International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 3, ISSN 2249 –071X

An Effective Web Usage Mining

L. Balaji
Sr. Asst. Professor
Computer Science Engineering,
Shri Vishnu Engineering College For Women,
Vishnupur, Bhimavaram,
West Godavari District, Andhra Pradesh, India

Dr. Y.S.S.R. Murthy
Computer Science Engineering,
Shri Vishnu Engineering College For Women,
Vishnupur, Bhimavaram,
West Godavari District, Andhra Pradesh, India

Abstract — Web usage mining is the area of data mining which deals with the discovery and analysis of usage patterns from web data, specifically web logs, in order to improve web based applications. Web usage mining consists of three find phases–, pre-processing, pattern discovery, and pattern analysis. After completion of these three phases the user can the required usage patterns and use these information for specific needs. Web log files are the primary data source for web usage mining. This usage analysis includes tasks like page access frequency, finding the common traversal paths through a website. These log files contains information that can't be directly interpreted, for example information like who is accessing, which pages are accessing by whom, how much time user is accessing a particular page ,can't be obtained directly these log files. Since log files are unformatted text files, they are complex to interpret and analyze. In this paper we propose a novel approach using universally accepted formatting language XML. In our approach text based log files are converted into XML format using parsers. Once a log file is in XML format, using DOM API or types of parser API's we can retrieve the required information in an easy manner such as user and session identification and the paths that are frequently accessed. This paper presents several data preparation techniques based on XML parsers in order to increase the usability of websites.

Keywords **— Data Mining, Server logs, XML, Web mining,**

I. INTRODUCTION

Web usage mining tries to make sense of the data generated by the web surfer's sessions or behaviours. While Web content and structure mining utilize the real or primary data on the web, Web usage mining mines the secondary data derived from the interactions of the users while interacting with the web. The web usage data includes the data from web server access logs, proxy server logs, browser logs, user profiles, registration data, user sessions or transactions, cookies, user queries, bookmark data, mouse clicks and scrolls and any other data such as the results of interactions [1]. Web usage mining focuses on data collection, pre-processing and the data mining techniques.

A. Web log data

In Web Mining, data can be collected at the server-side, client-side and proxy servers. Each type of data collection differs not only in terms of the location of the data source, but also the kinds of data available and its methods of implementation. Logs are mostly stored simply as text files, each line corresponding to one access (i.e. one request). The most widely used log file formats are, by the Common Log File format (CLF) and the Extended Log File format (Ex LF)

B. Server-Level-Collection

A Web server log is an important source for performing Web Usage Mining because it explicitly records the browsing behaviour of site visitors. The data recorded in server logs reflects the (possibly concurrent) access of a Web site by multiple users. The Web server can also store other kinds of usage information such as cookies and data in separate logs [5].

Limitations

• However, the site usage data recorded by server logs may not be entirely reliable due to the presence of various levels of caching within the Web environment. Cached page views are not recorded in a server log.
• In addition, any important information passed through the POST method will not be available in a server log. Packet sniffing technology is an alternative method to collecting usage data through server logs. Packet sniffers monitor network traffic coming to a Web server and extract usage data directly from TCP/IP packets.

C. Client-Level-Collection

Client-side data collection can be implemented by using a remote agent [3] (such as JavaScript or Java applets) or by modifying the source code of an existing browser (such as Mosaic or Mozilla) to enhance its data collection capabilities. The implementation of client-side data collection methods requires user cooperation, either in enabling the functionality of the Java scripts and Java applets, or to voluntarily use the modified browser.

Advantage

Client-side collection has an advantage over server-side collection because it ameliorates both the caching and session identification problems. However, Java applets perform no better than server logs in terms of determining the actual view time of a page.

Limitations

- It may incur some additional overhead especially when the Java applet is loaded for the first time.
- Java scripts, on the other hand, consume little interpretation time but cannot capture all user clicks (such as reload or back buttons). These methods will collect only single-user, single-site browsing behaviour.
- A modified browser is much more versatile and will allow data collection about a single user over multiple Websites. The most difficult part of using this method is convincing the users to use the browser for their daily browsing activities.

D. Proxy-Level-Collection

A Web proxy [4] acts as an intermediate level of caching between client browsers and Web servers. Proxy caching can be used to reduce the loading time of a Web page experienced by users as well as the network traffic load at the server and client sides. The performance of proxy caches depends on their ability to predict future page requests correctly. Proxy traces may reveal the actual HTTP requests from multiple clients to multiple Web servers. This may serve as a data source for characterizing the browsing behaviour of a group of anonymous users sharing a common proxy server.

E. Web log format

In this paper we take W3C extended log format shown below

date time c-ipcs-username s-sitename s-computername s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status time-taken cs-version cs-host cs(User-Agent) cs(Referer)

W3C Extended Logging Field Definitions

Prefix	**Meaning**
S	Server actions.
C	Client actions.
Cs	Client-to-server actions.
Sc	Server-to-client actions.

Table 1: Web log field prefixes and their meanings

Field	**Appeared as**	**Meaning**
Date	Date	The date on which the activity occurred.
Time	Time	The time at which the activity occurred.
Client IP Address	c-ip	The IP address of the client who accessed your server.
User Name	cs-username	The name of the authenticated user who accessed your server. This does not include anonymous users, who are represented by a hyphen (-).
Service Name	s-sitename	The Internet service and instance number that was accessed by a client.
Server Name	s-computername	The name of the server on which the log entry was generated.
Server IP Address	s-ip	The IP address of the server on which the log entry was generated.
Server Port	s-port	The port number (the client is connected to.
Method	cs-method	The action the client was trying to perform (for example, a GET method).
URI Stem	cs-uri-stem	The resource accessed; for example, Default.htm.
URI Query	cs-uri-query	The query, if any, the client was trying to perform.
Protocol Status	sc-status	The status of the action, in HTTP or FTP terms.
Bytes Sent	sc-bytes	The number of bytes sent by the server.
Bytes Received	cs-bytes	The number of bytes received by the server.
Time Taken	time-taken	The duration of time in milliseconds, that the action consumed.
Protocol Version	cs-version	The protocol (HTTP, FTP) version used by the client. For HTTP this will be either HTTP 1.0 or HTTP 1.1.
Host	cs-host	Displays the content of the host header.
User Agent	cs(User-Agent)	The browser used on the client.
Cookie	cs(Cookie)	The content of the cookie sent or received, if any.
Referrer	cs(Referer)	The previous site visited by the user. This site provided a link to the current site.

Table 2. Web log fields and their meanings

Log files were designed to produce site-level performance statistics. It's thus no surprise that they can't provide even the minimum information needed to effectively investigate a potential usability problem. Here are some specific ways log files provide insufficient or misleading data:

- *Who is visiting your site*. For you to know who is visiting your site, the log file must contain a person ID such as a login to the server or to the user's own computer. However, most web sites do not require users to log in, and most web servers do not make a "back door" request to learn the user's login identity on his/her own computer.
- *The path visitors take through your pages*. The path that visitors follow within your site is clear if the log file contains an entry for every page viewed. However, when browsers are set to view pages from cache (usually the default), or when corporate or ISP servers retrieve pages from a central cache, then some pages will not be logged by the web server and the log file will have gaps. For example, with caching, pages viewed using the Back button typically are not logged. In addition, nothing appears in the log file when visitors arrived at a page by typing its URL, using a bookmark, or following an email link. In these cases one can try to infer from Referrer data.
- *How much time visitors spend on each page*. The log file records the time when a data transmission was initiated, but not the time when the transfer was completed. In addition, it is unclear when during the download process the user began viewing a page.

However, by comparing the timestamps of the current request and the next request, you can calculate roughly how much time a visitor is spending on a page— unless the visitor walks away while the computer is displaying the page. Some timing details may also be obtained by analyzing the transmission of graphics files associated with a page.

- *Where visitors are leaving your site*. The log file records the last page transferred by the server for that user session, but there are two reasons why it might not be the last page viewed. First, the last page viewed may have been displayed from cache. Second, the user may have left his/her workstation for a period of time that exceeds what the log analysis software regards as a session.

F. Using XML format of web log file

The log files contain information that can't be directly interpreted. for example information like who is accessing, which pages are accessing by whom, how much time user is accessing a particular page can't be obtained directly from these log files. Since log files are unformatted text files complex to interpret and analyze. In our approach text based log files are converted into XML format using parsers. Once log file is in XML format, using DOM API or others types of parser API's we can retrieve the required information in an easy manner.

The general log file format is shown below:

date time c-ipcs-username s-sitename s-computername s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status time-taken cs-version cs-host cs(User-Agent) cs(Referer)

and it's equivalent XML formatted file is shown below.

```
<web-log>
<record>
<date></date>
<time></time>
<c-ip></c-ip>
<cs-username></cs-username>
<s-sitename></s-sitename>
<s-computername></s-computername>
<s-ip></s-ip>
<s-port></s-port>
<cs-method></cs-method>
<cs-uri-stem></cs-uri-stem>
<cs-uri-query></cs-uri-query>
<sc-status></sc-status>
<sc-win32-status></sc-win32-status>
<sc-bytes></sc-bytes>
<cs-bytes></cs-bytes>
<time-taken></time-taken>
<cs-version></cs-version>
<cs-host></cs-host>
<cs-User-Agent></cs-User-Agent>
<cs-Cookie>      </cs-Cookie>
<cs-Referer></cs-Referer>
</record>
</web-log>
```

II. OUR APPROACH

Our XML based web usage mining process is shown in below figure

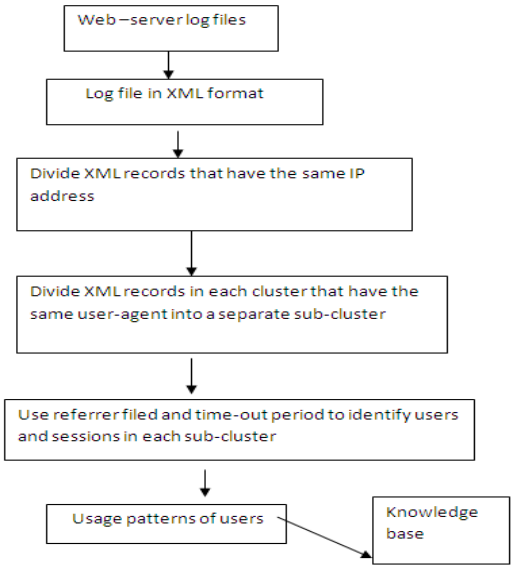

Fig.1. XML based web usage mining process

A. Data Cleaning

The first task to do is data cleaning, which should remove entries unhelpful to data analyzing and mining. Firstly, it should remove entries that have status of "error". Secondly, some access records generated by automatic search engine agent should be identified and removed from the access log. Primarily, it should identify log entries created by so called crawlers that are used widely in Web Information Retrieval Search engine tools. Such data. Offer retrieval mechanisms nothing to the analyzing of user navigation behaviours. Many crawlers voluntarily declare themselves in agent field of access log, so a simple string match during the data cleaning phase can strip off a significant amount of agent traffic. In addition, to exclude these accesses, employs several heuristic methods that are based on indicators of non-human behaviour. These indicators are (1) the repeated request for the same URL from the same host; (2) a time interval between requests too short to apprehend the contents of a page; and (3) a series of requests from one host all of whose referrer URLs are empty. The referrer URL of a request is empty if the URL was typed in, requested using a bookmark, or requested using a script. The last task of data cleaning, which is also disputable is whether it needs to remove log entries covering image, sound, and video files. Once log file is converted into XML format we start data cleaning phase in which we remove log entries involving image files and failed requests.

Next user identification phase starts. In this phase we group log records having the same IP address. The different group suggests different IP address. Same IP address doesn't means to be a single user, because when proxy servers are used, different users requests will come from same IP address. Therefore, in each cluster, we again

group log records having the same user-agent into a sub-cluster, i.e. this cluster contains log records of same IP address and same user-agent.

After this step, log records in each cluster having the same IP address are again sub-clustered, But we can't) simply judge, same IP address and same user-agent means a single user so we use the referrer field to find users and sessions using time-out periods. If a page is requested that is not directly reachable by a hyperlink from any of the pages visited by the user, this indicates there is another user with the same IP Address. Using time-out periods, if the time between page requests exceeds a certain limit, we can assume that the user is starting a new session.

Data Pre-processing can be done using XML (Extended Markup Language). XML provides a structure to the records which are present in web logs. Hence, understanding of web logs becomes easier. Logs recorded in the web log which is a text file are converted into DOM tree structure using XML parsers.

Consider the following sample log file.

192.168.5.234 - [10/nov/2009:03:0:01 -0500] "GET A.aspx HTTP/1.0" 200 3290 – IE5/Win2k
2. 192.168.5.234 - [10/nov//2009:03:0:09 -0500] "GET B.aspx HTTP/1.0" 200 2050 a.ASPX IE5/Win2k
3. 196.132.0.21- [10/nov/2009:03:0:10 -0500] "GET C.ASPX HTTP/1.0" 200 4130 - IE4/Win98
4. 196.132.0.21- [10/nov/2009:03:00:12 -0500] "GET B.ASPX HTTP/1.0" 200 5096 C.ASPX- IE4/Win98
5. 196.132.0.21- [10/nov/2009:03:0:15 -0500] "GET E.ASPX HTTP/1.0" 200 3290 C.ASPX IE4/Win98
6. 192.168.5.234 – [10/nov/2009:03:0:19 -0500] "GET C.ASPX HTTP/1.0" 200 2050 A.ASPX IE5/Win2K
7. 196.132.0.21- [10/nov/2009:03:00:22 -0500] "GET D.ASPX HTTP/1.0" 200 8140 B.ASPX IE4/Win98
8. 192.168.5.234 - [10/nov/2009:03:0:22 -0500] "GET A.ASPX HTTP/1.0" 200 1820 - IE4/Win98
9. 192.168.5.234 - [10/nov/2009:03:0:25 -0500] "GET E.ASPX HTTP/1.0" 200 2270 C.ASPX IE5/Win2k
192.168.5.234 - [10/nov/2009:03:00:25 -0500] "GET C.ASPX HTTP/1.0" 200 7220 A.ASPX IE4/Win98
192.168.5.234 - [10/nov/2009:03:10:33 -0500] "GET B.ASPX HTTP/1.0" 200 3290 C.ASPX IE4/Win98
192.168.5.234 - [10/nov/2009:03:0:58 -0500] "GET D.ASPX HTTP/1.0" 200 3290B.ASPX IE4/Win98
192.168.5.234 - [10/nov/2009:03:01:10 -0500] "GET E.ASPX HTTP/1.0" 200 3290 D.ASPX IE4/Win98
192.168.5.234 - [10/nov//2009:03:01:15 -0500] "GET A.ASPX HTTP/1.0" 200 3290 - IE5/Win2k
192.168.5.234 - [10/nov/2009: 03:01:16 -0500] "GET C.ASPX HTTP/1.0" 200 3290 A.ASPX IE5/Win2k
192.168.5.234 - [10/nov/2009: 03:01:17 -0500] "GET F.ASPX HTTP/1.0" 200 3290 C.ASPX IE4/Win98
192.168.5.234 - [10/nov/2009: 03:01:25 -0500] "GET F.ASPX HTTP/1.0" 200 3290 C.ASPX IE5/Win2k
192.168.5.234 - [10/nov/2009: 03:01:30 -0500] "GET B.ASPX HTTP/1.0" 200 3290 A.ASPX IE5/Win2k
192.168.5.234 - [10/nov/2009: 03:01:36 -0500] "GET D.ASPX HTTP/1.0" 200 3290 B.ASPX IE5/Win2k

Fig.2. A Sample Web Log File

The above log file is converted into XML based log file using XML parsers. Since all fields are not required for analysis we take data and time and client IP address and referrer and cs-uri-stem fields only. As shown below.

```
<record>
<date>25/Apr/2009 </date>
<time>03:04:41</time>
<c-ip> 192.168.5.234</c-ip>
<cs-Referer> -</cs-Referer>
<cs-uri-stem> A.aspx </cs-uri-stem>
</record>
<record>
<date>25/Apr/2009</date>
<time>03:05:34 </time>
<c-ip> 192.168.5.234</c-ip>
<cs-Referer>- </cs-Referer>
<cs-uri-stem>L.aspx </cs-uri-stem>
</record>
<record>
<date>25/Apr/2009</date>
<time>03:05:39 </time>
<c-ip> 192.168.5.234</c-ip>
<cs-Referer>A.ASPX </cs-Referer>
<cs-uri-stem>B.ASPX </cs-uri-stem>
</record>
<record>
<date>25/Apr/2009</date>
<time>03:06:02 </time>
<c-ip>192.168.5.234 </c-ip>
<cs-Referer> -</cs-Referer>
<cs-uri-stem> A.ASPX</cs-uri-stem>
</record>
……..
```

Figure 3: A ***partial XML file after completion of data cleaning phase***

B. User and session identification:

After data cleaning the XML file is analyzed for user and session identification. For this task we used DOM API and XSLT to transform XML tree into any form required for our analysis. The following sample code shows our approach of extracting only required fields, i.e. time, IP address, URL and referrer fields.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  - <xsl:template match="web-log/record">
    - <tr>
      - <td>
          <xsl:vlaue-of select="time" />
        </td>
      - <td>
          <xsl:vlaue-of select="c-ip" />
        </td>
      - <td>
          <xsl:vlaue-of select="cs-Referer" />
        </td>
      - <td>
          <xsl:vlaue-of select="cs-uri-stem" />
        </td>
      </tr>
    </xsl:template>
  </xsl:stylesheet>
```

Fig.4. An XSLT Transformation Applied to

Cleaned XML log file

Using DOM API, we can access any element in the tree by means of simple methods. User–identification task is greatly complicated by the existence of local caches, corporate firewalls, and proxy servers. For example, if the IP address of the ame, but the proxy information has changed, indicating that the user may be behind a firewall in a different users within the network, you can mark the IP address for different users; further one can also access information, refer to institutions with information and site topology to construct a user's browsing path, if the current page request to drop the user has viewed the page with no link between the existence of IP addresses that the same number of users. The Web Usage Mining methods that rely on user cooperation are the easiest ways to deal with this problem. However, even for the log/site based methods, there are heuristics that can be used to help identify unique users. Even if the IP address is the same, if the agent log shows a change in browser software or operating system, a reasonable assumption to make is that each different agent type for an IP address represents a different user. If a page is requested that is not directly reachable by a hyperlink from any of the pages visited by the user, again, the heuristic assumes that there is another user with the same IP address.

Identifying the user during a session in another issue is determining whether the access log is not an important record of the request. This requires the path to add to complete these records. If the current page the user requested the last page of the http request is no hypertext links, the user may use the browser "BACK" function call to cache the page in the machine. Check reference information to determine which page from the current request, if the user access to record the history of more than one page contains a link to the page with the current request, the request is the time closest to the source as the current request, if the reference information is not complete, you can take advantage of the topology of the site instead. For logs that span long periods of time, it is very likely that users will visit the Web site more than once. The goal of session identification is to divide the page accesses of each user into individual sessions. The simplest method of achieving this is through a timeout, where if the time between page requests exceeds a certain limit, it is assumed that the user is starting a new session. Many commercial products use 30 minutes as a default timeout, and established a timeout of 25.5 minutes based on empirical data. Once a site log has been analyzed and usage statistics obtained, a timeout that is appropriate for the specific Web site can be fed back into the session identification algorithm.

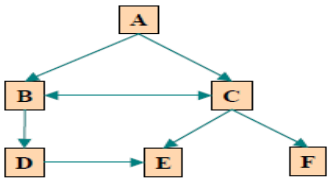

Fig.5. Site Topology

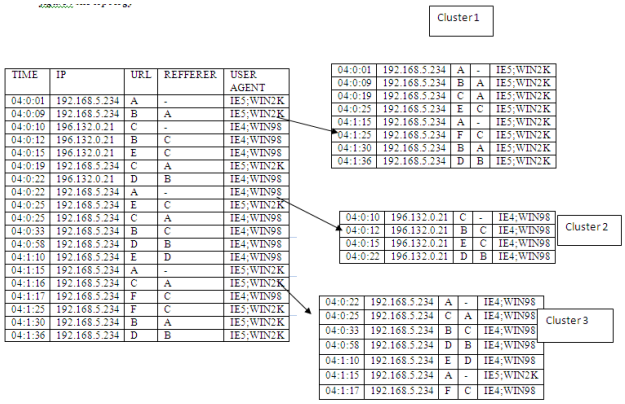

Fig.6. XML parser converted output file is clustered sub clusters based on IP address and user agent

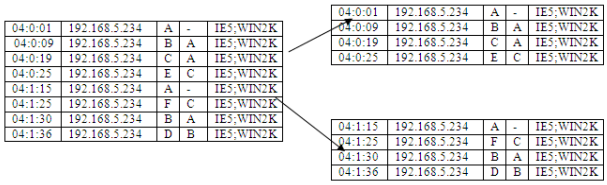

Fig.7. sessions produced using time heuristic of 30 minutes on cluster 1

III. RESULTS

To validate the effectiveness and efficiency of our methodology mentioned above, we have made an experiment with the web server log. The data source size for our experiment is 11MB. Our experimental results are shown below. After data cleaning, the number of requests declined from 7890 to 3297. Finally, on the basis of user identification's results, we have identified 1945 sessions by a threshold of 30 minutes and path completion.

Entries in raw web	log Entries after data cleaning	Number of users	Number of sessions
7890	3297	1352	1945

The results are shown in a the below bar chart

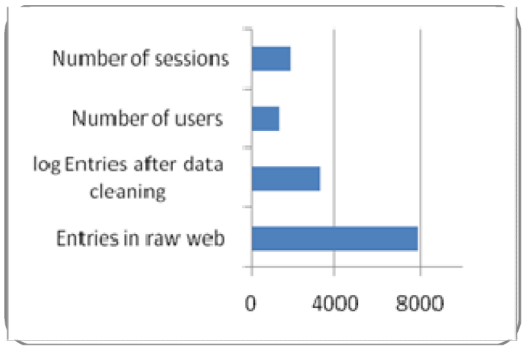

Fig.8. Results of our approach identifying users and sessions

IV. CONCLUSION

This paper has presented the details of data pre-processing tasks that are necessary for performing Web Usage Mining, the application of data mining and knowledge discovery techniques to web server access logs. We give some rules based on heuristics in every phase of data pre-processing in order to design and implement them easily. Our experiments have let us estimate data pre-processing importance and our methodology's effectiveness.It not only reduces the log file size but also increases the quality of the available data. As we used XML DOM technology, in the future XML technology based web services demand more, so using XML in our approach could increase effectiveness of the web usage mining process.

REFERENCES

[1] Raymond Kosala, Hendrik Blockeel, "Web Mining Research: A Survey", Katholieke Leuven, Belgium
[2] Miha Grcar, "User Profiling: Web usage Mining", Jozef Stefan Institute, Solvenia
[3] Yongjian Fu, Ming-Yi Shih, "A Framework for Personal Web Usage Mining", University of Missouri, Rolla.
[4] Jan Kerkhofts, Dr. Koen Vanhoof, "Web Usage Mining on Proxy Servers", Limburg University Centre, July 30,2001.
[5] Jaideep Srivastava, Robert cooley, "University of Minnesota", Minneapolis
[6] Berendt B., Mobasher B., Nakagawa M., Spiliopoulou M. The Impact of Site Structure and User Environment on Session Reconstruction in Web Usage Analysis. Proc. WEBKDD 2002: Mining Web Data for Discovery Usage Patterns and Profiles, LNCS 2703, Springer-Verlag, 2002:159-179
[7] Pirolli P., Pitkow J., Rao R. Silk from a sow's ear: Extracting usable structures from the Web. In: Proc. 1996 Conference on Human Factors in Computing Systems (CHI-96), Vancouver, British Columbia, Canada, 1996.
[8] Tanasa D., Trousse B. Advanced data preprocessing for intersites Web usage mining. Intelligent Systems, IEEE,2004(19): 59 – 65
[9] Catledge L., Pitkow J. Characterizing browsing behaviors on the World Wide Web, Computer Networks and ISDN Systems, 1995,27(6):1065-1073.
[10] Chen M.S., Park J.S., Yu P.S. Data mining for path traversal patterns in a web environment. In Proceedings of the 16th International Conference on Distributed Computing Systems, 1996:385-392.

AUTHOR'S PROFILE

L. Balaji
Sr. Asst. Professor
Computer Science Engineering,
Shri Vishnu Engineering College For Women,
Vishnupur, Bhimavaram,
West Godavari District,
Andhra Pradesh, India

Dr. Y.S.S.R. Murthy
Computer Science Engineering,
Shri Vishnu Engineering College For Women,
Vishnupur, Bhimavaram,
West Godavari District,
Andhra Pradesh, India