Methodology

3. METHODOLOGY

The navigation pattern of an online user often reflects user's mental model and for this reason, websites owners and developers pay more attention to the navigation pattern, so that their site can closely reflect the user's psychological model and provide successful user experience. This observation has made user navigational paths a key factor in construction of the scenarios used in Website interaction design and in the evaluation of existing designs.

The main difficulty in studying navigational patterns is that they are prone to change from time to time and different users employ different strategies for surfing. Some factors that have a direct impact on the navigation are user's browsing goal, expertise, familiarity with site, time and work pressures and perceived cost of information. It is also found out that the site design and structure of the site also influences the user's searching pattern.

In order to study these patterns effectively, browsing patterns gathered from a site is very important. Several solutions have been proposed and the usage of clustering and classification is more frequently used on such solutions. This research work is one another attempt made to propose a hybrid system that uses clustering and classification methods to discover the user's navigation pattern and analyze them from the server's web log file.

3.1. WEB USAGE MINING

A general web usage mining scenario is shown in Figure 3.1.

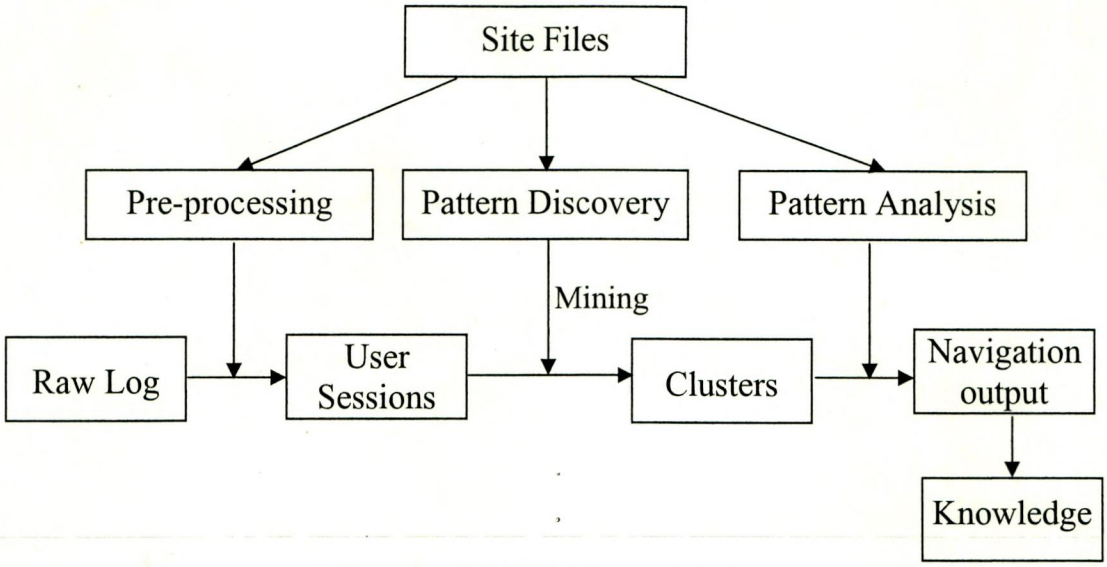

Figure 3.1: Web Usage Mining

Mining user navigation pattern from server log files is composed of various steps. The first step is the preprocessing of raw log file into meaningful user sessions. User navigation patterns have details about the interest of a user browsing a website. The system presented in this research work can be used to study the user behaviour while navigating within a website. The steps involved and the various techniques used in the present research work are detailed in the following sections.

3.2. LOG FILES

In this section a detailed description of the weblog file format is used for the present study. A log file is defined as a file which records the activity on a web server. Log files yield information such as, which files are requested, when files are requested, who requested them, and where they were referred from. Each line in the log file represents a single "hit" on a file on the web server, and consists of a number of fields and the format of the log used for analysis differ from server to server. The format of the web log file is shown in Table 3.1. A hyphen ('-') in any of these fields indicates missing data.

TABLE 3.1

WEB LOG FILE FORMAT

S. No	Field Name	Description	Example value
1	IP Address	IP address of the remote host (client) which made the request to the server	127.0.0.1
2	UserIDand Password	Provides the username and their corresponding password used during the access of a content-secured transaction	Voder23 12ert35
3	Timestamp	The date, time and time zone when the server finished processing the request.	[10/Oct/2000:13:55:36 -0700]
4	Access request	Request line from the client. It has three parts, the METHOD, URL STEM and PROTOCOL used during transmission.	GET /download/windows/asctab31.zip HTTP/1.0
	Method	Can be GET (standard request for a document or program) or POST (during transmission indicates the server that data is following) or HEAD (used by link checking programs, not browsers, and downloads just the information in the HEAD tag information)	GET POST HEAD
	URL	the address of the web content to be retrieved	/download/windows/asctab31.zip
	Protocol	protocol used during transmission along with the version number	HTTP/1.0
5	Status	The resulting status code. A status code of 200 means success transaction. There are four classes of codes, (i) Success (200 series) (ii) Redirect (300 series) (iii) Failure (400 series) and (iv) Server Error (500 series)	200
6	Bytes	The number of bytes transferred. In case if this matches the size of the file requested, then it was a successful download. If the number had been less, then that would indicate a failed or partial download	3784
7	User agent	The User Agent is whatever software the visitor used to access this site. It's usually a browser, but it could equally be a web robot, a link checker, an FTP client or an offline browser.	Mozilla/4.7 [en]C-SYMPA (Win95; U)
8	Referral	Refers to the previous page visited by the same user	/movies/tamil

3.3. PREPROCESSING

Preprocessing of a web log file simply reformats the entries of a log file into a form that can be used directly by the subsequent steps of the log analyzer. Preprocessing "consists of converting the usage, content, and structure information contained in the various available data sources into the data abstractions necessary for pattern discovery". The preprocessing is performed in four steps as given below.

(i) Cleaning
(ii) User identification
(iii) Session identification and
(iv) Formatting.

i) Cleaning

In the **first step**, (i.e.,) cleaning of data, unwanted data is deleted. Examples of unwanted data include requests for images, Javascripts, flash animations, video, etc. These data are not required for user navigation and hence are deleted form the log file.

ii) User identification

There are some methods for user identification, **second step**, identify cookies, Identd, through IP address and username. Cookies are defined as data sent by the server to the client; data locally stored in cookies and is sent to the server with each request. The disadvantages of using cookies for user identification are twofold.

(i) Some users lock the use of cookies, so server cannot store information locally in the user machine;
(ii) User, for want of disk space, often deletes cookies.

Identd is a protocol defined in RFC 1413 (2010), which can be used to identify users. This is performed by connecting the users with a unique TCP connection. The problem with this is that users should configure this protocol, if not is ignored by server and are not recorded in the log file.

Another method is to use user names and this again poses a problem, as it is often empty, as indicated by a hyphen ('-') in web log file.

The fourth method is the identification of users through the IP address. This is the most frequently used method, easy to capture and is never empty. This has a problem of same users with multiple IP addresses. In the present research work, session identification is used to solve this problem. Moreover, this problem is not of main concern with user navigation pattern and does not deal with a specific user.

iii) Session identification

The **third step**, Session Identification, is performed using session timeout value. Session timeout is a process that automatically prevents user access to a system or application after a period of inactivity. The purpose of timeouts is to lock out unauthorized users when a system is unattended or when someone forgets to log out of an application. Typical session timeouts are between 25-30 minutes. In the present research work, a 30 minute session timeout is used during experimentations.

iv) Formatting

The original log file is pruned from unwanted data and is formatted which is done in the **last step**. This formatted data is taken as input in the next process of web log analysis.

3.4. PATTERN DISCOVERY

Pattern discovery "draws upon methods and algorithms developed from several fields such as statistics, data mining, machine learning and pattern

recognition". Already several methods and techniques have been developed for this step. Some of the frequently used solutions are statistical analysis, clustering, classification, association rules, sequent patterns and dependency modeling. Etminani *et al.* (2009) used an Ant-based clustering approach to discover navigation patterns for discovering navigation pattern. This method has some issues when combined with log file knowledge discovery. The issues are

(i) One of the critical *problems* of *ant-based clustering* is the high-execution times.

(ii) The second issue is the inability of ant-based algorithm to detect the completion of clustering process.

(iii) Another important issue is that one ant can produce the same results as many ants in the models function like stochastic sampling algorithms and the result is repeated and are considered during clustering.

All the above mentioned problems motivated the present research work to use a classification algorithm, after the ant-based clustering. The classification algorithm used is the Longest Common Subsequence (LCS) algorithm. The study divides the pattern discovery and analysis phase into two phases, the online and offline phase. The offline phase consists of analyzing the log file and producing the clusters. The online phase, on receiving a new user request, the URL and session to which the user belongs are identified and are classified to the correct cluster. Both the ant-based clustering algorithm and the LCS algorithm are explained in the next section.

3.5. PATTERN ANALYSIS

The final stage is the analysis of the patterns discovered in the previous step. This is done in two stages:

(i) Validation : To identify relevant rules or patterns from which interesting patterns can be discovered

(ii) Interpretation : Mathematical interpretations which can be used by scientists to discover knowledge

3.6. ANT-BASED CLUSTERING (OFFLINE PHASE)

Clustering is concerned with the division of data into homogenous subgroups. Informally, the objective of this division is twofold: data items within one cluster are required to be similar to each other, while those within different clusters should be dissimilar. Problems of this type arise in a variety of disciplines ranging from sociology and psychology, to commerce, biology and computer science, and algorithms for tackling them continue to be the subject of active research. Consequently, there exists a multitude of clustering methods, which differ not only in the principles of the algorithm used (which of course determine runtime behaviour and scalability), but also in many of their most basic properties, such as the data handled (numerical vs. categorical and proximity data), assumptions on the shape of the clusters (e.g., spherically shaped), the form of the final partitioning (hard vs. fuzzy assignments) or the parameters that have to be provided (e.g., the correct number of clusters).The basic ant algorithm starts with an initialization phase, in which

(i) All data items are randomly scattered on the grid

(ii) Each agent randomly picks up one data item and

(iii) Each agent is placed at a random position on the grid.

Subsequently the sorting phase starts: this is a simple loop, in which

(i) one agent is randomly selected;

(ii) The agent performs a step of a given step size (in a randomly determined direction) on the grid and

(iii) The agent (probabilistically) decides whether to drop its data item.

In the case of a 'drop'-decision, the agent drops the data item at its current grid position (if this grid cell is not occupied by another data item), or in the immediate neighbourhood of it (it locates a nearby free grid cell by means of a random search). It then immediately searches for a new data item to pick up. This is done using an index that stores the positions of all 'free' data items on the grid. The agent randomly selects one data item i out of the index, proceeds to its position on the grid, evaluates the neighbourhood function f*(i), and (probabilistically) decides whether to pick up the data item. It continues this search until a successful picking operation occurs. Only then the loop is repeated with another agent.

For the picking and dropping decisions the following threshold formulae are used:

$$p_{pick}(i) = \left(\frac{k^*}{K^* + f(i)} \right)^2 \tag{3.1}$$

$$p_{drop}(i) = \begin{cases} 2\, f(i) & \text{if } f(i) < k^- \\ 1 & \text{otherwise} \end{cases} \tag{3.2}$$

f (i) is a neighborhood function as given in Equation (3.3)

$$f(i) = \max\left(0, \frac{1}{\sigma^2} \sum_{j \in L} \left(1 - \frac{d(i,j)}{\alpha} \right) \right) \tag{3.3}$$

where $d(i, j) \in [0, 1]$ is a measure of the dissimilarity between data points i and j , $\alpha \in [0, 1]$ is a data-dependent scaling parameter, and σ^2 is the size of the local neighborhood L. Handl and Meyer in 2002 proposed an extension of this algorithm where the parameter α is adaptively updated during the execution of the algorithm. This algorithm is given in Figure 3.2.

After discovering the URLs, each URL group is assigned a unique number. Then each user's requested URL is substituted with its corresponding

number. The result of this step is a file that contains records, each record representing a navigational sequence of users in numbers. During pattern discovery, each user's navigational sequence is defined as an array. The size of the array is equal to the number of groups identified in the previous step. The element 'i' is 1 if the related user has seen one of the pages in group 'i', otherwise it is 0. Dissimilarity of two sequences s1 at point i and s2 at point j in the grid is computed through the following Equation (3.4).

$$d(i,j) = \frac{\sqrt{\sum_{k=1}^{N} (sl_k - S2_k)^2}}{N} \qquad (3.4)$$

where N is the number of groups and d(i,j) becomes 1 when two sequences do not have any similar elements, and becomes 0 when they are exactly the same.

```
(1)  Procedure Lumer and Faieta
(2)     randomly scatter data items on the toroidal grid
(3)     randomly place agents on the toroidal grid
(4)     for t = 1 to max_iterations
(5)        j = random agent
(6)        move agent j randomly by stepsize grid cells
(7)        l = does agent j carry a data item?
(8)        e = is agent j's grid position occupied by a data
           item?
(9)        if (l = TRUE) and (e = FALSE) then
(10)          i = data item carried by agent j
(11)          drop = (random() ≤ Pdrop(i))   // see
              equations (2) and (3)
(12)          if drop = TRUE then
(13)             Let agent j drop data item i at its current
                 position
(14)          end if
(15)       end if
(16)       if (l = FALSE) and (e = TRUE) then
(17)          i = data item at agent j's grid position
(18)          pick = (random() ≤Ppick(i)) // see equations
              (1) and (3)
(19)          if pick = TRUE then
(20)             let agent j pick up data item i
(21)          end if
(22)       end if
(23)    end for
(24) end procedure
```

Figure 3.2: Ant based algorithm

The output of this program is a grid that contains numbers: >-1 and =-1 indicates if there is/is not a data item, respectively. So, clusters should be extracted according to these numbers and size of the local neighborhood, σ^2.

This process is illustrated in Figure 3, which shows the positions of the data points in different phases of running this algorithm. Figure 3.3a shows the distribution of data points at the first step of program execution. As the execution continues, Figures 3.3b, c and d shows the results after 200, 400 and 600 iterations, respectively. Clusters are created through moving of the ants.

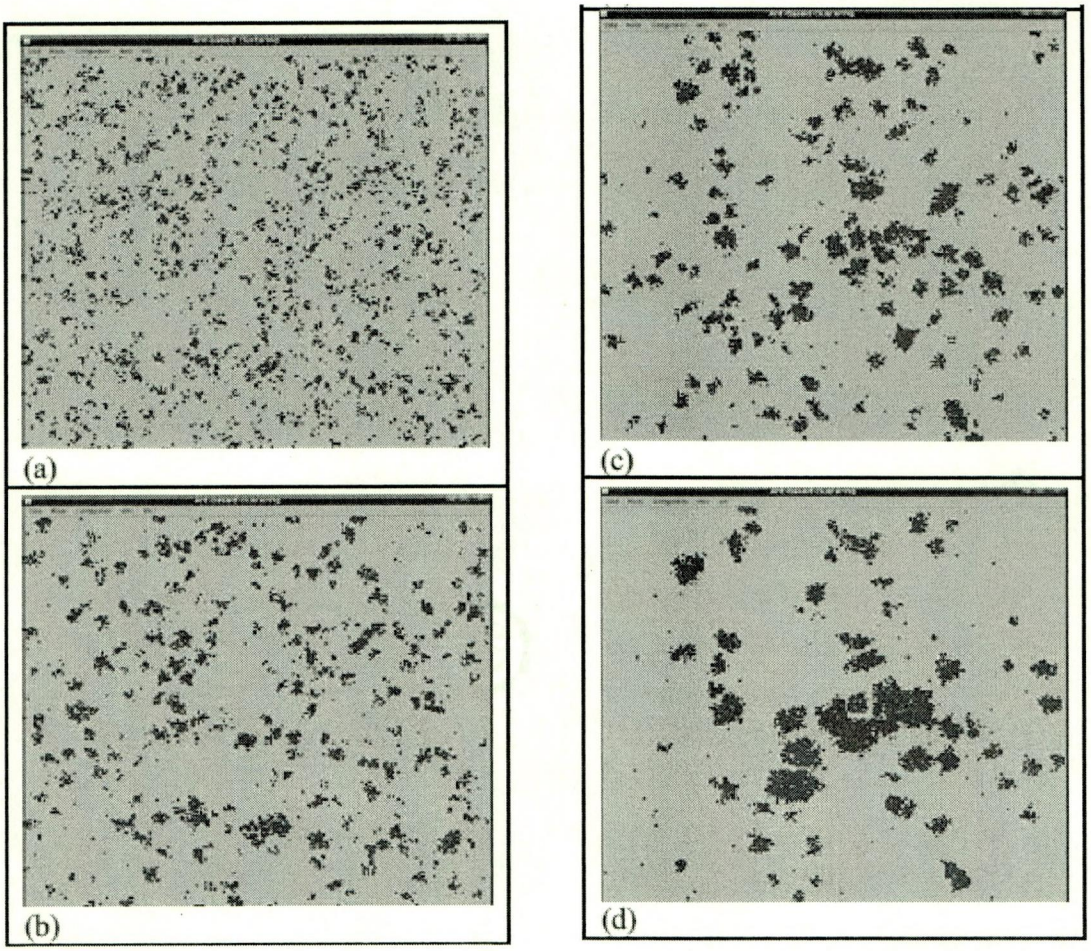

Figure 3.3: Demo sequence of ant-based clustering of user sequences

The above part described the offline phase of the proposed system and is given in Figure 3.4.

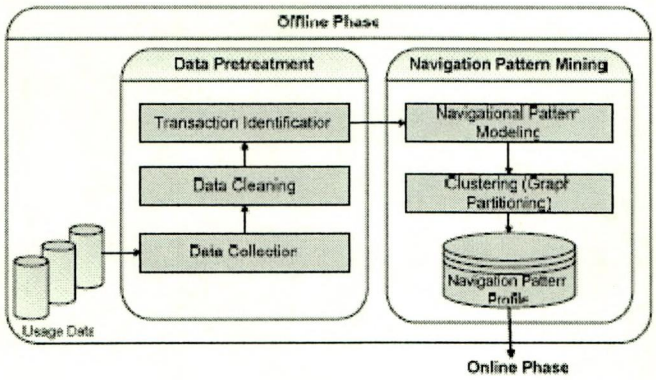

Figure 3.4 Offline Phase

During the online phase, when a new request arrives at the server, the URL requested and the session to which the user belongs are identified, the underlying knowledge base is updated, and a list of suggestion is appended to the requested page. The online phase is shown in Figure 3.5.

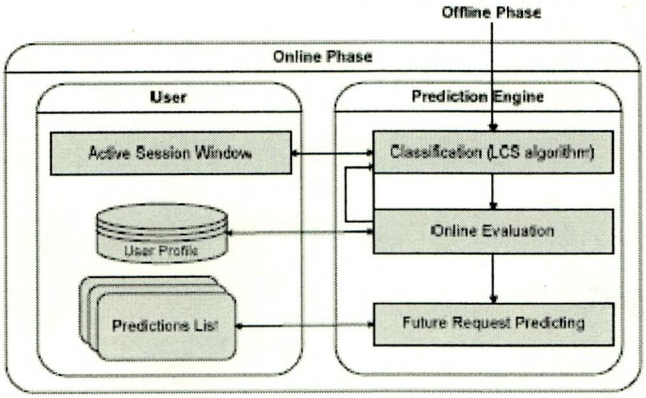

Figure 3.5: Online Phase

3.7. LONGEST COMMON SUBSEQUENCE PROBLEM (LCS)

The main aim of Longest Common Subsequence (LCS) problem is to find the longest subsequence common to all sequences in a set of sequences (often just two). This method is illustrated below.

3.7.1. Prefixes

The sub problems become simpler as the sequences become shorter. Shorter sequences are conveniently described using the term, prefix. A prefix of a sequence is the sequence with the end cut off. Let S be the sequence

(AGCA). Then, a prefix of S is the sequence (AG). Prefixes are denoted with the name of the sequence; followed by a subscript to indicate how many characters the prefix contains. The prefix (AG) is denoted S2, since it contains the first 2 elements of S. The possible prefixes of S are

S1 = (A)

S2 = (AG)

S3 = (AGC)

S4 = (AGCA)

The solution to the LCS problem for two arbitrary sequences, X and Y, amounts to constructing some function, LCS(X, Y), that gives the longest subsequences common to X and Y. That function relies on the following two properties.

3.7.2. First Property

The first property states that if two sequences X and Y both end with the same element, then their LCS will be found by removing the last element and then finding LCS of the shortened sequence. For example, let X = 'BANANA' and Y='ATANA'. The last character for both sequences is 'A' which is removed and repeating the same procedure removes N and A. Thus the removed sequence will be 'ANA' and the shortened sequence will be 'BAN' and 'AT', The first removed character (A) is appended at the end of the removed sequence, thus becoming 'AANA', which is the LCS of the original sequences and is given in terms of prefixes as,

$$LCS\ (X_n, Y_m) = (LCS\ (X_{n-1}, Y_{m-1}), x_n) \qquad (3.5)$$

Where the comma indicates the following element, x_n, is appended to the sequence. Thus, the LCS for X_n and Y_m involves determining the LCS of the shorter sequences, X_{n-1} and Y_{m-1}.

3.7.3. Second Property

The second property is used when the two sequences X and Y does not end with the same symbol. Then, the LCS of X and Y is the longest sequence of LCS (X_n,Y_{m-1}) and LCS(X_{n-1},Y_m). To understand this property, consider the two following sequences:

X = ABCDEFG (n elements)

Y = BCDGK (m elements)

The last character of the LCS of these two sequences either ends with a G (the last element of sequence X) or doesn't.

Case 1: the LCS ends with a G

If the LCS ends with G, then it cannot end with K. Thus K can be removed from sequence Y. If K is in the LCS, it would be its last character; as a consequence K is not in the LCS and can then write LCS(X_n,Y_m) = LCS(X_n, Y_{m-1}).

Case 2: the LCS does not end with a G

Then remove G from sequence of X and LCS (X_n,Y_m) = LCS (X_{n-1}, Y_m).

In any case, the LCS is either LCS (X_n, Y_{m-1}) or LCS (X_{n-1}, Y_m), which are both common subsequences to X and Y. LCS(X, Y) is the longest. Thus its value is the longest sequence of LCS (X_n, Y_{m-1}) and LCS (X_{n-1}, Y_m).

From the above explanation, the LCS can be formulated as below.

Let two sequences be defined as follows: X = (x_1, x_2...x_m) and Y = (y_1, y_2...y_n). The prefixes of X are $X_{1, 2,...m}$; the prefixes of Y are $Y_{1, 2,...n}$. Let LCS (X_i, Y_j) represent the set of longest common subsequence of prefixes X_i and Y_j. This set of sequences is given by the following.

$$LCS(X_i, Y_{i)} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ (LCS(X_{i-1}, Y_{j-1}), x_i) & \text{if } x_i = y_i \\ \text{longest}(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_i \end{cases} \quad (3.6)$$

To find the longest subsequences common to X_i and Y_j, the elements x_i and y_i are compared. If they are equal, then the sequence LCS (X_{i-1}, Y_{j-1}) is extended by that element, x_i. If they are not equal, then the longer of the two sequences, LCS (X_i, Y_{j-1}), and LCS (X_{i-1}, Y_j), is retained. If they are both the same length but not identical then both are retained. A worked out example of LCS is given in http://en.wikipedia.org/ wiki/Longest_common_subsequence _problem.

3.8. ONLINE PHASE – LCS APPROACH

The main objective of prediction engine in this part of architecture is to classify user navigation patterns and predicts users' future requests. For this purpose, an algorithm is used to classify current user activity. A fixed-size sliding window over current active session is used to capture the current user's activities. In order to classify user session windows, we look for the cluster that includes the larger number of pages in that session. For this purpose we a Longest Common Subsequences (LCS) algorithm is used to classify current user activates.

From the clustering results, we have a set of clusters $np' = <np_1, np_2, \ldots np_n>$ which $np_i = <P_1, P_2, \ldots, P_k>$ is the set of k web pages as a user navigational patterns for each $1 \leq i \leq n$. Sequence $W' = <P_1, P_2, \ldots P_m>$ is a current active session and m is size of active session window.

Before classifying an active session to construct the prediction list, the pages in active session windows is sorted based on values stored in the co-occurrence matrix M. After this step, for building the prediction list, the system must find the cluster based on LCS algorithm. After applying this algorithm,

the system finds a cluster with highest degree of LCS in respect to sequence W'.

When the prediction engine finds more than one cluster based on LCS algorithm, then the prediction engine selects a cluster in such a way that, if the difference between positions of last elements of longest common subsequence founded in the cluster and the position of first element of this sequence is minimized, the system must chooses this cluster. In this module, if the first page in the next user activity is different with prediction list, it needs again to classify with new user activities.

3.9. CHAPTER SUMMARY

In this chapter, the various techniques used in the stages of navigation pattern discovery were explained. The results obtained while using Ant-based clustering and clustering followed by LCS algorithm is discussed in the next chapter, Results and Discussion.