WEB MINING FOR BETTER WEB USABILITY

Golam Mostafiz
Student ID: 03201078

Department of Computer Science and Engineering

December 2007

BRAC University, Dhaka, Bangladesh

DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of
Supervisor

Signature of
Author

ACKNOWLEDGMENTS

Special thanks to Dr. Mumit Khan, who is my supervisor and helped me out whenever I faced difficulty regarding my thesis. And also thanks to Shahid M. Asif, who gave me the initial concept of web usability.

ABSTRACT

To develop more effective user-oriented learning techniques for the Web, we need to be able to identify a meaningful session unit from which we can learn. Without this, we could have a high risk to mix up the different user's activity in the web. We are interested to detect boundaries of sequences between related sessions that would group the activities for a learning purpose. But identification of user session is not always easy where logged on and cookie information is not available.

The problem of predicting user access in web pages has recently gets a significant attention. Several algorithms have been proposed, which find important applications, like user profiling, web perfecting, design of adaptive web sites, etc. In all these applications the main issue is the development of an effective prediction system. Because of its importance in reducing user perceived latency present in every Web-based application, which is a usability issue.

This thesis paper describes a data mining technique for identify user sessions from huge amount of web log data and a web system, which makes prediction about the user target page by using those sessions to guide the user in World Wide Web.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

1.1 Introduction

It is impressive to have adaptive techniques for Web based information retrieval system (IRSs), where increase use of the Web, amount of information available, and variety of regular user, which meets individual user needs effectively. This is very important to track individual user and their activities in the web when we are doing user oriented learning. For this necessity, we need to identify series of request that is made by a user within a unit amount of time. Which, we refer as **Session**. This paper focuses on the temporal ordering of activities clustered according to similar request made to the server to identify user's session boundary.

Again, Modeling and predicting user's access in the Web has attracted lot of research interest. It has been improve the web performance through caching and prefetching, recommend related pages, improve search engines and personalize the browsing in a web site. The core issue of this research is development of a system that deduces the future user access. The approach is, use user history to make prediction about future. This system can be easily adapted to the related topics like related topics, user profiling, recommender systems, design of adaptive web sites, etc.

The main objective of this research is, identify the user session and using those session as input to develop a system, which will run on server, predict the next request of the user. Thus reduce user perceived latency as well as offer a better web usability.

CHAPTER II

Methodology

2.1 Session Identification

If we view a user with an interest in a specific topic as acting in a particular role, then it is not unreasonable to assume that the activities in the same session are likely to correspond to one role. That means, if we notice the web behavior of a user, we can see that in any particular time period, user is interested in one specific topic. As a result we can say consecutive request for that user in the given period is similar. Now we can set session boundary by grouping those similar requests.

Doing clustering on the requested URL field of server logs can do now grouping of similar requests. But there are some problems. Errors occur when two adjacent activities for related search statements are allocated into different sessions. Alternatively errors occur when unrelated activities are allocated into the same session.

2.2 Predict the future request

Web servers are in better place in making predictions about future references, since they log a significant part of requests by all Internet clients for the resources they own. We are talking about a system, which will predict the user next request.

Whenever we are ready with the identified session, we can use those session as input of that predictive system. Our main topology is, make the system richer with time by feeding those session. The proposed architecture is given below,

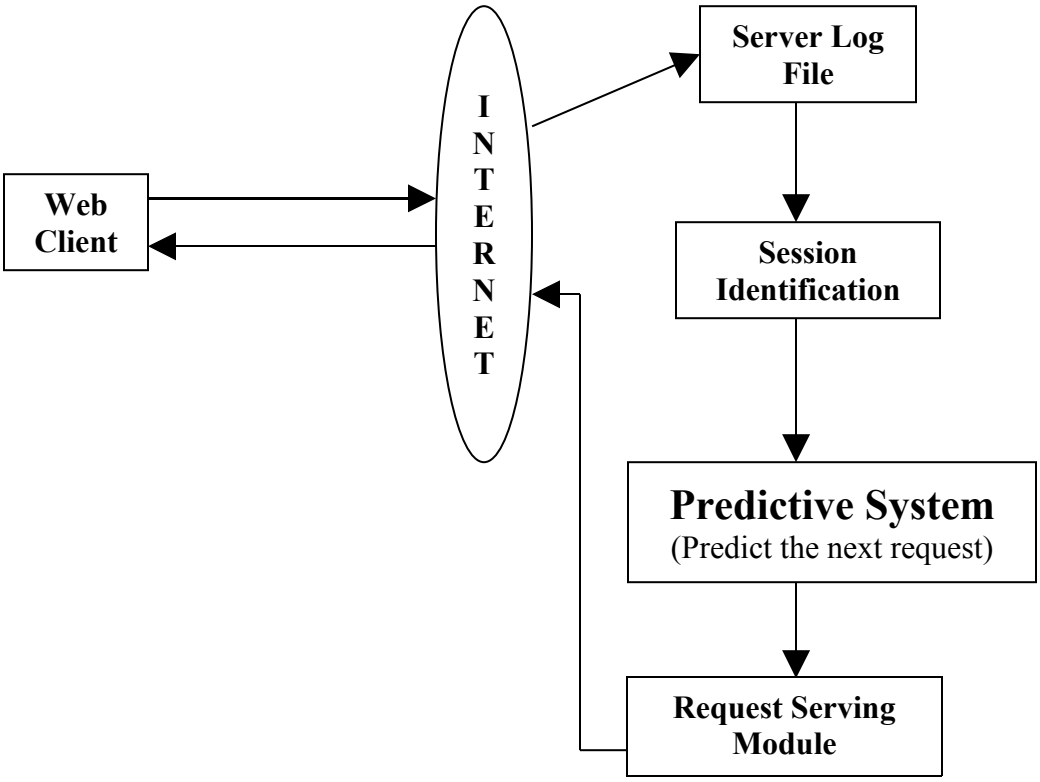

Fig 2.1: Architecture of predictive system

When a client makes any request to server, server stores that request as a server log entry. We are taking that request from server log and identify sessions and use those session in the predictive system. Now what the predictive system does is, for the first time it creates a graph using those identified sessions. Where each node is a web page containing some other attributes. And an edge represents the link from go to one page from another. And after that it keeps update the graph by calculating the probability and generating new edges and nodes.

So from here we can tell that, with the more time the predictive system run in server, it will give more effective result. After getting the result from the predictive system we are able to guide or feedback the user by the request-serving module.

Before go to the experiment, discussion about web log format and some algorithm is necessary.

Web log or server log follows a standard format. Each of servers can use different web log format. But there are must be some common fields. A typical configuration for the access log might have following fields.

IP Address.
Date.
Time.
Method- Get/Post.
Requested URL.
Referred Page.
User Agent (Browser).

Web Caching is a problem in this case. When a page is loaded locally without making any request to the server (e.g. when we do refresh) that is called cached page. Now this is a noticeable problem because we don't have any log entry for that request in server side. But we can follow a series of steps to overcome from caching problem. The main principle is, if the referring page file of a session is not part of the previous page file of that session, the user must have accessed a cached page. The steps are,

Step 1: Form a table with the web logs.

Step 2: Sort all users based on their IP address and the browser type.

Step 3: Calculate session using heuristics (Timeout).

Step 4: Do path competition

Now in **Step 3**, we are assuming that, if the difference between consecutive requests from same user is 30 minutes, they are not grouped in same session.

In **Step 4** (Path Competition),

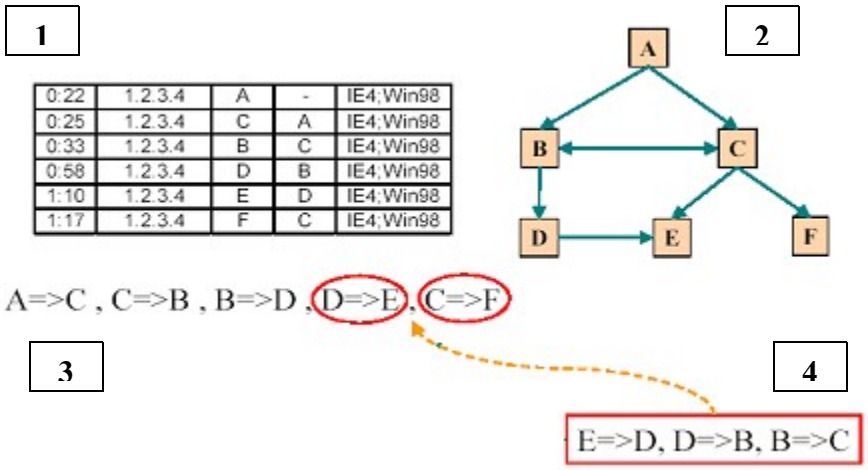

Fig 2.2: Path competition

Assume that in a given moment, we have 6 log entries (marked 1 in figure 2.2) and a graphical representation of the web logs (marked 2 in figure 2.2). Now consecutive requests came from user (marked 3 in figure 2.2). If we notice, then after 4th request D to F, we are jumping directly C to F although we don't have any path. So we can say confidently between 4th and 5th request user made more 3 requests (marked 4 in figure 2.2).

2.3 Algorithm for predictive system:

```
Page source = session [0];
boolean edgeFound=false;

for each web logs i in session
        Page destination=session[i];
        edgeFound= getEdge(source,destination);
        if(edgefound= =true)
                increment the value of edge usage;
        else
                create a new edge between the source and destination;
                increment the value of edge usage;
        updatePrababilityMatrix(source);
        source=destination;
        edgeFound=false;
end for

updatePrababilityMatrix(source)

        edges[]=getOutgoingEdges(source);
        for each edge i in edges[]
                totalRequest += edges[i].getUsage();
        end for
        for each edge i in edges[]
                destination=edges[i].getConnectedTo(edges[i]);
                prMatrix[source][destination]= edges[i].getUsage()/totalRequest;
        end for
```

CHAPTER III

Experiment

In my thesis experiments consisted of two stages: automatically detecting session boundaries then use those session to predict the future request of a user.

3.1 Data

For experiment the data I have used are raw web log entry. Each web log entry contains those fields, which are described earlier in this paper. For session identification and making prediction about the future request, experiments are done on 5600 web log entries.

3.2 Session identification

For session identification, the experiment was done in MATLAB. As we discussed before I will do clustering on web logs to identify sessions, so I have used Kmeans clustering and hierarchical clustering in MATLAB by feeding the web logs as input. After doing this two type of clustering I visualize the result by doing silhouette plot in MATLAB.

Now we need to about the term “silhouette”. It’s a plot tool in MATLAB, which displays a measure of how close each point in one cluster is to points in the neighboring clusters. This measure ranges from +1, indicating points that are very distant from neighboring clusters, through 0, indicating points that are not distinctly in one cluster or another, to –1. That means each entry with more positive silhouette value, have more isolated from other classes.

Before clustering we need to specify the number of classes as a parameter. Now here complexity arises because we don't know the actual number of users. We assumed in our experiment, that we have 5 users, who made all the requests. So to fix that parameter we assigned the value 5 on it.

3.3 Predicting the Future request

The experiment for predicting future request is made by JAVA. The system works as follows,

Suppose a user start his session by going to www.A1.com which is the homepage. And from the homepage he goes to B1. The second requested URL will looks like www.A1.com/B1. From B1 he goes to C1 and then again backs to the page B1. So 3rd and 4th requested URL respectively looks like www.A1.com/B1/C1 and www.A1.com/B1. Initially the predictive system will create a graph as following,

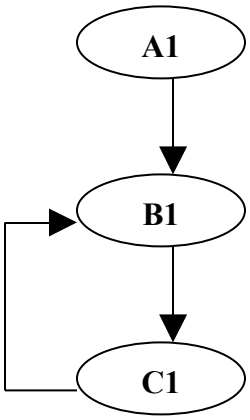

Fig 3.1: Example of creating graph from user request - 1

Now consider a situation where graph is already created like follows,

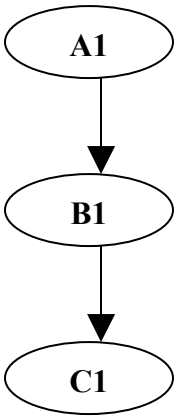

Fig 3.2: Example of creating graph from user request - 2

For same session, it will just update the graph for the last request by creating a new edge from C1 to B1.

Every time graph is updated, it recalculates the number of an edge is used to go from one particular source to destination. Assume that the edge use to go from homepage (A1) to B1 is n times. Now for any further request form homepage to B1, the graph will update itself by incrementing the value of edge usage n+1, instead creating a new edge.

To make my idea clearer, the sample resulting graph of my experiment is shown on below, where we updated the graph by using only 10 sessions. www.smsync.com is the home page, so represented as root in the graph. Any further request from root to any pages are the inter nodes of the graph. Leaves are only those pages or files, from where we didn't get any request. Each edge is labeled with number of time used.

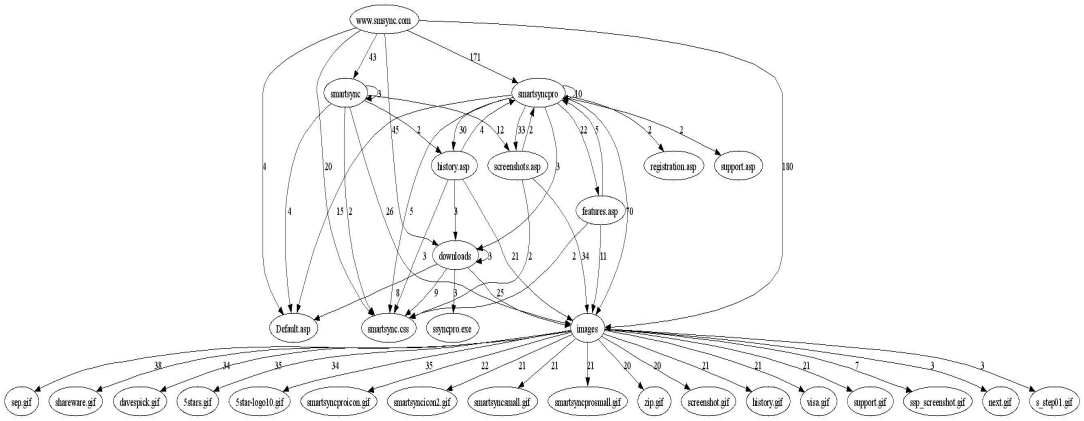

Fig 3.3: Graph from user request – 1

The actual graph of my experiment is shown in the result section Fig 4.7.

I have used a probability matrix to predict the future request. Every time when we updating the graph, probability matrix should updated simultaneously. Each edge holds the information about the number of time it is used to go from one page to another. The outgoing edge from a node holds important concept. Which is, we can go only those pages from a node, where out going edges of that node let us go. So the sum of all outgoing edges usage, is the total number of requests are made from that particular node. Now how we can predict the future request by using that probability matrix in my experiment is shown below,

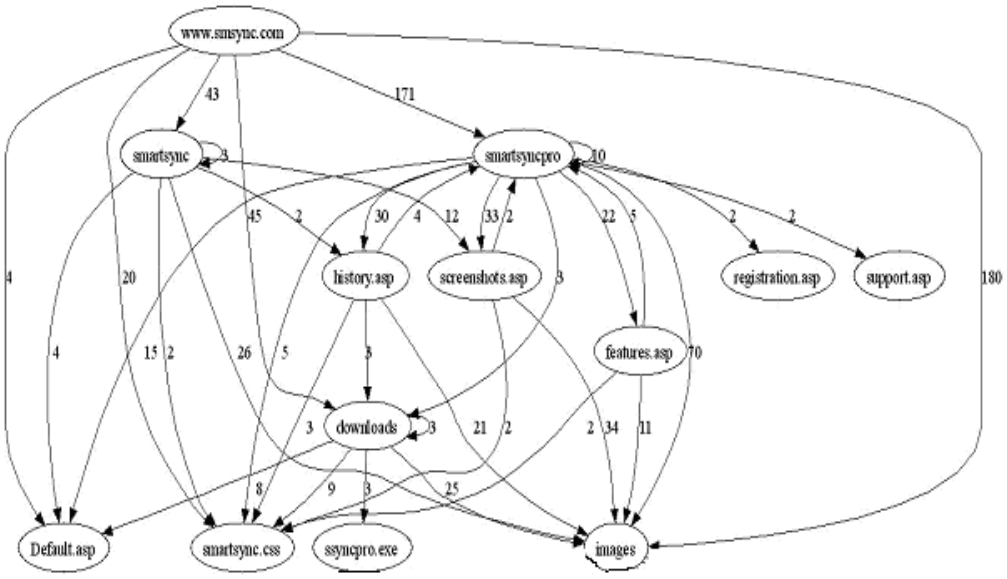

Fig 3.4: Graph from user request – 2

From the figure 3.4 above we can see that, from root www.smsync.com, we can go Default.asp, smartsync, smartsyncpro, downloads, smartsync.css, and images.

So total request from the root is, 4+20+43+45+171+180= 418

Now probability to go from root to,

File name	Probability
Default.asp, 3/413	0.00726
smartsync.css, 20/413	0.0484
smartsync, 43/413	0.104
smartsyncpro, 171/413	0.414
downloads, 45/413	0.109
images, 180/413	0.4358
Total	**1.00**

Table 3.1: Calculation for updating probability matrix

Now from root, images have the higher probability to go, which is 04358. In probability matrix each rows represents sources and columns represents the destinations.

For root, the probability matrix entries will be,

Pr [www.smsync.com] [Default.asp] = 0.00726
Pr [www.smsync.com] [smartsync.css] = 0.0484
Pr [www.smsync.com] [smartsync] = 0.104
Pr [www.smsync.com] [smartsyncpro] = 0.414
Pr [www.smsync.com] [downloads] = 0.109
Pr [www.smsync.com] [images] = 0.4358

Now notice we cannot go www.smsync.com to history.asp. In that case the probability matrix entry will be,

Pr [www.smsync.com] [Default.asp] = 0.00

According to the calculation, sum of each row must be equal to 1.00.

CHAPTER IV

Result

4.1 Session Identification

I have done Kmeans and hierarchical clustering respectively on requested URL, requested time and bytes downloaded to identify the session boundary. The result of my experiment for session identification is given below,

K-Means Clustering On **Requested URL**:

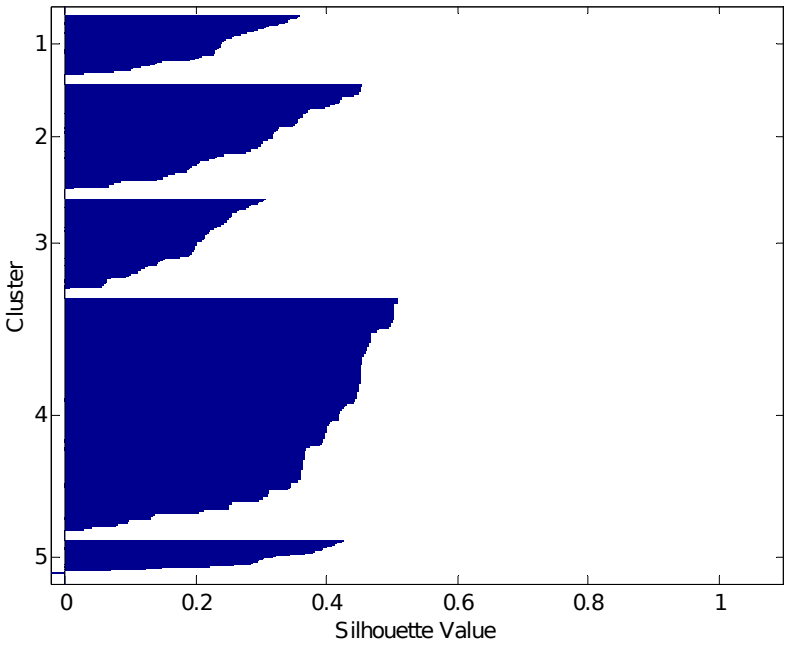

Fig 4.1: K-Means clustering on requested URL

Hierarchical Clustering On **Requested URL**:

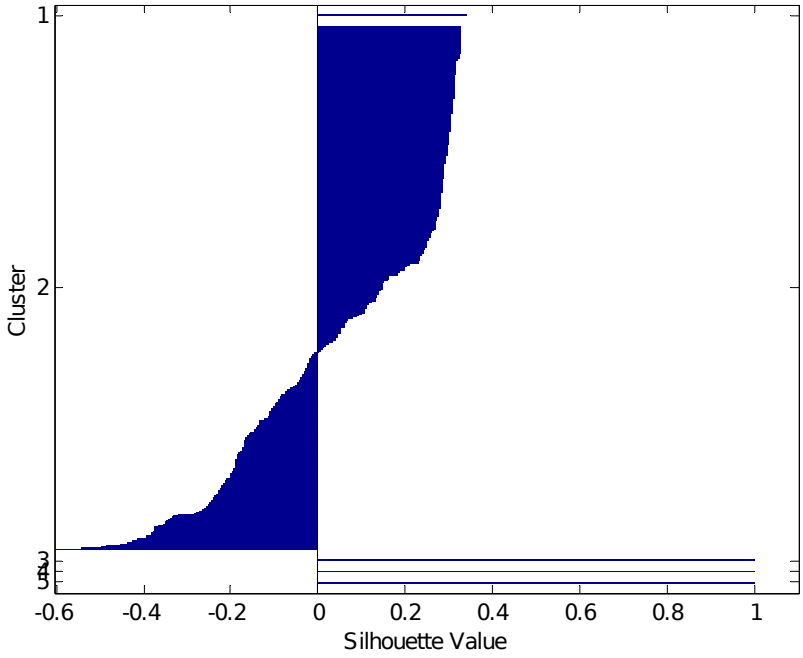

Fig 4.2: Hierarchical clustering on requested URL

K-Means Clustering On **Requested Time**:

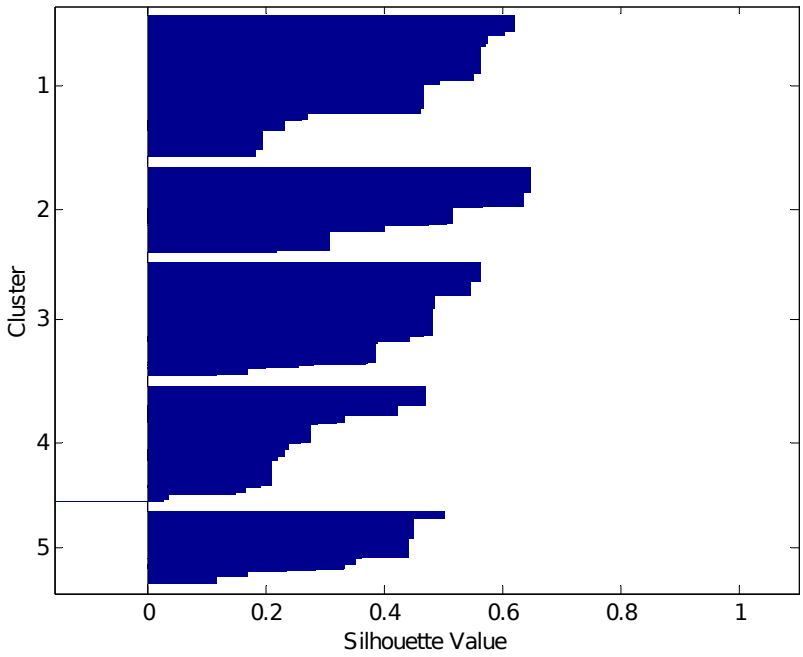

Fig 4.3: K-Means clustering on requested time

Hierarchical Clustering On **Requested Time**:

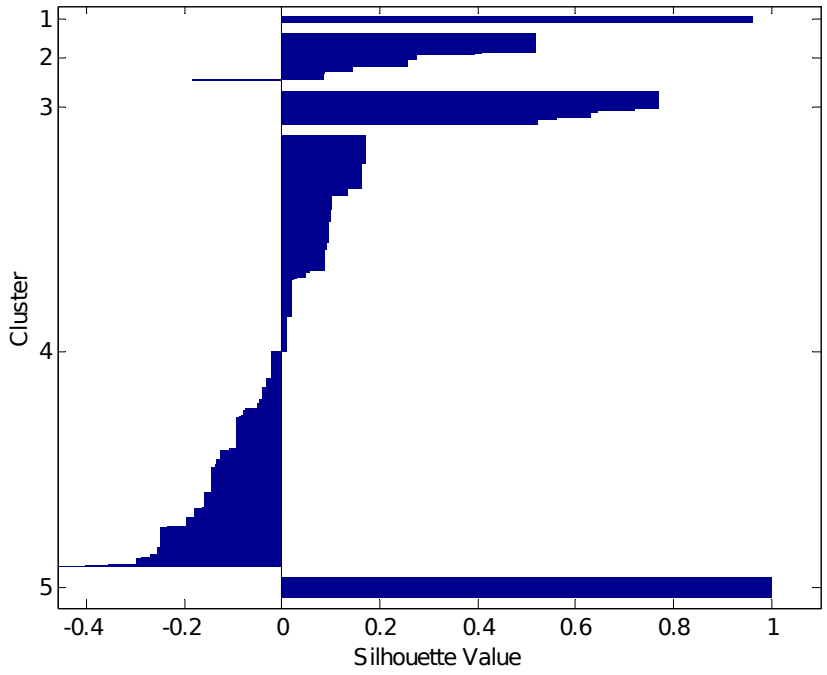

Fig 4.4: Hierarchical clustering on requested time

K-Means Clustering On **Byte Downloaded**:

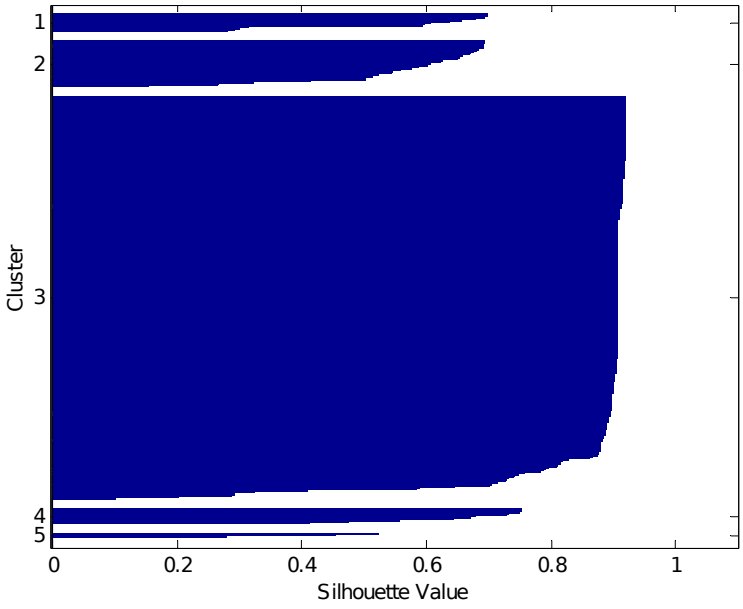

Fig 4.5: K-Means clustering on byte downloaded

Hierarchical Clustering On **Byte Downloaded**:

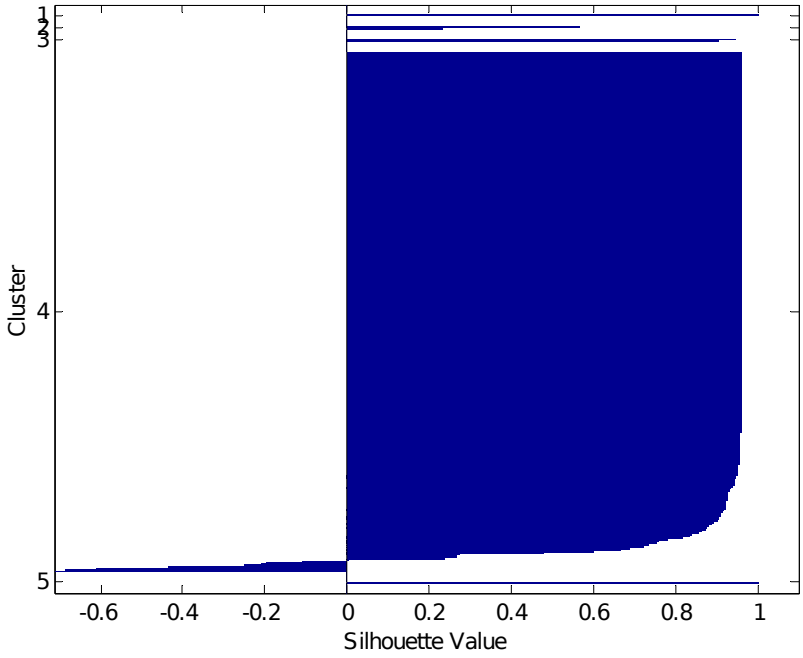

Fig 4.6: Hierarchical clustering on byte downloaded

4.2 Predictive System

The output of the predictive system can be described by following ways,

Partial portion of textual representation of the graph is given below,

```
AVG Time Stayed In smartsync.css Is 20.0
 -> smartsync.css -> www.smsync.com
AVG Time Stayed In Default.asp Is 7.0
 -> Default.asp -> www.smsync.com
AVG Time Stayed In features.asp Is 13.0
 -> features.asp -> smartsyncpro -> www.smsync.com
```

The page smartsync.css is the child node of www.smsync.com, where average time of 20 sec user stayed in this page. For the page Default.asp is also similar to the smartync.css. features.asp is the child of smarsynctpro and which is further child of www.smsync.com, where user stayed average time of 13 sec.

In my experiment, I calculated most viewed page, most pressed button and the pages in which user spent their time most.

```
Most Viewed Pages,      page 1 : smartsync.css
                        page 2 : Default.asp
                        page 3 : screenshots.asp

Most Pressed Button,    page 1 : sep.gif
                        page 2 : davespick.gif
                        page 3 : 5star-logo10.gif

Most Time spent in,     page 1 : smartsync.css
                        page 2 : Default.asp
                        page 3 : features.asp
```

It is possible in my predictive system to generate all the possible paths to go from root to any particular destination page based on the previous learning.

If I set the destination smartsync.css, then the predictive systems result is given below,

```
Give Page Name to See The Path : smartsync.css

By 7 Way We Can Reach To smartsync.css

-> www.smsync.com -> smartsync.css
Total used: 20

-> www.smsync.com -> smartsync -> smartsync.css
Total Used : 2

 -> www.smsync.com -> smartsyncpro -> downloads -> smartsync.css
Total Used : 9

 -> www.smsync.com -> smartsyncpro -> smartsync.css
Total Used : 5
```

```
 -> www.smsync.com -> smartsyncpro -> history.asp -> smartsync.css
Total Used : 3

 -> www.smsync.com -> smartsyncpro -> screenshots.asp -> smartsync.css
Total Used : 2

 -> www.smsync.com -> smartsyncpro -> features.asp -> smartsync.css
Total Used : 2
```

Now I want to show the result of predicting future request. I trained the system by training data, which are web logs. After training, I used 641 number of web request as input. The predictive system will predict the request and check each time with the given input whether it is correct or not. I am showing for 6 inputs among 641.

```
Expected Request = /images/support.gif
Actual Request = /images/support.gif
HIT

Expected Request = /downloads/Default.asp
Actual Request = /downloads/Default.asp
HIT

Expected Request = /smartsync/screenshots.asp
Actual Request = /smartsyncpro/Default.asp
MISS

Expected Request = /images/screenshot.gif
Actual Request = /images/screenshot.gif
HIT

Expected Request = /smartsyncpro/screenshots.asp
Actual Request = /smartsyncpro/screenshots.asp
HIT

Expected Request = /images/visa.gif
Actual Request = /smartsync/history.asp
MISS
```

After processing all 641 requests in my experiment, I have hit rate and miss rate. Which is given below,

```
Total Request: 641
Total Hit: 436
Total miss: 205
Hit Rate: 68.0%
Miss Rate: 32.0%
```

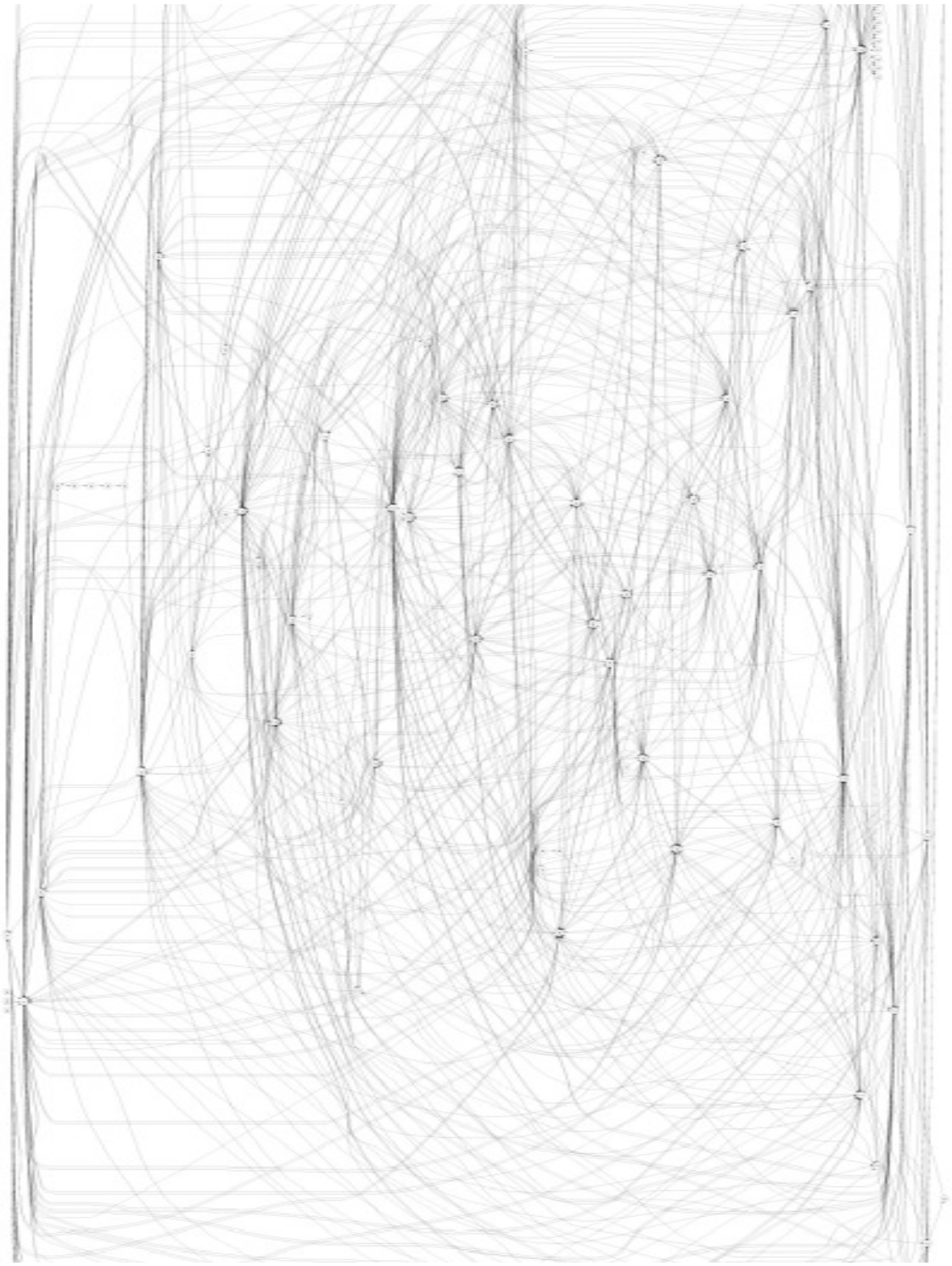

Fig 4.7: Part of the actual graph

CHAPTER V

Discussions

For user session identification, I did Kmeans and hierarchical clustering on Requested URL, Requested time and byte downloaded. The result is plotted with respect to silhouette value. For requested URL when I did Kmeans clustering fig-4.1, all requests are clustered well, because all requests of each class have positive silhouette value. That means all classes are properly separated from one another.

But when I did hierarchical clustering on the requested URL fig-4.2, the result I found is not that good. Because all request of class 3 has negative silhouette value. So requests of class 3 are not properly isolated from other classes. One another noticeable thing is, number of request in class 1,4 and 5 is very low compare to class 2 and 4. So classes are not properly balanced too.

Same way if we see the remaining results of my experiment fig-4.3 to fig-4.6, we can say that Kmeans clustering have more effective result than hierarchical clustering.

In my experiment the actual graph based on which I have took decisions is given in fig-4.7**.** Now I want to show an interesting part, where I tried to find the entire possible path that can be used to go from one page to another destination page. The experiment will suggest us the best path to take (1st path). In my experiment, to suggest best the path, I have done the calculation based on the number of times that path is used. If we notice then we can see, user used that path most which is the shortest (1st path).

I have shown previously the hit ratio and the miss ratio, where I tried predicting the next request of the user. Hit ratio is 68% and miss ratio is 32%. 68% of hit ratio is not that effective. But one think I should say, that in my experiment

number of web pages, the content of the web pages, and the number of link in a web page are all unknown. The system will learn these information with time. So situation like this 68% of hit ration is not that bad also. Again based on time this system will learn more, thus hit ration will increase. The data I have used in my experiments is not widely available. 68% of hit ratio could be result of using lower quality data.

CHAPTER VI

Conclusion

In this paper, I have presented a method for detecting session boundaries by using a minimal amount of user information that is typically available in web logs. In my paper I have talked about two errors that can be occurred when we are identifying the user session. One is, two adjacent activities for related search statements are allocated into different sessions. Another is unrelated activities are allocated into the same session. In future work, I intend to explore methods of improving automatic session boundary detection by reducing percentages errors.

We considered the problem of predictive Web prefetching, that is, of deriving users' future requests for Web documents based on their previous requests. Predictive prefetching suits the Web's hypertextual nature and reduces significantly the perceived latency. Web prefetching has common characteristics with other Web applications that involve prediction of user accesses, like user profling, recommender systems and design of adaptive web sites. Hence, the proposed method can be easily extended to these kinds of applications.

LIST OF REFERENCES

1. Ay¸se G¨oker and Daqing He, Analysing Web Search Logs to Determine Session Boundaries for User-Oriented Learning, School of Computer and Mathematical Sciences,The Robert Gordon University

2. Balabanovic M., Shoham Y., and Yun Y.: An Adaptive Agent for Automated Web Browsing. Tech. Rep. CS-TN-97-52, Dept. of Comp. Sci., Stanford University

3. Alexandros Nanopoulos, Dimitris Katsaros, Yannis Manolopoulos, Effective Prediction of Web-user Accesses, Data Engineering Lab, Department of Informatics Aristotle University, Thessaloniki 54006, Greece

4. Judy Kay and Andrew Lum ,Creating User Models from Web Logs School of Information Technologies University of Sydney, NSW, 2006 Australia