Chapter 4

A MECHANISM FOR PREDICTIVE WEB PREFETCHING USING WEB MINING

4.1 INTRODUCTION

Owing to increasing network bandwidth and computing power, the usage of internet has grown at the breakneck rate. Over the years, there has been a significant increase in the number of users who access Internet through high speed DSL but still the access latencies perceived by them is high. Therefore, a significant reduction in web latency assumes importance for the users of the Internet and also for the Internet service providers who desire to increase the web surfing speed.

Web latency can be reduced either by pushing the bandwidth at the expense of incurring higher costs or by implementing better technological solutions such as introduction of cache(s) at the server, proxy or the client side. Web caching is an effective technique to alleviate the server bottleneck and reduce network traffic, thereby reducing network latency. It is the automatic creation of temporary copies of information residing on computers other than host servers in order to make this information readily available to people around the world.

Caching can be introduced at the various levels such as client, proxy, and server [116, 117 and 118]. Effective client and proxy caches reduce the client perceived latency. In fact, the required page desired by the user, if had been fetched earlier is made available from the local cache instead of importing it from the web server. This further reduces the server load and the network traffic in terms of reduction of the packets travelling across the network. Although web cache schemes reduce the network and I/O bandwidth consumption, they still suffer from a low hit rate, stale data and inefficient resource management. The benefits gained through caching can be limited when the web resources
tend to change very frequently e.g. when a web-site contains dynamic pages even if its local copy is available in the cache. Nevertheless, an efficient web cache management increases overhead so much so that it causes a major web site crash also known as the *Slashdot effect* [119].

*Prefetching* of web pages is another potential research area that can significantly reduce the web access latency. It refers to the process of predicting a client’s future requests for web objects and bringing those objects into the cache before it is explicitly requested by the user. Compared with web cache schemes, web prefetch schemes focus on the spatial locality of objects especially when the current requests are causally related with previous requests. The major advantage of using prefetching is that it prevents bandwidth over-utilization. However without a carefully designed prefetching scheme, it may happen that several already transferred web pages might never be requested by the client. This would definitely result in the bandwidth wastage.

In this work, a web prefetching framework is being introduced that predicts web pages likely to be accessed by the user. Therefore, it fetches the predicted web pages from web servers and loads them into the proxy server’s cache while the user may be busy in performing some other tasks. Subsequently whenever the user desires a web page then if prefetched, the web page is delivered to the user immediately. Various data mining techniques can be used to accomplish this task like association rules, clustering and markov predictors. Each has its own strengths and weaknesses as listed below:

1. Association Rules are probably the most elementary data mining technique that are used to find associations among web pages that frequently appear together in users’ sessions. The main limitation of association rules is that they tend to generate many rules, which result in contradictory predictions for a user session.

2. Clustering techniques look for groups of similar items among large amount of data based on general idea of distance function. The similarity between the groups computed thereof becomes the basis for the prediction of the items likely to be needed by the user. However, this process leads to decreased precision because it does not use all the pages directly.
3. Markov models are popularly used in the identification of patterns based on the sequence of previously accessed items. However, Markov model implementations have been hindered due to the fact that low order Markov models lack accuracy as their models do not use enough history. On the other hand, high order Markov models incur high state space complexity thus involving larger cost.

Each of the above techniques has been widely used for predicting web page accesses and owing to their limitations individually, these techniques are inadequate towards efficient predictions of web pages. Since, the purpose behind implementing such tools for predicting web page access was to achieve reliable prediction accuracy, in this work various data mining techniques have been integrated towards improvement in the prediction accuracy and reducing user perceived latency at the web.

4.2 PROPOSED ARCHITECTURE FOR PREDICTIVE WEB PREFETCHING

A framework for the web prefetching mechanism has been developed that consults the past history of the user session behaviour maintained in the logs to predict and prefetch the pages even before they have been accessed by the user. The framework can be deployed either at the proxy side or the search engine side as explained below:

- One for the particular set of users who interact with the Internet through proxy servers. The Predictive Prefetching Engine (PPE) has been introduced at the proxy level and hence the name Proxy side Predictive Prefetching Engine (PPPE) [120].
- Other is the generalized version of this PPE, which can be used by the larger set of users who belong to same organization and thus have similar set of browsing patterns. For this to happen PPE is introduced at the search engine side and hence the name Search engine side Predictive Prefetching Engine (SPPE) [121].

The basic organization of the PPE is same for both the frameworks and is explained in the following subsections.
4.2.1 PROXY SIDE PREDICTIVE PREFETCHING ENGINE (PPPE)

Internet is a client server architecture wherein a client sends his request for a resource over the WWW to a server. The server responds by serving the request. The session involves the exchange of messages and protocols. However, due to exponential increase of WWW, there are a large number of clients that interact with servers through millions of networks connected with each other leading to a significant increase in the WWW latency and traffic on the net.

Since a proxy server sits between a web browser and a web server, it is a potential tool that can be suitably employed to reduce the WWW latency i.e. it can intercepts all requests to the web server to see if it can fulfil the requests by itself. If not, then only it may forward the request to the web server. In fact, the proxy servers can be employed to achieve two main purposes:

- **Reduce latency:** A Proxy server saves the results of all the requests from various clients for a certain amount of time. For instance, consider a case where both users X and Y access the www through a proxy server. Let us assume that user X requests for a certain web page say Page 1. Sometime later, user Y also requests the same page. Instead of forwarding the request to the web server where page 1 actually resides, which can be a time-consuming operation, the proxy server simply returns this page from its cache where all the downloaded pages are retained before being over written by new arrivals. Since proxy server is often on the same network as the user, this is a much faster operation, thereby reducing the perceived latency to some extent.

- **Filter unwanted Requests:** Proxy servers can also be used to filter unwanted requests. For example, a company might use a proxy server to prevent its employees from accessing a specific set of Web sites.

The www latency can be further reduced if the behavior of the user can be predicted and accordingly the predicted pages are prefetched and stored temporarily in the cache of the proxy server. As soon as the user asks for a page, the request can be fulfilled if the requested page is available in the cache.
Considering this added advantage of proxy server, a prediction engine called *Predictive Prefetching Engine* (PPE) has been proposed that resides on the proxy server and processes the past references to deduce the probability of future access for the documents accessed so far. The complete framework of Proxy side Predictive Prefetching Engine (PPPE) shown in Fig. 4.1 constitutes the following components and processes:

![Fig. 4.1 Framework for Proxy side Predictive Prefetching Engine (PPPE)](image)

1. **Proxy server log**: The records of all the users/clients that send requests to the server are kept on the web logs which are used to form the mine able warehouse. These warehouses are used to track the user activity. Since a proxy
server sits between the client and the web server, it is comparatively easy to manage than the web server logs especially when it has to maintain the log for limited clients.

2. **Transaction Preprocessor**: Since in the proposed work, the log is being maintained on the proxy side, the transaction preprocessor operates on these proxy logs to accomplish preprocessing. Preprocessing involves the following tasks:
   
i. *Reduction of Search Space*: The foremost is to reduce the search space for mining which is done by cleaning the proxy log for any unwanted records. It may include clearing the log from the irrelevant items like the image files (GIF and JPEG) and java script files (JS) etc as these do not contribute for the patterns relevance.
   
ii. *User and Session Identification*: A user session is defined as the sequence of requests made by the single end user during a visit to a particular site. Within a single session, a user may follow links to several pages that belong to the similar pattern but during the same session, it may also be possible that the user might visit some other pages that do not belong to the same pattern. It may also be possible that the documents are interleaved in the session.
   
iii. *Path completion*: It forms the next step of preprocessing within a session. It is necessary as it may happen that some of the important accesses are not recorded in the log due to use of cache. After all the preprocessing is done, the cleaner version of the proxy log is formed called Data mart.
   
3. **Data Mart**: Data mart acts as a database on which various data mining operations operate for generating the rules.

4. **Rule Generator**: It extracts the information from the data mart and applies the various data mining operations e.g. association rules, sequential patterns, markov predictors etc. to generate the rules for prediction.

5. **Knowledge Base (KB)**: The various rules formed by the rule generator forms the part of this repository.
6. **Page Loader:** For a given request made by the user, page loader consults the rules of the KB and if the user’s requested pages exist in the heads of the rules, then the pages present in the body of those particular rules are prefetched. For instance, the $k^{th}$ entry in the knowledge base may have the following format:

$$ R_k: D_i \Rightarrow D_j; \quad \text{if document } D_i \text{ has been requested then prefetch document } D_j. $$

Similarly, the $n^{th}$ entry in the knowledge base may have the following format:

$$ R_n: D_j \Rightarrow D_k; \quad \text{if document } D_j \text{ has been fetched then prefetch document } D_k \text{ also.} $$

This method follows the forward chaining in the knowledge base till the time no more rules can be fired. To prevent increase in the network traffic due to the chaining-activation process, all the prefetched documents are stored in the datastructure maintained in proxy cache as the *Hintlist*.

### 4.2.2 SEARCH ENGINE SIDE PREDICTIVE PREFETCHING ENGINE (SPPE)

The general architecture of a common web search engine contains a front-end process and a back-end process. In the front-end process, the user enters the search keywords into the search engine interface, which is usually a web page with an input box. The application then parses the search request into a form that the search engine can understand, and then the search engine executes the search operation on the index files. After ranking, the search engine interface returns the search results to the user. In the back-end process, a spider or robot fetches the Web pages from the Internet, and then the indexing subsystem parses the Web pages and stores them into the index files. The search engine retrieves the web pages according to the user query. Since *relevancy* is a subjective term, the search results may have varying degree of relevancy for different set of users. Given this fact, there is an opportunity to significantly improve the relevancy of
search results for a well defined set of users (example, employees of the same organisation), whose search habits are largely homogenous.

The proposed work introduces the Predictive Prefetching Engine (PPE) which sits behind the search engine interface. The intent of introducing the PPE [38] is that it will increase the relevancy of the pages returned by the search engine according to the demand of the particular set of users which are termed as group clients. PPE also prefetches the pages if it lies in the rule-database that is generated by applying the various data mining operations on the group-client-log. This log is maintained by the search engine on the request of the various organisations which are assigned a particular set of IP addresses by the Internet Service Providers. The interaction of the PPE with the user and the process of retrieving the relevant web pages from the WWW is explained with the complete framework as given in Fig 4.2.
The various components of SPPE are as:

1. **Search Engine Interface:** It is the part of the search engine’s front end and is basically a web page with the input box. The user enters his/her query containing the keywords into this input box and hits the search button.

2. **IP Matcher:** The job of this process is to extract the IP address from the query coming from a particular user. This IP address is then matched with the particular range of IP addresses for which different *Group-Client-Agents* (GCAs) are defined. Once the GCA is identified, it gets activated.

3. **Group-Client-Agent (GCA):** As the name suggests, it is an agent. It plays the crucial role as it works on PPE. There are $n$ GCA’s for $n$ group-clients and hence each GCA has its own corresponding PPE to work upon. One group-client refers to a group of users within one organisation. Every organisation is assigned a unique set of IP addresses. These IP addresses forms a part of one group-client.

4. **Group-Client-Log (GC-Log):** This log is maintained by the search engine on the Group-client’s request. The format of the log is same as that of the web server maintained by the search engine and contains every entry from that particular group-client. Each record in the log file contains the client’s IP address, the date and time the request is received, the requested object and some additional information such as protocol of request, size of the object etc.

5. **Clean Log:** This is the cleaner version of the GC-log formed by removing all the image files like .jpg and .gif from it as they yield no productive information about the path followed by the user in a particular session.

6. **RST Clusters:** The clean log is then operated upon by the clustering technique known as Rough Set Clustering. The purpose of clustering the sessions is to reduce the search space for applying the various datamining operations. RST
operates on the principle of indiscernibility which is defined as equivalence between the objects. RST is chosen as the clustering technique as it aids in decision making in the presence of uncertainty. The result of applying RST is the lower approximation set which contains all the user sessions which definitely contain the target set.

7. **Rule Generator**: This phase takes as input the lower approximation set and applies the k-order markov predictors. The order k is chosen dynamically everytime depending on the size of the lower approximation set. The output of this phase is the rules of the form $D_i \Rightarrow D_j$.

8. **Rule Repository**: The rules formed in the last phase are then stored in this repository.

9. **Database**: This database contains the URLs of all the pages whose references are stored in the rule repository. The database is enriched by the URLs of rules from all the $n$ PPE’s.

10. **Page loader**: Its job is to prefetch the pages populated in the hint-list by the GCA onto the client’s cache.

Since the proposed framework consists of many complex activities, the overall processing of the transactions from the calculation of user accesses to the generation of rules to the Prefetching of the pages into the cache is divided into three main phases as shown in Fig 4.3.

The step wise working of these phases is as follows:

1. In the first phase, a novel approach for clustering the web user sessions using Rough Set Clustering is proposed [122]. Among the many other clustering techniques available, RST is chosen in order to aid decision making in the presence of uncertainty [123 and 124]. It works on the user transactions obtained from the GC-log after performing the various preprocessing steps.

2. The second phase is the Rule Determiner phase. In this, a novel algorithm for the determination of the rules has been proposed using the K-order Markov Predictors [125]. These rules will let know which pages to be prefetched. The job of
determining the rules is performed by the Rule Generator component which are then stored in the Rule Repository component of the proposed framework.

3. Third phase is the Rule Activator phase. After the determination of the rules, the next requirement is the rule activation mechanism.

The following sections describe the algorithmic detail of each phase of the proposed work.

4.3 PHASE I: CLUSTERING USER SESSIONS
The user sessions are the most crucial part of web usage mining process as they help in analyzing the web user’s behaviour. Therefore, it becomes very important to identify the user sessions efficiently. The web user’s behaviour is stored in the web logs. These logs can be maintained by the proxy servers or by the web servers or by both of them. To mine the useful data out of these logs, preprocessing is done. The preprocessing of log files is aimed to the preparation of the web data in order to mine significant usage patterns. These usage patterns are divided into user sessions. The objective of user session is to separate independent accesses made by different users or by the same user at distant points in time. These user sessions are then acted upon by the data mining technique called clustering. Although there exist a large number of techniques to correctly work upon these user sessions, but most of them require the complete database. Whereas, it is impossible that the database is complete at every instance as due to the use of cache by some systems, entry of few links from the web log may be missing or the user may have moved back in the browser, which would also result in the lack of entry in the web log. Rough Set Clustering has been used in this work that helps in decision making due to the uncertainty present in the database. The detailed discussion on these two functional components is given in the next subsections.

4.3.1 PREPROCESSING OF WEB LOG

Preprocessing is a complete process which includes several steps of data cleaning, user identification and session identification and has been explained in greater detail in chapter 2.

- **Data cleaning module**: The data cleaning module is intended to clean web log data by deleting irrelevant and useless records in order to retain only usage data that can be effectively exploited to recognize users’ navigational behaviour. Since web log data contain all the user interactions, they often comprise of noisy data which do not yield to any productive information. By removing these unwanted records, the storage size required for storing such files is greatly reduced. The noisy data may depend on the choice of useful data required. In our case, following records are removed as they don’t yield the required information.
1. Requests which contain the `.jpeg`, `.bmp` or any other extension that shows that it is an image file are removed. Generally, users do not make any explicit request for such files. However, they get downloaded with the web page due to the HTML tags. Since web usage mining tracks for the user trails to be followed and since such graphics are not explicitly requested by the users, they are removed.

2. Requests with access method different from “GET” can also be removed as the requests containing the value different from “GET” in the field of access method do not refer to the explicit requests from the user. They are often concerned with the CGI accesses, properties of the server or the visits of robots etc., all of which don’t yield any productive information. Hence, these are considered non-significant and thus are removed from the log file.

3. Requests which represent the HTTP error code are also removed. A status with value 200 shows a succeeded request. A status with value different from 200 represents a failed request and is thus removed.

4. A log file may contain a number of records corresponding to the requests originated by the web robots. Web robots are the programs that automatically download complete web sites by following every hyperlink on every page within the web site in order to update the index of the search engine. Requests created by web robots are not considered as usage data and thus have to be removed. In order to identify such requests, two methodologies can be adapted.

   a. Firstly, all records that contain the ‘Robot.txt’ in the requested URL are straight-away removed.

   b. Second methodology is based on the fact that robots retrieve web pages in the automatic and exhaustive fashion. Hence they are characterized by very high browsing speed. The browsing speed can be calculated as:
Browsing speed = Total number of pages visited / Total time spent to visit those pages

Thus, Browsing speed for different IP addresses is calculated and those IPs whose browsing speed exceed the user defined threshold (pages/second) are regarded as made by robots and are consequently removed. Table 4.1 (a) and (b) shows sample web log before and after data cleaning steps of preprocessing.

Table 4.1 (a) Sample Raw Web Log before Preprocessing

<table>
<thead>
<tr>
<th># IP Address</th>
<th>User id</th>
<th>Time</th>
<th>Method/URL/Protocol</th>
<th>Status</th>
<th>Size</th>
<th>Referrer</th>
<th>Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>123.456.78 .9</td>
<td>-</td>
<td>[25/Apr/2006:03:04:41-0500]</td>
<td>&quot;GET A.html HTTP/1.0&quot;</td>
<td>200</td>
<td>3290</td>
<td>-</td>
<td>Mozilla/3.04 (Win98, I)</td>
</tr>
<tr>
<td>123.456.7.8.9</td>
<td>-</td>
<td>[25/Apr/2006:03:05:41-0500]</td>
<td>&quot;POST A.html HTTP/1.0&quot;</td>
<td>403</td>
<td>4321</td>
<td>-</td>
<td>Mozilla/3.04 (Win98, I)</td>
</tr>
<tr>
<td>123.456.78 .9</td>
<td>-</td>
<td>[25/Apr/2006:03:05:39-0500]</td>
<td>&quot;GET A.html HTTP/1.0&quot;</td>
<td>200</td>
<td>4130</td>
<td>-</td>
<td>Mozilla/3.04 (Win98, I)</td>
</tr>
<tr>
<td>123.456.78 .9</td>
<td>-</td>
<td>[25/Apr/2006:03:05:39-0500]</td>
<td>&quot;GET A.gif HTTP/1.0&quot;</td>
<td>200</td>
<td>4130</td>
<td>-</td>
<td>Mozilla/3.04 (Win98, I)</td>
</tr>
<tr>
<td>123.456.78 .9</td>
<td>-</td>
<td>[25/Apr/2006:03:06:02-0500]</td>
<td>&quot;GET A.html HTTP/1.0&quot;</td>
<td>200</td>
<td>5096</td>
<td>B.html</td>
<td>Mozilla/3.04 (Win98, I)</td>
</tr>
<tr>
<td>123.456.78 .9</td>
<td>-</td>
<td>[25/Apr/2006:03:07:42-0500]</td>
<td>&quot;GET A.html HTTP/1.0&quot;</td>
<td>200</td>
<td>3290</td>
<td>-</td>
<td>Mozilla/3.01 (X11, I, IRIX6.2, IP22)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>123.456.78 .9</td>
<td>-</td>
<td>[25/Apr/2006:03:07:55-0500]</td>
<td>&quot;GET b.jpg HTTP/1.0&quot;</td>
<td>200</td>
<td>250</td>
<td>A.html</td>
<td>Mozilla/3.01 (X11, I, IRIX6.2, IP22)</td>
</tr>
<tr>
<td>123.456.78 .9</td>
<td>-</td>
<td>[25/Apr/2006:03:09:50-0500]</td>
<td>&quot;GET A.html HTTP/1.0&quot;</td>
<td>200</td>
<td>8140</td>
<td>L.html</td>
<td>Mozilla/3.04 (Win98, I)</td>
</tr>
</tbody>
</table>
### Table 4.1 (b) Sample Web Log after Preprocessing

<table>
<thead>
<tr>
<th>IP Address</th>
<th>User id</th>
<th>Time</th>
<th>Method/ URL/ Protocol</th>
<th>Status</th>
<th>Size</th>
<th>Referrer</th>
<th>Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>123.456.7 8.9</td>
<td>-</td>
<td>[25/Apr/2006: 03:04:41 - 0500]</td>
<td>&quot;GET A.html HTTP/1.0&quot;</td>
<td>200</td>
<td>3290</td>
<td>-</td>
<td>Mozilla/3.04 (Win98, I)</td>
</tr>
<tr>
<td>123.456.7 8.9</td>
<td>-</td>
<td>[25/Apr/2006: 03:05:39 - 0500]</td>
<td>&quot;GET A.html HTTP/1.0&quot;</td>
<td>200</td>
<td>4130</td>
<td>-</td>
<td>Mozilla/3.04 (Win98, I)</td>
</tr>
<tr>
<td>123.456.7 8.9</td>
<td>-</td>
<td>[25/Apr/2006: 03:06:02 - 0500]</td>
<td>&quot;GET A.html HTTP/1.0&quot;</td>
<td>200</td>
<td>5096</td>
<td>B.html</td>
<td>Mozilla/3.04 (Win98, I)</td>
</tr>
<tr>
<td>123.456.7 8.9</td>
<td>-</td>
<td>[25/Apr/2006: 03:07:42 - 0500]</td>
<td>&quot;GET A.html HTTP/1.0&quot;</td>
<td>200</td>
<td>3290</td>
<td>-</td>
<td>Mozilla/3.01 (X11, I, IRIX6.2, IP22)</td>
</tr>
</tbody>
</table>

- **User Identification:** The user’s IP address is not sufficient for identifying a user. Many users can be assigned same IP address. Also many users can have access to the same computer. Cookies can be used for better user identification but are not brought into use due to privacy reasons. In such cases, User IDs are brought into use. The following criteria is used to identify the users:

  1. If two records have different IP address, they are distinguished as two different users else if they have same IP address, then the user agent field is checked.

  2. If the browser and operating system information in the user agent field is different in two records then they are identified as different users.

- **Session Identification:** It is the next step after identifying the users. A session is a sequence of activities that the user makes during one visit to the web site. The session identification aims to divide the page accesses of each user into individual sessions. There are several heuristics available but the most widely and significantly used is the time-based-heuristic by R. Cooley [128]. According to
the authors, if the time difference between the two accesses, even from the same IP, is more than 30 minutes, then it is treated as a new session. Also, if the page stay time gap determined from the timestamp field exceeds 10 minutes, then also it is treated as a new session.

Once the sessions are identified, the next step followed in the phase is the clustering of the sessions.

4.3.2 CLUSTERING USER SESSIONS

Once the user sessions are identified, the next major step is to form the clusters. Clustering is one of the data mining techniques which bind together the similar objects. Web data clustering is the process of grouping web data into ‘clusters’ so that similar objects are in the similar classes and dissimilar objects are in different classes. Clustering is different from classification as groups are not predefined. In simple words, clustering is used to increase the intra-group interaction and decrease the inter-group interaction. Amidst the uncertainty of the web data, Rough Set Clustering is the most appropriate clustering technique. It aids decision making in the presence of uncertainty. It classifies imprecise, uncertain or incomplete information expressed in terms of data acquired from experience. A brief discussion on Rough Set Clustering is given in next section.

4.3.2.1 ROUGH SET CLUSTERING

In RST, a set of all similar objects is called an elementary set, which makes a fundamental atom of knowledge. Any union of elementary sets is called a crisp set and other sets are referred to as rough set. As a result of this definition, each rough set has boundary-line elements. For example, some elements cannot be definitively classified as members of the set or its complement. In other words, when the available knowledge is employed, boundary-line cases cannot be properly classified. Therefore, rough sets can be considered as uncertain or imprecise. Upper and lower approximations are used to identify and utilize the context of each specific object and reveal relationships between objects. The upper approximation includes all objects that possibly belong to the concept
while the lower approximation contains all objects that surely belong to the concept. The nomenclature used by RST for clustering the web user sessions is as:

**Rough Set:** A rough set, first described by Zdzisław I. Pawlak, is a formal approximation of a crisp set (i.e., conventional set) in terms of a pair of sets which give the *lower and the upper approximation* of the original set.

**Information system:** Formally, an information system is a pair $A = (\mathbb{U}, A)$ where $\mathbb{U}$ is a non-empty, finite set of objects called the universe and $A$ is a non-empty, finite set of attributes on $\mathbb{U}$. With every attribute $a \subseteq A$, a set $V_a$ is associated such that $a: \mathbb{U} \rightarrow V_a$. The set $V_a$ is called the **domain** or **value set of attribute** $a$.

**Indiscernibility:** It is core concept of RST and is defined as equivalence between objects.

**Equivalence Relation:** Objects in the information system about which we have the same knowledge form an equivalence relation. The equivalence relation has the following properties:

If a binary relation $R \subseteq X \times X$

- which is reflexive (i.e. an object is in relation with itself $xRx$),
- symmetric (if $xRy$ then $yRx$)
- and transitive (if $xRy$ and $yRz$ then $xRz$)

is called an **equivalence relation**.

Formally any set $B \subseteq A$, there is associated an equivalence relation called **B-Indiscernibility** relation is defined as follows:

$$\text{IND}_A(B) = \{(x, x') \in \mathbb{U}^2 | \forall a \in B \ a(x) = a(x')\}$$

If $(x, x') \in \text{IND}_A(B)$, then objects $x$ and $x'$ are indiscernible from each other by attributes from $B$. 

85
Equivalence relations lead to the universe being divided into equivalence class partition and union of these sets make the universal set.

**Target set:** It is generally supposed by the user.

**Lower Approximation:** It is the union of all the equivalence classes which are contained by the target set. The lower approximation is the complete set of objects that can be positively (i.e., unambiguously) classified as belonging to target set $X$.

**Upper Approximation:** It is the union of all equivalence classes which have non-empty intersection with the target set. It represents the negative region, containing the set of objects that can be definitely ruled out as members of the target set.

The pseudocode for clustering the user-sessions from the web log is explained in the next subsection.

**4.3.2.2 PSEUDOCODE FOR FINDING CLUSTERED SESSIONS**

Once the raw log is cleaned for all the unwanted entries which don’t lead to any constructive information regarding the user’s browsing patterns, they are then treated upon by RST to cluster the user sessions. The pseudocode for the task comprises four modules interconnected to each other and is explained in Fig 4.4.

*Clustering_Approximation* is the main module which takes as input the web log in the form of table and results in the equivalence set. To find the desired result, it calls several other algorithms which are briefly explained as below:

1. *Preprocessing_log* is the module which takes web log as input and starts with a new empty table transaction set. The image files are removed from the web log. The first entry to the Transaction Set (TS in pseudo code) is the first row of the Web Log (WL in pseudo code). Thereafter, the IP address of the 2nd row of WL is compared with the IP address of the 1st row of WL. It also compares the time interval between the two accesses. If the difference
is less than 30 minutes, it implies that both the accesses are of the same session and vice versa. This process is repeated for all the entries of WL. TS is formed accordingly thus separating the different sessions from the WL.

### Clustering Approximation (WL, ES)

Input WL (Web log table)
Output ES (Equivalence set)
begin

### Preprocessing log (WL, TS)

Input WL (web log table)
Output TS (transaction set table)
begin

- j=1;
- Remove all the .jpeg and .gif files
  - TS[j] = WL[i];
  - for (i=0; i<WL.Length; i++)
    - if ((WL.ipaddress[i+1]==WL.ipaddress[i]) && (WL.time[i+1] - WL.time[i] <= 30))
      - TS[j] = TS[j] ∪ WL[i]
    - endif
  - endfor

end

RST (TS, CS) //call Rough Set Clustering algorithm
def

### RST (TS, CS)

Input TS (transaction set table)
Output CS (clustered set)
repeat

- Target_set = { Φ }
- Threshold_calculator (TS, threshold)
  // call threshold calculator algorithm
  for (k=0; k<TS.length; k++)
    - pages_per_session = ∑ TS_k.pages;
    - if(pages_per_session > threshold)
      - Target_set = Target_set ∪ TS_k
    - endif
  - endfor

Matcher(TS, ES)
  // call Matcher algorithm

Lower_approx = Target_set[p] | ES_k ⊆ Target_set;
Upper_approx = Target_set[p] | ES_k ∩ Target_set = Φ

where Target_set has 1,2,..., p elements

forever
Once all the sessions and their entries are marked into the TS, \textit{Preprocessing\_log} calls another module which is rough set clustering algorithm.

2. \textit{RST (TS, CS)} is the module which takes as input the transaction set and outputs clustered set. It takes target set which is empty in the beginning. To fill that target set, threshold value is required. Target set will contain all those sessions whose number of visited pages is more than this threshold and to find this threshold value it calls another module \textit{Threshold\_calculator}. Once the target set is computed, another module
called *Matcher* is called. Its job is to find the equivalence set. From the equivalence set, lower approximation and upper approximation is found using the formula given in the algorithm.

3. *Threshold_calculator (TS, Threshold)* calculates the total number of pages in the various sessions in TS and divides this count by the number of sessions in TS. This value is then used to fill the *target set*.

4. *Matcher (TS, ES)* finds the equivalence set. In the beginning this set is also empty. It gets filled progressively as the matcher finds all the sessions in TS which contains the same number of visited pages and in the contiguous manner.

Simulated example of the application of the proposed mechanism is given in next section.

### 4.3.2.3 SIMULATED EXAMPLE

To find the clustered sessions and in particular those sessions which fall in the Lower Approximation set, the following steps are used:

- Assuming that the step involving *preprocessing the web log* has already been applied. Each user session comprises of the visited web pages in an orderly fashion. Let there be 10 user sessions which emerge from the web log. Table 4.2 lists the sessions which have been shown on the vertical side and visited pages on the horizontal side.
- Apply RST by filling of the Target Set. In the beginning, target set is empty. To fill this target set, threshold value is required for which threshold calculator module is called.

Threshold > Total pages accessed / n (No of Sessions)

- \[
\text{40} / 10
\]
- \[
= 4
\]
Use this threshold value in filling of target set. According to the proposed work, target set takes all the sessions whose number of visited pages is more than the threshold value i.e. all the sessions which have visited more than 4 pages will be part of the target set as shown:

**TARGET SET = \{S1, S4, S5, S8, S9\}**

- Now the *Matcher module* comes into play that finds the *equivalence set* which again is empty in the beginning and gets filled progressively. As per its definition described in the nomenclature in the previous section, following sets form the equivalence set.

**EQUIVALENCE SET: \{{S1, S8}, {S2, S7}, {S3, S6}, {S5, S9}, {S10}\}**

- Now, Lower and Upper Approximation Sets are found. As per the nomenclature, Lower Approximation Set will comprise of all the equivalence classes which are contained by the target set and hence following classes form the Lower Approximation set.

**LOWER APPROXIMATION SET: \{S1, S8, S5, S9\}**
Upper Approximation set is the union of all equivalence classes which have non-empty intersection with the target set and according to the definition:

**UPPER APPROXIMATION SET:** \{\text{NULL}\} as no set qualifies to be in the upper approximation.

Hence, by making use of rough set clustering, those user sessions from the web log have been deduced in which the user spends his quality time. By clustering the important sessions using RST, the web log has been narrowed so that only these sessions could be used as input to *Rule Generator* phase of PPE as shown in both Fig. 4.1 and Fig 4.2. The advantage of narrowing the web log is that the complexity of the PPE has been reduced.

**4.4 PHASE II: RULE DETERMINER**

The Rule Determiner phase calls upon the Rule Generator component of PPPE/SPPE. This component takes the user sessions which are determined by the lower approximation set and results in the rules which help in the prediction of the web pages. The user sessions are applied upon the k-order Markov Predictors. Markov predictors have been consistently used by the researchers to make the prediction of web pages. E.g. *Dependency Graphs* and *Prediction by Partial Match* are the popular examples which make use of markov predictors. The major drawback of all the existing methodologies is that the value of ‘k’ is fixed. The disadvantage of keeping the value of ‘k’ low is that it does not care for enough past history of the user’s browsing session and thus cannot predict efficiently. On the other hand, keeping the value of ‘k’ higher increases the prediction precision but at the same time the amount of data that must be stored in the prediction model increases which increases the cost of the prediction model. Thus, it is important to formulate the value of k such that its value is decided dynamically. In fact, the optimum value of ‘k’ has to be chosen. The proposed work introduces a mechanism to decide upon this value dynamically depending upon the size of data. Here, the minimum threshold that will be used in deciding rules would be half of the maximum number of time a particular sequence of web pages is called i.e. if the maximum time a
particular page sequence called is 6 then minimum threshold to consider other page sequences must be 6/2 = 3 and k would be set at this minimum threshold. So we will make use of 3-order markov predictors to generate the rules.

4.4.1 PSEUDOCODE FOR FINDING RULES USING K-ORDER MARKOV PREDICTORS

Once the user sessions are clustered and fall in lower approximation set, these sessions are then acted upon by the k-order markov predictors. The pseudocode for this is as shown in Fig. 4.5. This pseudocode works upon the lower approximation set as follows:

1. It starts by the application of 1st order markov predictors. This is done by calculating the total individual web pages of all the user sessions that form the part of the lower approximation set. These individual pages then form the row and column of the table and the total links of each web page to the other in the user sessions are noted down.

2. The next step is to find the threshold value that could lead us to the maximum level up to which the markov predictors can be applied on the basis of this available data. To find this threshold value, the maximum value from the table is picked. This value depicts the maximum number of links that any web page has shown. The threshold value is the half of this number. E.g. if the maximum link that any page has shown is 6 then the threshold value chosen is 6/2 i.e. 3. This means that the minimum threshold required for other sequences to survive must be at least half of this maximum value. So for the next level, those sequences are picked from the table who qualify this minimum threshold criterion.

3. In the next level which is the 2nd order markov predictors, the selected sequences form the row and the individual web pages form the column. Again the sequences which cleared the minimum threshold criterion are selected for the next level. This process is followed iteratively till the sequences keep qualifying the minimum threshold criterion.
Algorithm markov (L[r], A[m])
Input L[r] (Lower approximation set)
Output A[m] (Database of rules)

\{
    max = 0;
    for i = 0 to n-1
    \{
        for j = 0 to n-1
        \{
            f[i][j] = freq(P[i]->P[j]);
            if (max < f[i][j])
                max = f[i][j];
        \}
    \}

    threshold = max / 2;
    m = 0;
    for i = 0 to n-1
    \{
        for j = 0 to n-1
        \{
            if (f[i][j] >= threshold)
            \{
                A[m] = P[i] -> P[j];
                m++;
            \}
        \}
    \}

    do
    \{
        l = 0;
        for i = 0 to m-1
        \{
            for j = 0 to n-1
            \{
                f1[i][j] = freq(A[i]->P[j]);
                if (f1[i][j] >= threshold)
                \{
                    l++;
                \}
            \}
        \}

        if (l != 0)
        \{
            for i = 0 to l-1
            \{
                A[i] = B[i];
                m = l;
            \}
        \}
    \} while (l != 0)
\}

Fig. 4.5 Algorithm for Generating Rules
On the completion of this iterative process, the last sets of sequences which come out qualify for the rules for prediction. In case, if two or more rules have same head then the decision has to be taken as to which tail to follow. In such cases, association rules are applied. As shown in the pseudocode, the support and confidence of such rules is calculated and the rules which qualify with 100% support are only considered for prediction. The working of the above proposed mechanism for rule determination is illustrated by the example given in the next section.

4.4.2 EXAMPLE

After applying Rough Set Clustering (RST) over the preprocessed web log, Lower Approximation Set is formed that clusters those user sessions in which the user has spent the quality time. Consider the Lower Approximation Set of three useful sessions S2, S3 and S4 as given in Table 4.3.

<table>
<thead>
<tr>
<th>User Sessions</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>P0, P2, P8, P0, P1, P2, P7, P0, P3, P4, P5, P3</td>
</tr>
<tr>
<td>S3</td>
<td>P0, P6, P0, P3, P4, P9, P6, P3, P4, P9, P2, P5, P2, P1</td>
</tr>
<tr>
<td>S4</td>
<td>P0, P1, P8, P3, P4, P5, P10, P0, P5, P2</td>
</tr>
</tbody>
</table>

Step 1: Apply Markov Predictors on the data stored in Table 4.3 by calculating the individual pages that have been visited in these user sessions. It may be noted that there are P0 to P10 (i.e. 11 pages) individual pages that have participated the user sessions S2, S3 and S4. These pages then form the row and column for the application of 1st order Markov predictors as shown in table 4.4. The values in these columns depict the existence of links that individual page shares with the other in these sessions.
Step 2: Find the minimum threshold value according to the equation 4.1 that could guide which sequences of web pages gets qualified for the application of the next higher level of markov predictors as given below:

\[
\text{Threshold} = \text{Maximum No of Links shown by any page/2} \quad \text{(4.1)}
\]

\[
= 4/2
= 2
\]

The threshold computed above (i.e. 2) is now used to select the web page sequences that qualify for the application of 2nd order markov predictors. The selected web page sequences are stored in Table 4.5.

Table 4.5 Qualified Web Page Sequences for the 2nd order Markov predictors

<table>
<thead>
<tr>
<th></th>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Step 3: Set k = 2 and apply 2nd order markov predictors on the short listed data of web page sequences of Table 4.5 to obtain Table 4.6
Table 4.6: 2nd order Markov Predictors

<table>
<thead>
<tr>
<th></th>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0→P1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P0→P3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P3→P4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>P4→P5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P4→P9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P5→P2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Step 4: Repeat Step 2 on Table 4.6 to obtain Table 4.7

Table 4.7: Qualified web page sequences after 2nd order markov predictors

- P0→P3→P4
- P3→P4→P5
- P3→P4→P9

Step 5: Repeat Step 3 for k = 3 on Table 4.7 to obtain Table 4.8

Table 4.8: 3rd order Markov predictors

<table>
<thead>
<tr>
<th></th>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0→P3→P4</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3→P4→P5</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3→P4→P9</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

It may be noted that in Table 4.8, we are now left with ground level web page sequences which do not qualify for the application of next higher level of markov predictors as all the entries in the table are less than the minimum threshold value. Thus, the qualified web pages sequences that the Rule Generator phase has pruned and will now move to the knowledge base or rule repository of the PPPE/SPPE are given as:

R1: P0→P3→P4
R2: P3→P4→P5
R3: P3→P4→P9
4.5 PHASE III: RULE ACTIVATOR

As discussed in the previous subsections, PPE can be deployed at two levels:

- At Proxy level
- At Server level

The common components of the PPE for both the frameworks have been explained in the previous subsections. The rule formation process encompasses both the 3rd and the 4th phases starting from the time when the user’s browsing patterns were recorded in the web log to the clustering of the user sessions, to application of dynamically chosen k-order markov predictors on these user sessions till the rules are actually formed. The rules thus formed i.e. R1, R2 and R3 need to be fired for prefetching the predicted pages. However, the firing process at proxy level differs slightly from the firing process at the server level. Therefore, both the cases have been discussed separately in the following subsections.

4.5.1 RULE ACTIVATION FOR PPPE

The architecture for Proxy Side Predictive Prefetching Engine (PPPE) given in Fig 4.1 has been modified to include a number of agents between the client machine and the proxy server as shown in Fig 4.6. It may be noted that for every client machine there is a designated agent that caters to the requests/responses to that client machine. The prefetching scheme undertaken by the agents who sit between the client machine and the proxy server works as follows:

1. Let the request be for document A.
2. The agent scans the rule database for the rules of the form A→X for some document X.
3. The agent then scans the database in forward chaining mode considering every rule or part of the rule which has X in its sequence (e.g. A→Y→X→Z). The intermediate rules are stored in the temporary area called Hint List. The rule scanning stops when there are no more rules to be fired.
4. Thus, the agent brings all the documents that succeed X to the hint list maintained by the agent itself and accordingly prefetch the corresponding web page from the proxy server to the client’s cache.

Fig.4.6 Proposed Architecture for Prefetching the Documents from the Rule Database of PPPE to the Client’s Cache.

5. The agent continues the scan and populates the hint list till such time the user requests for a web page which doesn’t appear in the sequence.

6. In case of new request, the agent cleans up the hint list and repeats step 2 to 5. However, when two or more rules with same head but different tails are encountered then the agent applies subsequence association rules to decide as to which rule to fire for prediction, the details of which is given in section 4.5.3.

4.5.2 RULE ACTIVATION FOR SPPE

As for rule activation for PPPE, a set of agents were included between the client machines and the proxy server as shoed in Fig 4.7. Whereas in case of SPPE, there is already a Group-Client-Agent (GCA) active in architecture as shown in Fig 4.2. Thus the
responsibility of rule activation can be very easily assigned to these GCAs. The overall process of how the GCA works in SPPE is shown in the flowchart of Fig 4.7.

Once the IP Matcher identifies the GCA according to the client machine, GCA gets activated and starts working on the prefetching scheme. The step by step workflow is explained as below:

1. Let the request be for document A.
2. The agent scans the rule database for the rules of the form $A \rightarrow X$ for some document X.
3. The agent then scans the database in forward chaining mode considering every rule or part of the rule which has X in its sequence (e.g. $A \rightarrow Y \rightarrow X \rightarrow Z$). The intermediate rules are stored in the temporary area called Hint List. The rule scanning stops when there are no more rules to be fired.
4. Thus the agent brings the URLs of all the documents that succeed X from the Database of URLs to its hint list and accordingly prefetch them to the client’s cache.
5. The agent continues the scan and populates the hint list till such time the user requests for a web page which doesn’t appear in the sequence.
6. In case of new requests, the agent cleans up the hint list and repeats steps 2 to 5. If the GCA finds two rules with same head and different tails, then it applies the subsequence association rules to find their confidence. The confidence is calculated based on their past history. The rule whose past history generates the maximum confidence is considered by GCA for prefetching as explained in the section 4.5.3. This helps in saving the network bandwidth which is generally considered an issue in the design of the prefetching mechanism.
7. If in case the document A doesn’t match as the head of the rule in the Rule-Repository, the request is forwarded by the GCA to the crawler. The crawler then crawl the web pages from the WWW and after indexing, add them to the Database of URLs.
8. Once the GCA has populated its hint-list with the web pages, it sends the signal to the page loader. The page loader then prefetches the client’s cache with the respective GCA’s hint list.

4.5.3 MATCHING PROCESS FOR RULE ACTIVATION

When two or more rules comes with same head and different tails, the agent/GCA (as the case may be) fires the rule from the rule repository by following procedure/steps.
Consider the rules stored in the Table 4.9, where R1 and R2 have same head and different tails. Now according to the prefetching mechanism, agent/GCA prefetches pages P3 and P4 but it has to determine as to which tail to be prefetched i.e. P5 or P9. After critically viewing the simulated results of section 4.4.1, table 4.7 shows two rules whose head is same but not the tail. As if the user visits page P3 then P4 will be prefetched but after that which page is more appropriate to prefetch P5 or P9 as shown below:

![Diagram](image)

It would be inappropriate to prefetch both the pages as this will be wastage of
- Network bandwidth and
- Cache storage.

So to decide, which of them is the most appropriate, we need to have a look on the past history of these web pages. Instead of looking into the enormously large web log, the lower approximation set formed by RST is usefully employed to search the past behavior of the user. Consequently, subsequence association rules would be used, the pseudocode of which is given in fig 4.8.

```
Algorithm association (A[m], R [ ])

Input A[m] (training data set formed from the past history of the rule)
Output R[ ] (Rule for prediction)
{
    calculate supp(A[m]);
    calculate conf (A[m]);
    if (confidence=100 %)
        Generate rule R [ ];
}
```

Fig 4.8 Algorithm for Matching the Right Rule for Prediction
The step by step procedure for the application of subsequence association rule is given below:

**Step 1:** Generate Table 4.9 using the history of such conflicting rule from the Lower Approximation Set shown in Table 4.3. The data contained in Table 4.9 becomes the training data for step 2.

Table 4.9: History table of the conflicting rule

<table>
<thead>
<tr>
<th>Past history</th>
<th>Head of the rule</th>
<th>Possible tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0, P2, P8, P0, P1,</td>
<td>P3, P4</td>
<td>P5</td>
</tr>
<tr>
<td>P2, P7, P0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P0, P0, P6, P0</td>
<td>P3, P4</td>
<td>P9</td>
</tr>
<tr>
<td>P6</td>
<td>P3, P4</td>
<td>P9</td>
</tr>
<tr>
<td>P0, P1, P8</td>
<td>P3, P4</td>
<td>P5</td>
</tr>
</tbody>
</table>

2: Apply subsequence association rules on this training data using the formula given in equation 4.2.

\[
\text{Conf}(X \Rightarrow Y) = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X)} \quad \ldots \quad (4.2)
\]

Where \( \text{Conf}(X \Rightarrow Y) \): is the confidence of the rule that is defined in terms of support for the rule.

\( \text{Supp}(X) \): is the support of an itemset \( X \) and is defined as the proportion of transactions in the data set which contain the itemset.

The training set of table 4.9 is then used to calculate the confidence for if the rule \( P3 \rightarrow P4 \rightarrow P5 \) or \( P3 \rightarrow P4 \rightarrow P9 \) is to be fired as shown in Table 4.10 and Table 4.11

Table 4.10: Confidence of accessing page P5 using subsequence association rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>( \frac{\text{Supp}(X \cup P5)}{\text{Supp}(P0)} )</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0 \rightarrow P5</td>
<td>( \frac{\text{Supp}(P0 \cup P5)}{\text{Supp}(P0)} )</td>
<td>3/6=50%</td>
</tr>
<tr>
<td>P2 \rightarrow P5</td>
<td>( \frac{\text{Supp}(P2 \cup P5)}{\text{Supp}(P2)} )</td>
<td>2/2=100%</td>
</tr>
<tr>
<td>P8 \rightarrow P5</td>
<td>( \frac{\text{Supp}(P8 \cup P5)}{\text{Supp}(P8)} )</td>
<td>1/2=50%</td>
</tr>
<tr>
<td>P1 \rightarrow P5</td>
<td>( \frac{\text{Supp}(P1 \cup P5)}{\text{Supp}(P1)} )</td>
<td>1/2=50%</td>
</tr>
<tr>
<td>P7 \rightarrow P5</td>
<td>( \frac{\text{Supp}(P7 \cup P5)}{\text{Supp}(P7)} )</td>
<td>1/1=100%</td>
</tr>
</tbody>
</table>
Table 4.11: Confidence of accessing page P9 using subsequence association rules

<table>
<thead>
<tr>
<th>P0→ P9</th>
<th>( \frac{\text{Supp}(P0 \cup P9)}{\text{Supp}(P0)} )</th>
<th>( \frac{2}{6} = 33.3% )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P6→ P9</td>
<td>( \frac{\text{Supp}(P6 \cup P9)}{\text{Supp}(P0)} )</td>
<td>( \frac{2}{2} = 100% )</td>
</tr>
</tbody>
</table>

Table 4.10 shows that P7→P5 and P2→P5 have the highest confidence i.e. 100%. While Table 4.11 shows that P6→P9 has the highest confidence i.e. 100%. Using Markov models, it was determined that the next page to be accessed by the user after accessing the pages P3 and P4 could be either P5 or P9. Whereas subsequence association rules take this result a step further by determining that if the user accesses page P2 or P7 before pages P3 and P4, then there is a 100% confidence that the user will access page P5 next. Whereas, if the user visits page P6 before visiting pages P3 and P4, then there is a 100% confidence that the user will access page P9 next.

The proposed framework thus introduces a mechanism that efficiently provides the prediction of the web pages that the user is likely to access in near future based on the past history. In the past, various data mining techniques were applied individually which were either not accurate enough to predict the web pages to be accessed next by the user or consumed a lot of network bandwidth.

This work is the careful amalgamation of the various data mining techniques including Rough Set Clustering, dynamically chosen k-order Markov Predictors and the Association Rules.

PPE was implemented and test run was carried on the core components and encouraging results were obtained in terms of rule formation from the web log. The results obtained thereof are discussed in the next chapter.