammon, 1969]** Sammon, Jr., J.W. *A non-linear mapping for data structure analysis*, IEEE Transactions on Computers, 18, pp. 401-409, 1969.

**[Spehard, 1962]** Shepard, R.N. *The analysis of proximities: Multidimensional scaling with an unknown distance function*, Psychometrika, 27, pp. 125-140, 1962.

**[Stephen, 2000]** Stephen, G. E. *Visual Analysis of Website Browsing Patterns Lecture*

*Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 65-77, 2000.

**[Suryavanshi et al., 2005a]** Suryavanshi, B.S., Shiri, N., Mudur, S.P. *An Efficient Technique for Mining Usage Profiles Using Relational Fuzzy Subtractive Clustering*, Proceedings of the 2005 International workshop on Challenges in Web Information Retrieval and Integration (WIRI 2005), pp. 23-29, 2005.

**[Suryavanshi et al., 2005b]** Suryavanshi, B.S., Shiri, N., Mudur, S.P., *Incremental Relational Fuzzy Subtractive Clustering for Dynamic Web Usage Profiling*, In Proc. WEBKDD Workshop on Training Evolving, Expanding and Multi-faceted Web Clickstreams, Chicago, Illinois, USA, 2005.

**[Suryavanshi B.S., 2006c]** Suryavanshi, B.S., *A New Class of Techniques for Web Personalization*, Masters Thesis, Concordia University, March 2006.

**[Torgerson, 1952]** Torgerson, W. S. *Multidimensional scaling: I. Theory and Method*, Psychometrika, pp. 401-419, 1952.

**[Tufte, 1983]** Tufte, E.R. *The Visual Display of Quantitative Information*, Chesire, Connecticut: Graphics Press, 1983.

**[Van, 2005]** Van den Poel Dirk, Wouter Buckinx *Predicting Online-Purchasing Behavior*, European Journal of Operational Research, 166 (2), pp. 557-575, 2005.

**[Waterson, 2002]** S.J. Waterson, J.I. Hong, T. Sohn, Landay, J.A. *What Did They Do? Understanding Clickstreams with the WebQuilt Visualization System*, Proc. Advanced Visual Interfaces, 2002.

**[Wills, 1997]** G.J. Wills *Nicheworks — Interactive Visualization of Very Large Graphs*, Proc. Graph Drawing 97, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1997.

**[Xie and Beni, 1991]** Xie, X.L., Beni, G. *A validity measure for fuzzy clustering*, IEEE Transactions on PAMI, 13(8), pp. 841-847, 1991.

**[Yager and Filev, 1994]** Yager, R.R., Filev, D.P. *Approximate clustering via the mountain method,* IEEE Transaction on System Man Cybern; 24:1279–1284, 1994.

[Zadeh, 1965] Zadeh, L. Fuzzy sets, Information Control, vol. 8, pp. 338–353, 1965.

# Appendix

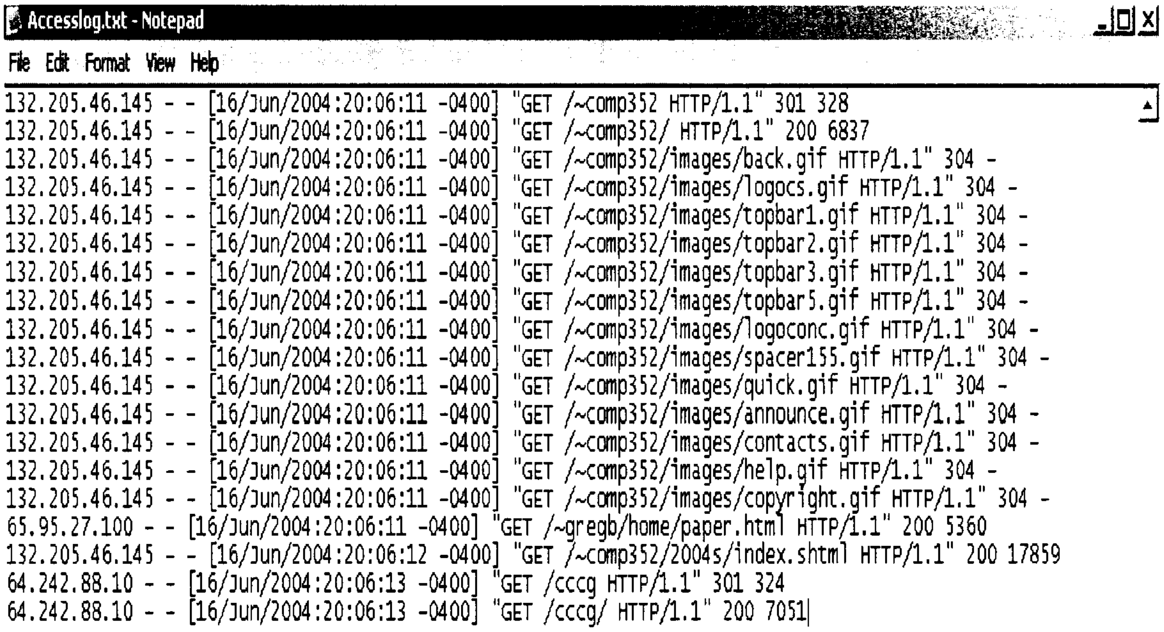
## Collection of Web Data

Data resulting from users browsing the web site can be collected explicitly or implicitly. Explicit data is collected through active involvement of the user, typically from fill-in forms (registration) and questionnaires. Such data may contain generic information such as date of birth and area code, along with some dynamic information, which is likely to change over time such as favorite television programs or football teams. Explicit data collection requires users to exert most of the efforts and make the initial investment, and hence depends much on user's motivation. On the other hand, users are not directly involved in implicit data collection. The navigation behavior and preferences of the user can be learnt typically through analysis of the user accesses and usage of the website. The web server is an important data source for performing web usage mining because it explicitly records the browsing details from visitors to the site. The data recorded in server logs reflects the (possibly concurrent) access of a web site by multiple users. Data collection can also be done at the client side by using a remote agent (such as Javascripts or Java applets) or by modifying the source code of an existing browser (such as Mosaic or Mozilla) to enhance its data collection capabilities. However, these methods require explicit user cooperation, which is not always possible. Another data source is web proxy server, which acts as an intermediate level of caching between client browsers and web servers. This may serve as a good data source for characterizing the browsing behavior of a group of anonymous users sharing a common proxy server. For more details see

[Srivastava et. al. 2000].

# Data Preprocessing

Log files produced on the web servers are text files with a row for each HTTP transaction. Fig. 27 shows a typical log file.



```
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352 HTTP/1.1" 301 328
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/ HTTP/1.1" 200 6837
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/back.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/logocs.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/topbar1.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/topbar2.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/topbar3.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/topbar5.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/logoconc.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/spacer155.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/quick.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/announce.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/contacts.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/help.gif HTTP/1.1" 304 -
132.205.46.145 - - [16/Jun/2004:20:06:11 -0400] "GET /~comp352/images/copyright.gif HTTP/1.1" 304 -
65.95.27.100 - - [16/Jun/2004:20:06:11 -0400] "GET /~gregb/home/paper.html HTTP/1.1" 200 5360
132.205.46.145 - - [16/Jun/2004:20:06:12 -0400] "GET /~comp352/2004s/index.shtml HTTP/1.1" 200 17859
64.242.88.10 - - [16/Jun/2004:20:06:13 -0400] "GET /cccg HTTP/1.1" 301 324
64.242.88.10 - - [16/Jun/2004:20:06:13 -0400] "GET /cccg/ HTTP/1.1" 200 7051
```

**Figure 27: Sample log file from web server**

Let us consider a typical log entry:

*195.162.218.155 - - [27/Jun/2004:00:01:54 -0400] "GET /cccg/ HTTP/1.1" 200 38890 "/events/" "Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1; SKY11a)"*

The first part of this line, 195.162.218.155, specifies the IP-address of the client who made the request to the server. This IP-address can be used to identify further accesses by the same user to this site. Different IP-addresses naturally correspond to different visitors of the site. The second component in this line gives us information on when the request was made. The third part “GET /cccg/ HTTP/1.1” of this entry is called the request line and consists of three parts: The GET-part specifies the method used to request the page. If the website visitor requests a normal web page, then the method used will always be GET. Other possibilities are POST, to send values in a form to the server, and HEAD. The second part of the request line indicates which file was requested. The final part specifies the protocol used to request the file. The two numbers, 200 and 38890, following the request line indicate a status code and the size of the returned file, respectively. The status code 200 indicates that the request was successfully completed. Other status codes indicate various types of errors, from which “error 404: Page not found”, is probably the most familiar to the reader. The next part of the line designates the referrer. This is the page that refers to the requested page. From this line, it can thus be concluded that there was a link from “/events/” to the page “/cccg/”. Finally, the last component of the line specifies the agent used (browser).

The required tasks in usage data preprocessing include data cleaning, page identification, user identification, session identification (or sessionization), and the inference of missing references due to caching. We briefly discuss some of these tasks below. For a more detailed discussion interested readers are referred to [Cooley, 2000; Cooley et al., 1999, Mobasher 2004].

Data cleaning results in filtering out log entries that are not required or are irrelevant for our task. These include entries that: (i) result in any error (indicated by the error code), (ii) use a request method other than “GET”, or (iii) record accesses to image files (.gif, .jpeg, etc), which are typically embedded in other pages and are only transmitted to the user’s machine as a by product of the access to a certain web page which has already been logged. Also references due to spiders and web robots are removed. Sessionization is the process of segmenting the access log of each user into sessions. Web sites without the benefit of additional authentication information from users and without mechanisms such as embedded session IDs must rely on heuristic methods for sessionization. The goal of a sessionization heuristic is reconstruction of the real sessions, where a real session is the actual sequence of activities performed by one user during a visit to the site. Generally, sessionization heuristics are time-oriented where either global or local time-out estimates are applied to distinguish between consecutive sessions.

The preprocessing tasks ultimately result in a set of M pages (URLs), $U = \{url_1, url_2, \ldots, url_M\}$, and a set of $N_U$ user sessions, $S = \{s_1, s_2, \ldots, s_{Nu}\}$, where each $s_i \in S$ is a subset of U. Conceptually, we can view the $i^{th}$ session $s_i$ as a sequence or vector of $l$ pairs

$$s_i = \langle (p_1, w(p_1)), (p_2, w(p_2)), \ldots, (p_l, w(p_l)) \rangle,$$

where, each $p_i = url_j$ for some $j \in \{1, \ldots, M\}$, and w $(p_i)$ is the weight associated with each $p_i$ in session $s_i$. In most web usage mining tasks, we can choose two types of weights: binary, representing the existence or non-existence of a page in the session; or duration of time spent on each page in the session by the user.

Whether the sessions are viewed as sequences or as sets (without considering the order of web page access) depends on the goal of the analysis and the intended applications. For sequence analysis and the discovery of frequent navigational patterns, one must preserve the ordering information in the underlying session. On the other hand, for clustering tasks, we can represent each user session as an M-dimensional vector, where dimension values are the weights of these pages.

In this work, time-oriented heuristic is used and binary weights are considered. Hence, each session is an M-dimensional binary vector such that:

$s_{ij}$ = 1 if the i<sup></sup>th session had the jth URL clicked

$s_{ij}$ = 0, otherwise.

If more than 45 minutes of time has elapsed between two accesses from the same IP-address, the older session is assumed to be complete and a new session is begun.