CHAPTER 3

PREPROCESSING USING CONNOISSEUR
ALGORITHMS

3.1 Introduction

In this thesis work, a model is developed in a structured way to mine the frequent patterns in e-commerce domain. Designing and implementing offline modules are the general practice since data has to be cleaned and preprocessed. However, in case of real time systems e-business environments have to act on latest data that is available to them. This is essential for them not only to fierce global competition, but also to market products according to the wish of customers. So an algorithm supporting the real time environment is needed. In this research work, such an algorithm is developed and used. The architecture of this model supports both online and offline modules. Figure 3.1 is the structure of the developed model that finds frequent patterns in e-commerce domain, which can also support real time environments.

3.2 Structure of the model

The developed model constitutes three major modules. First, one is the Data collection and preprocessing module. This module suggests the methods of collecting e-commerce transaction data and describes how noise in the web log data is cleaned and transformed into a form that is suitable for mining. First phase of the association rule mining, the frequent itemset discovery is implemented in the second module. Cleaned and preprocessed data are mined in this module to find frequent
itemsets. In the third module, frequent patterns are mined using the frequent itemsets discovered in the second module. Figure 3.2 depicts the general control flow of the model.

Figure 3.1 Structure of the model
3.2.1 Data collection and preprocessing

E-commerce Data can be collected from various sources like web server logs, packet sniffers, application server side etc. Proposed model rely on web server logs. The web usage data are generated by interaction between person browsing the site and servers of the e-commerce platform. These are stored as log files. The information available in web log files can say what prospective customers seek from a site. Figure 3.3 shows the block diagram of data collection module. The data collected from these weblog files are rich containing information on customer activity. The actions like the page visited, what the customers look at, what they put into their shopping cart and so on.

In this work, data are collected from the web servers of cooper power systems. The weblogs for a period of one week is chosen as the transactional data for mining purpose.
Data collected in the log files of the server cannot be used for mining purpose in the form, as it is stored. So preprocessing is an essential activity before mining the data. Major aspects of data preprocessing are removing records, which are irrelevant for mining, user identification, and session identification. Algorithms are developed and implemented for the cleaning and preprocessing activities. Final output is the database table containing Date, time, client IP address, page visited and time stamp. This data mapped with shopping cart details can provide valuable information. This database is again altered to a form suitable for the data mining module.

3.2.2 Frequent itemsets discovery

Second module of the model finds the frequent itemsets. Association rule mining used in the model is a very popular data mining technique and it finds relationships among the different entities of records. Since the introduction of frequent itemsets in 1993 by Agrawal et al. [74], it has received a great deal of attention in the field of pattern discovery and data mining. Strategy of making association rule mining an
offline task and referring to the discovered pattern can be ineffective because customer preference may change over time. Therefore, an algorithm, which analyzes the weblogs that can also support real time, has to be designed. The new algorithm designed for finding frequent itemsets is Limited Level Tree algorithm - LLT. LLT algorithm is memory efficient, compact, faster compared to other traditional algorithms and supports streaming real time data.

LLT algorithm works in two phases. In the first step, the given database is scanned and candidate itemsets are generated for each transaction. These candidate itemsets are stored in a special compact three level tree structure called LLTree. Once the tree is constructed, there is no need for scanning the original database again. In the second step, this tree structure is explored to find frequent itemsets for different user specified minimum threshold levels. The process flow is depicted in the figure 3.4.
3.2.3 Rule generation

Rule generation is the second step of Association rule mining and is a straightforward task. It is implemented in the third module of the model. An Association Rule is of the form $X \rightarrow Y$. Amongst itemsets that occur together in a database where $X$ and $Y$ are disjoint itemsets. Support
and confidence measures serve as the basis for routine techniques in Association rule mining. The support and confidence are predefined by users to drop the rules that are not so interesting or useful. The association rule indicates that the transactions containing X tend to also contain Y. Association Rules are generated from the discovered frequent itemsets. This module accepts user specified minimum support and confidence for selecting the useful rules, which is of interest.

### 3.3 Data Collection

The data from the e-commerce platform has to be collected for mining frequent patterns. Possible data sources and data source suited for this research work is discussed in this section. The format of the data collected and its descriptions are discussed in detail.

#### 3.3.1 E-commerce data classifications

Ecommerce data are classified mainly as web usage data, web content data, web structure data and business data.

**Web usage data** is generated by the interactions between the persons browsing a site and the servers on the e-commerce platform. This data can further be divided into log files and business event records. The useful information delivered by web log mainly includes IP address of the remote host making the request, Date/Time when the request occurs, URI of the object requested, and referrer field. The referrer field contains important information for marketing purposes, since it can track how people found the particular site. The information stored in a cookie log helps to improve the connectionless state of web server interactions,
enabling servers to track client access across their hosted web pages. A problem is that web logs only contain the name of the page requested; details of the content displayed on the web page may not even be captured by it. In addition, some events cannot also be determined from web logs. e.g., abandoning shopping carts, searches fail to find any results.

**Web content data** include HTML/XML pages, web page templates, email templates for campaigns, images, etc.

**Web structure data** provides the topology of a site, which represent the designer’s view of the content organization within the site.

**Business data** includes merchandising information e.g., products, assortments, and price list, business rules e.g., promotion rules, and rules for cross-sells and up-sells and sale transaction data.

Frequent pattern discovery needs only the web usage data. The data used in Web usage mining comes mainly from the Web server logs. Other sources of information are either very specific or very costly [30]. Potential Data sources are many [84]. Some among them are listed below. Figure 3.5 shows the pictorial representation of the possible data sources for web usage mining.

- Web server logs
- Proxy Server Logs
- client logs
- Packet sniffers
- Application server logs
3.3.1.1 Web server logs

Web servers are the richest and the most common source of data. Web server logs maintain a history of page requests. When a web user interacts with a site data recording their behavior is stored in web server logs. These logs may contain invaluable information about the users behavior in the site. Large amount of data can be collected from these log files. Data generation is discussed in detail in [15][30][46]. Entries of web server log files follow standard format such as Common Log Format (CLF) [57], Extended Common Log Format (ECLF) [24]. These logs usually contain basic information e.g.: name and IP of the remote host, date and time of the request, the request line exactly as it came from the client, etc. However, web logs only contain the name of the page requested; details of the content displayed on the web page may not even
be captured by it. Moreover, web server logs typically do not collect user-specific information. These files are usually not accessible to general Internet users, only to the webmaster or other administrative person.

### 3.3.1.2 Proxy Server Logs

A Web proxy is a caching mechanism, which lies between client browsers and Web servers. Many Internet Service Providers (ISPs) give their customer proxy server services to improve navigation speed through caching. Proxy server logs contain the HTTP requests from multiple clients to multiple Web servers. Collecting navigation data at the proxy level is the same as collecting data at the server level. The main difference in this case is that proxy servers collect data of groups of users accessing huge groups of web servers. This may serve as a data source to discover the usage pattern of a group of anonymous users, sharing a common proxy server. The drawbacks are

- Proxy-server construction is a difficult task. Advanced network programming, such as TCP/IP, is required for this construction.

- The request interception is limited, rather than covering most requests.

- The proxy logger implementation in Web Quilt, a Web logging system, performance declines if it is employed because each page request needs to be processed by the proxy simulator.
3.3.1.3 Client logs

Usage data can be tracked on the client side by using JavaScript, Java applets [82], or even modified browsers [16]. Various browsers like Mozilla, Internet Explorer etc. can be modified or various JavaScript and Java applets can be used to collect client side data. They provide detailed information about actual user behaviors [31][79][83]. However, this implementation of client-side data collection requires user cooperation, either in enabling the functionality of the JavaScript and Java applets, or to voluntarily use the modified browser. The drawbacks of this approach are:

- The design team must deploy the special software and have the end-users install it.
- The technique makes it hard to achieve compatibility with a range of operating systems and Web browsers.

3.3.1.4 Packet sniffers

Apart from Web logs, users behavior can also be tracked down on the server side by means of TCP/IP packet sniffers, which provide many advantages [68].

(i) data are collected in real time; (ii) information coming from different Web servers can be easily merged together into a unique log; (iii) the use of special buttons can be detected so that information usually unavailable in log files can be collected. Apart from these advantages, packet sniffers are rarely used in practice. Some of the drawbacks of Packet sniffers are they suffer with scalability issues on Web servers with high traffic [68].
[92], moreover they cannot access encrypted packets used in secure commercial transactions through the Secure Socket Layer. These limitation turns out to be severe when applying Web Usage Mining to e-businesses [8].

### 3.3.1.5 Application server logs

The best approach of data collection for mining purpose is directly accessing the server application layer, as done in [7]. However, this is not always possible. First, there is an issue related to the copyright of server applications. Above all, following this approach, Web Usage Mining applications must be modified for the specific servers. In addition, they must be taken into account the specific tracking requirements.

### 3.3.2 Data source for this research work

Web log data is preferred for this research work. Data collected from Cooper Power systems server for one week period is used for implementation. The data format of server log differs according to the option when applying and installing web service. Generally, server log is stored in three formats.

(i). W3C Extended log format

(ii). NCSA Common log format

(iii). Microsoft IIS log format

All the three are text formats. The raw log files consist of 19 attributes such as Date, Time, Client IP, Auth User, ServerName, ServerIP, ServerPort, Request Method, URI-Stem, URI-Query, Protocol
Status, Time Taken, Bytes Sent, Bytes Received, Protocol Version, Host, User Agent, Cookies, and Referer.

A sample of a single entry log file from the collected data is displayed in Figure 3.6.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>IP Address</th>
<th>Protocol</th>
<th>Status</th>
<th>URI</th>
<th>Bytes Sent</th>
<th>Bytes Received</th>
<th>User Agent</th>
<th>Cookies</th>
<th>Referer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-06-15</td>
<td>14:25:40</td>
<td>74.254.143.252</td>
<td>80 GET</td>
<td>200</td>
<td>/custsupport/Xref.cfm</td>
<td>0</td>
<td>12049</td>
<td>www1.cooperwiringdevices.com Mozilla/4.0+(compatible;+MSIE+8.0;+Windows+NT+5.1;+Trident/4.0;+InfoPath.1;+.NET+CLR+2.0.50727;+.NET+CLR+3.0.4506.2152;+.NET+CLR+3.5.30729;+.NET4.0C;+.NET4.0E)__utmc=1;__utma=1.782737072.1308145429.1308145429.1308145429.1308145429.1;__utmz=1.1308145429.1.1.utmcsr=(direct)utmccn=(direct)utmcmd=(none)<a href="http://search.yahoo.com/search;_ylt=A0oG7iG9wPhNsEEAnVpXNyOA?p=%2BCooper+Wiring+Image&amp;ei=UTF-8&amp;norw=1&amp;fr=ie8&amp;fr2=sp-qrw-orig-top&amp;xargs=0&amp;pstart=1&amp;b=21&amp;xa=0m8YXT43eD7RXM.RmDeUuA--,1308234301">http://search.yahoo.com/search;_ylt=A0oG7iG9wPhNsEEAnVpXNyOA?p=%2BCooper+Wiring+Image&amp;ei=UTF-8&amp;norw=1&amp;fr=ie8&amp;fr2=sp-qrw-orig-top&amp;xargs=0&amp;pstart=1&amp;b=21&amp;xa=0m8YXT43eD7RXM.RmDeUuA--,1308234301</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.6 Single entry of raw log file

a) Date
The date from Greenwich Mean Time (GMT x 100) is recorded for each hit. The date format is YYYY-MM-DD. The example from figure 3.6 shows that the transaction was recorded at 2011-06-15.

b) Time
Time of transactions. The time format is HH:MM:SS. The example from figure.3.6 shows that the transaction time was recorded at 14:25:40.
c) **Client IP Address**

Client IP is the number of computer who access or request the site. 74.254.143.252 is the Client IP in the figure. 3.6.

**d) User Authentication**

Some web sites are set up with a security feature that requires a user to enter username and password. Once a user logs on to a Website, that user’s “username” is logged in the fourth field of the log file.

**e) Server Name**

Name of the server. In figure.3.6 the name of the server is CIPT0053.

**f) Server IP Address**

Server IP is a static IP provided by Internet Service Provider. This IP will be a reference for accessing the information from the server.

**g) Server Port**

Server Port is a port used for data transmission. Usually, the port used is port 80.

**h) Server Method (HTTP Request)**

The word request refers to an image, movie, sound, pdf, txt, HTML file and more. The above example in figure 3.6 indicate that Xref.cfm was the item accessed. It is also important to note that the full path name from the document root. The GET in front of the path name specifies the way in which the server sends the requested information. Currently, there are three formats that Web servers send information [64] in GET, POST, and Head. Most HTML files are served via GET Method while most CGI functionality is served via POST.

**i) URI-Stem**

URI-Stem is path from the host. It represents the structure of the websites. For example:-

/custsupport/Xref.cfm
**j) Server URI-Query**

URI-Query usually appears after sign “?” . This represents the type of user request and the value usually appears in the Address Bar. For example:

```
?q=tawaran+biasiswa&hl=en&lr=&ie=UTF-8&oe=UTF-8&start=20&sa=N
```

**k) Status**

This is the status code returned by the server; by definition, this will be the three digit number [70]. There are four classes of codes:

i. Success (200 Series)
ii. Redirect (300 Series)
iii. Failure (400 Series)
iv. Server Error (500 Series)

A status code of 200 means the transaction was successful. Common 300-series codes are 302, for redirect and 304 for a conditional GET. This occurs when server checks if the version of the file or graphics already in cache is still the current version and directs the browser to use the cached version. The most common failure codes are 401 (failed authentication), 403 (Forbidden request to a restrict subdirectory, and the dreaded 404 (file not found) messages. In the above transmission, a status is 200 means that there was a successful transmission.

**a) Bytes Sent**

The amount of data revisited by the server, not together the header line.

**b) Bytes Received**

Amount of data sent by client to the server.

**c) Time Stamp**

This attribute is used to determine how long a visitor spent on a given page.
**d) Protocol Version**
HTTP protocol being used (e.g. HTTP/1.1).

**e) Host**
This is either the IP address or the corresponding host name (www.cooper.com.my) of the remote user requesting the page.

**f) User Agent**
The user agent reported by the remote user’s browser. Typically, this is the string describing the type and version of browser software being used.

**g) Cookies**
Cookies can be used to track individual users thus make the sessionizer task easier. However, the use of cookies also raises the concern of privacy thus; it requires the cooperation of the users.

**h) Referer**
The referring page, if any, as reported by the remote user’s browser.

**C. Agent Log**
The Agent Log provides data on a user’s browser, browser version, and operating system. This is the significant information, as the type of browser and operating system determines what a user is able to access on a site (e.g. Java, forms).

**D. Error Log**
The average Web user will receive an "Error 404 File Not Found" message several times a day. When a user encounters this message, an entry is made in the Error Log.

**E. Referer Log**
The Referer Log indicates what other sites on the Web link to a particular server. Each link made to a site generates a Referer Log entry.
One hundred and twenty-four Web log files were collected from the Cooper power system IIS Web server during the period between August 15 and 21, 2011. The total size of the Weblog files was about 16,730 MB, and total number of HTTP requests was about 18,000,000,000. For application to the model, this raw log file received from the server cannot be used directly by the mining algorithm. Data preprocessing tasks such as data cleansing, user identification, session identification, and path completion must be applied to the Web log files. Figure 3.7 is a sample of raw web log data collected from the IIS web server of Cooper power system.
3.4 PREPROCESSING

Preprocessing is the preliminary data mining practice where data are transformed into a format that can be easily and effectively processed by the user to solve the purpose. The data mining algorithm cannot directly use the raw log files received from the web server. It needs to undergo data preprocessing. Preprocessed data is then stored in the database suitable for data mining.

For the data mining module of this work, not all information in the raw log file is needed. Therefore it must be preprocessed and converted to a form suitable for mining phase [29][42][58][64][72]. The required tasks of preprocessing are

i. Data cleaning
ii. User identification
iii. Session identification
iv. Transaction identification also called Path completion.

The flow graph of preprocessing is shown in the figure 3.8
3.4.1 Data cleaning

The task of data cleaning is to remove the irrelevant and redundant log entries for the mining process. There are three kinds of irrelevant or redundant data need to be cleaned.

i. Accessorial resources embedded in HTML file
ii. Robots’ and spiders requests
iii. Error requests.

**Accessorial resources** - HTTP protocol is connectionless. Therefore, a user’s request to view a particular page often results in several log entries since graphics and scripts are downloaded in addition to the HTML file. There is no need to include file requests that the user did not request.
explicitly. Elimination of these items, which are irrelevant, can be done by checking the suffix of the URL name. For instance, all log entries with filename suffixes such as gif, jpeg, GIF, JPEG, jpg, JPG, css and map can be removed. In addition, common scripts such as the files requested with the suffixes of “.cgi” can also be removed. While requests for graphical contents and files are easy to eliminate, Robots and Web spiders navigation patterns must be explicitly identified.

**Robots’ requests.** Web robots, also called spiders, are software tools that scan a Web site to extract its content. Spiders automatically follow all the hyperlinks from a Web page. Search engines such as Google periodically use spiders to capture all the pages from a Web site to update their search indexes. Removal of this is usually done by referring to the remote hostname, by referring to the user agent, or by checking the access to the robots.txt file.

**Error requests.** Error requests are useless for mining process. They can be removed by checking the status of request. Generally, the status less than 200 and greater than 299 must be removed. For example, if the status is 404, it is shown that the requested resource is not existence. This log entry in log can be removed then.

In addition, logs, which use a request method except “Get”, must be deleted. The enhanced algorithm D-cleaning cleans all the above discussed types of irrelevant records.

The Enhanced algorithm for data cleaning is given below.
3.4.2 User Identification

A user is defined as the principal using a client to interactively retrieve and provide resources. User identification is the process of attaching user to requested page. User identification is greatly complicated because of the use of buffer, local caches, corporate firewalls, and proxy servers. It makes web log not accurately record user’s browsing behavior. User identification from the cleaned log is a
complex task. In this work, the following procedures are used to identify the user.

i. Each IP address represents one user

ii. If the IP address is the same, but the agent log shows a change in browser software or operating system, then that IP address represents a different user

iii. Proxy caching causes sharing a single IP address, i.e. the one belonging to the proxy server. So it becomes impossible to use IP addresses as users identifiers. This problem can be partially solved by the use of cookies [27], by URL rewriting [62], or by requiring the user to log in when entering the Web site [6].

However, due to complexity, users with same IP address are treated as single users in this work. Algorithm UserIdent implements the above discussed procedures to identify the users. The enhanced algorithm for user identification is given below.
Algorithm 3.2 User Identification

Algorithm UserIdent.

Input: Cleaned Web Log database Webclean_database

Output: user set

1. Begin
2. IPSet = ∅
3. Userset = ∅
4. Browserset = ∅
5. OSSet = ∅
6. i = 0
7. While not eofWebclean_database Do
8. Read WebLogRecord from Webclean_database
9. If WebLogRecord.IP not in IPSet
   Then
   IPSet = IPSet U WebLogRecord.IP
   Browserset = Browserset U WebLogRecord.Browser
   OSSet = OSSet U WebLogRecord.OS
   i = i + 1
10. Userset = Userset U {Ui}
11. Else
12. If WebLogRecord.Browser not in Browserset OR
13. WebLogRecord.OS not in OSSet
   Then
   i = i + 1
   Userset = Userset U {Ui}
14. End If
15. End If
16. End While
17. End
3.4.3 User Session Identification

A user session means a delimited set of user clicks across one or more Web servers. After user identification, the pages access of each user must be divided into individual session, which is called session identification. The goal of session identification is to divide the page accesses of each user into individual sessions and find each user’s access pattern and frequency path. Because the HTTP protocol is stateless, it is virtually impossible to determine when a user actually leaves the Web site. So determining when a session should be considered finished is little difficult. Berendt et al., [12] described and compared three heuristics for the identification of sessions termination. Two were based on the time between users page requests, one was based on information about the referrer. Cooley et al., [24] proposed a technique to infer the timeout threshold for the specific Web site. Other authors proposed different thresholds for time oriented heuristics based on empiric experiments. Session identification is also discussed in detail in [6] [62] [87].

The main technique used in this work to identify user session is timeout mechanism. The rules are:

i. If there is a new user, there will be a new session

ii. In one user session, if the refer page is null, there will be a new session

iii. If the time between page requests exceeds a certain limit, it is assumed that the user will start a new session.

The algorithm SessionIdent follows the above said rules to identify the sessions. The enhanced algorithm for session identification is given below.
Algorithm 3.3 Session Identification.

Algorithm SessionIdent

Input: Web log database;
Output: sessions set

WBL = Web log database;

1 Begin
2 Session set = \emptyset
3 UserSet = \emptyset
4 k = 0
5 While not eof ( WBL ) Do
   a. LogRecord = Read (WBL)
6 If ( LogRecord.Refer = ‘-’ OR
7   LogRecord.time-taken > 30min OR
8   LogRecord.UserID not in UserSet)
9   Then
   a. k = k + 1
10 Sk = LogRecord.Url
11 Sessionset = Sessionset U { Sk }
12 End If
13 End While
14 End

3.4.4 Transaction Identification

A transaction is defined as a set of homogeneous pages that have been visited in a user session. Each user session can be considered as only one transaction composed of all the visited pages or it can be divided into
a smaller set of visited pages. There are three strategies to identify transactions.

i. Transaction Identification by Reference Length
ii. Transaction identification by Maximum Forward Reference
iii. Transaction Identification by Time Window

Formally, a transaction is composed of an IP address, user Identification and a set of Visited Pages, which are identified, by its URL and access time.

\[ t = \langle ip_t, uid_t, \{(l_1^t.\text{url}, l_1^t.\text{time}), \ldots, (l_m^t.\text{url}, l_m^t.\text{time})\} \rangle \]  \hspace{1cm} (3.1)

\[ \text{for } 1 \leq k \leq m, l_k^t.\text{url} \in L, l_k^t.\text{ip} = ip, l_k^t.uid = uid_t, \]

(i). Transaction Identification by Reference Length:

This strategy, proposed by Cooley et. al. [24],[26] is based on the assumption that the time that a user spends in an auxiliary page is lower than a content page. Obtaining a time \( t \) by maximum likelihood estimation and defining a threshold \( C_e \), the pages are added to the transaction if they are considered auxiliary-content.

\[ \text{for } 1 \leq k \leq (m-1) : l_k^t.\text{time} \leq C_e \text{ and } k = m : l_m^t.\text{time} > C_e \]  \hspace{1cm} (3.2)

While for only content pages transactions:

\[ 1 \leq k \leq m : l_k^t.\text{time} > C_e \]  \hspace{1cm} (3.3)

(ii) Transaction identification by Maximum Forward Reference:

This strategy is based on the idea proposed by Chen et al., [19]. A transaction is considered as the set of pages from the first visited page until the previous page where the user does a back reference. A back
reference appears when the user accesses again to a previously visited page in the current session, while a forward reference is to access to a page not previously visited in the current session. Therefore, the maximum forward reference is the content pages and the path to the maximum reference is composed of index pages.

iii) Transaction Identification by Time Window:

This strategy divides a user session into time intervals lower than a fixed threshold. In this strategy, the last visited page normally does not correspond to a content page unlike the previous strategy. If $W$ is the size of the time window, the accesses that are included to the transaction defined in (3.1) are those that fulfill (3.4).

$$l^t_m.time - l^t_1.time \leq W$$

This strategy is normally combined with the previous ones.

In this work Maximum forward reference length method is used to find the transactions.

The above preprocessing tasks ultimately result in a set of $n$ page views

$$P = \{p_1, p_2, \cdots, p_n\},$$

and a set of $m$ user transactions,

$$T = \{t_1, t_2, \cdots, t_m\},$$

where each $t_i \in T$ is a subset of $P$.

Conceptually, each transaction $t$ is viewed as an $l$-length sequence of ordered pairs:
where each $pt_i = p_j$ for some $j \in \{1, \cdots, n\}$, and $w(pt_i)$ is the weight associated with page view $pt_i$ in the transaction $t$.

The weights can be determined in a number of ways. Weights can be binary, representing the existence or nonexistence of a product-purchase or a documents access in the transaction, or they can be a function of the duration of the associated page view in the user’s session.

In this work since the focus is on association rule mining, binary weights is considered on page views within user transactions. Here ordering among the page views is ignored. The reason for that is, this work concentrates on discovering only frequent patterns not the sequential patterns. Thus, a transaction can be viewed as a set of page views

$$st = \{pt_i \mid 1 \leq i \leq l \text{ and } w(pt_i) = 1\}. \quad (3.6)$$

The resultant table format is shown in the Figure3.9.
Figure 3.9 Resultant table format

Figure 3.7 provides a part of raw Weblog data and the corresponding preprocessed transaction database is shown in the Figure 3.10. The period between August 15 and 21, 2011 consists of 8960 transaction records created by the target customers.
Existing Preprocessing algorithms uses various techniques to clean and transform the data. Almost all the algorithms remove the images. However, this is not the case when it comes to user session identification and path completion. Not all preprocessing algorithms perform that. Similarly, session identification is performed using different techniques. A comparative study is made in the table 3.1.
### Table 3.1 comparison of preprocessing algorithms

<table>
<thead>
<tr>
<th>Author</th>
<th>Removing images</th>
<th>Removing Robot text</th>
<th>User Session identification</th>
<th>Path completion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Login</td>
<td>IP</td>
</tr>
<tr>
<td>Yan, Jacobson[91]</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>√</td>
</tr>
<tr>
<td>Pitkow[92]</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Shahabi[93]</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Chen et al[89]</td>
<td>√</td>
<td>X</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R. Cooley[82]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Venkateswari et al</td>
<td>√</td>
<td>√</td>
<td>X</td>
<td>√</td>
</tr>
</tbody>
</table>

√ - Used  X - Not used

Graphs in the figure 3.11 and 3.12 gives a better view of the preprocessing algorithm comparison.
Figure 3.11 Preprocessing Algorithm comparison -I
3.5 Discussion

General description of the model and a brief explanation of the modules are seen in this chapter. The first module, which deals with collection and preprocessing of weblog data, are explained in detail. Various methods of data collection, steps in preprocessing, algorithms developed and implemented for preprocessing are elaborated and a comparative study of preprocessing algorithms are also made.