WEBPAGE CLASSIFICATION USING SVM LIGHT

A PROJECT REPORT

Submitted by

BHAVANA.S **(71205205011)**

DHARANIKHA.V **(71205205014)**

PREETHY.S **(71205205037)**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2009

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**WEBPAGE CLASSIFICATION USING SV LIGHT**" is the bonafide work of **BHAVANA S, DHARANIKHA V AN PREETHY S** who carried out the project work under my supervision.

24/4/09

SIGNATURE

Ms.S.Sathyavathi M.E

SUPERVISOR

Department of Information Technology
Kumaraguru College Of Technology
Coimbatore-641006

SIGNATURE

Dr.S.Thangasamy

DEAN

Department of
Computer Science and Engineering
Kumaraguru College Of Technology
Coimbatore-641006

The candidates with University Register Nos. **71205205011, 71205205014 & 71205205037** examined by us in the project viva-voce examination held on

27th APRIL, 2009

27/4/09

INTERNAL EXAMINER

27/4/09

EXTERNAL EXAMINER

DECLARATION

We,

Bhavana S	Reg.No:	71205205011
Dharanikha V	Reg.No:	71205205014
Preethy S	Reg.No:	71205205037

hereby declare that the project entitled **"Web Page Classification Using SVM Light"**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru College of Technology,Coimbatore.

Place: Coimbatore

Date: April 27,2009

Bhavana
[Bhavana S]

Dharanikha
[Dharanikha V]

S.Prf
[Preethy S]

Project Guided by

[**Ms.L.S Jayashree M.E., Ph.D**]

[**Ms.S.Sathyavathi M.E**]

ACKNOWLEDGEMENT

We express our sincere thanks to our Chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc., F.I.E.,** Vice Chairman **Dr.K. Arumugam B.E., M.S., M.I.E.,** Correspondent **Shri.M.Balasubramaniam and Joint Correspondent Dr.A.Selvakumar** for all their support and ray of strengthening hope extended. We are immensely grateful to our Principal **Dr.Joseph V.Thanikal M.E., Ph.D., PDF, CEPIT,** for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy,** Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks and gratitude to our project co-ordinator, **Ms.L.S Jayashree M.E., Ph.D.,** Asso.Prof., Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide and extend our gratitude towards **Ms.S.Sathyavathi M.E.,** Lecturer, Department of Information Technology for her helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

ABSTRACT

The World Wide Web is growing at a great speed, but do not form a logical organization and thus making the web page retrieval difficult. So the need for an efficient way to assist in locating the needed information becomes important. One of the ways of providing assistance in locating information is by providing categorized directories. But existing systems such as Yahoo still require human labor for categorizing the web documents. First, manual classification is slow and costly since it relies on skilled manpower. Second, the consistency of categorization is hard to maintain since different human experiences are maintained, since new categories emerge continuously from all the domains. Considering all these problems, the need for automatic classification becomes more important.

The goal of text categorization is to classify documents into a certain number of predefined categories. Evidence shows that Support Vector Machines (SVMs) are well suited for text categorization, outperforming other well-known text categorization methods. A document collection which is already classified by experts is used to train the tool. After the training, any unseen documents can be classified.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1. INTRODUCTION

1.1 GENERAL

Text categorization (TC) is defined as the assignment of natural language texts to one or more predefined categories based on their content. Text categorization is a Natural Language Processing. In the same way that author identification tries to establish the author of a document, and language identification tries to establish the language that a document is written in, text categorization aims to classify the topic or theme of a document. The goal of automatic text classifiers is to be able to classify correctly a previously unseen document. The aim is to minimize the amount of extra work to be done by the user.

Normally the first phase of text categorization is to transform documents into a representation suitable for the learning algorithm. The document should be pre-processed before the categorization starts. Each unique word in the training set of documents is defined as a separate dimension. Each document can then be described by creating a vector where every word that is in the document is represented by a weight. Each unique word w_i corresponds to a feature, with the number of times word w_i occurs in the document as its value. Word stemming, feature selection, stop-word removal and scaling document terms according to their inverse document frequencies are commonly used in the pre-processing of documents. These representation schemes are necessary because it avoids unnecessarily large feature vectors.

1.3 OBJECTIVE OF THE PROJECT

Using the new machine learning approach, the objective is to learn classifiers from examples which perform the category assignments automatically. One such approach introduced by Vapnik has been the use of Support Vector Machines (SVMs). The reason for the project is to ultimately aid any task that involves categorization of text, such as classifying web pages, sorting electronic mail, and learning the interests of the users. Support Vector Machines (SVM) was introduced as a statistical learning theory by Vapnik and his group in 1995 at AT&T Bell Laboratories. They are powerful tools for text categorization. Categorization is achieved by a linear or non-linear separating surface in the input space of the original data set. It is based on the Structural Risk Minimization principle."SVMs use the Vapnik-Chervonenkis (VC) dimension problem to characterize its complexity, which can be independent of the dimensionality of the problem". One property that makes SVMs ideally suitable for TC is that their ability to learn can be independent of the dimensionality of the feature space. SVMs measure the complexity of the hypotheses based on the margin with which they separate the data, not the number of features (Joachims, 1998)

1.2 PROBLEM DEFINITION

The growth of the Internet has caused an exponential increase in the amount of on-line text. As the amount of documents and its users rise, automatic document categorization becomes an increasingly important tool for helping people organize this data. Traditionally the task of assigning a number of appropriate categories to a document was performed manually by domain experts. This is a relatively time-consuming and tedious task. Cost is also an issue. The cost of human experts to work continuously to classify documents. Their capabilities are limited: there is only so much a human can do. They cannot work continuously for 24 hours, and given the sheer volume of text, it is impractical.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 FEASIBILITY STUDY

2.1.1 EXISTING SYSTEM

The main approach used for automated categorization of texts in the late 1980's involved knowledge-engineering automatic classifiers i.e. manually building a set of rules encoding expert knowledge of how to classify documents. However, this had its disadvantages. Rules had to be manually defined by an engineer and with the aid of a domain expert. Additionally, should the set of categories be updated or the classifier recategorized, then these two professional figures must intervene again, and the work repeated (Sebastian, 1999).

POPULAR TEXT CLASSIFIERS

K-NEAREST NEIGHBOUR (KNN)

K-nearest neighbor (KNN) is a well-known statistical approach to text classification. Its logic is quite simple. To classify a new document, the system finds the k documents in the training set that are most similar, and then assigns a category to the document on the basis of these nearest neighbors.

NAÏVE BAYES (NB)

A naïve Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong independence assumptions. Depending on the precise nature of the probability model, naïve Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naïve Bayes models uses the method of maximum likelihood.

DECISION TREES

Decision trees are powerful and popular tools for classification and prediction. The attractiveness of decision trees is due to the fact that, in contrast to neural networks, decision trees represent rules. Rules can readily be expressed so that humar can understand them or even directly used in a database access language like SQL so that records falling into a particular category may be retrieved.

2.1.2 PROPOSED SYSTEM

In the proposed system, a general inductive process automatically builds a classifier for a category c_i by "observing" the characteristics of a set of documents that have been previously classified manually under c_i by a domain expert; from these characteristics, the inductive process gleans the characteristics that a novel document should have in order to be categorized under c_i'. This has its advantages in that the process results in an automatic builder of classifiers. Updating the original set of categories only requires the inductive, automatic construction of a new classifier from a different set of manually categorized documents. The range of application that can be solved using this approach is very large.

2.2 HARDWARE REQUIREMENTS

Processor	: Pentium 4
RAM	: 512 MB
Hard disk	: 40 GB
Internet connection	: 256 kbps and above
Monitor	: 17"
Key Board	: 108 keys

2.3 SOFTWARE REQUIREMENTS

Platform	: Windows NT/2000/XP
Language	: java 2
Software tool	: SVM ^{light}

2.4 SOFTWARE OVERVIEW

FEATURES OF JAVA

PLATFORM INDEPENDENCE

The write-once-run-anywhere ideal has not been achieved, but closer than with other languages. It eliminated the need to port an application to different platform because the same executable code can be run on any machine irrespective of the platform.

OBJECT ORIENTED

In the java fundamental unit, or object, is referred to as a class. A class is a grouping of code that models the behavior of an object in software. All objects have a state, and the state could be on or off. The state of an object is described by instance variables. Variables that are controlled by a class are inaccessible to any other class, except under special conditions. In java, a method is used to control a class, or object. A method is a small body of code that performs a function that can be reused. The function of a class is limited to what is prescribed by its methods. A class must only be created once, after which it may be reused many times. Another instance of the class may be created, without reinventing the object. A method is all that is needed to control any number of instances of a class.

COMPILER/INTERPRETER COMBO

Code is complied to byte codes that are interpreted by Java Virtual Machines (JVM). This provides portability to any machine for which a virtual machine has been written. The two steps of compilation and interpretation allow for extensive code checking and improved security.

SECURITY

Java is a secure environment for several reasons. The first stage in the interpreter is a byte code verification to test whether code confirms to java specification as it is received. Then the interpreter provides a distinct name space for each class that is uploaded, preventing accidental name references. Since there are no pointers when it is compiled, java is immune to memory allocation problems. At run time it does not allow illegal processes, incorrect parameters, or violation of access restrictions.

DYNAMIC BINDING

The linking of data and methods to where they are located is done at run time. New classes can be loaded while a program is running. Linking is done "on the fly". Even if libraries are recompiled, there is no need to recompile the code that uses classes in those libraries. This differs from c++, which uses static binding. This can result in fragile classes for cases where linked code is changed and memory pointers then point to the wrong addresses.

GOOD PERFORMANCE

Interpretation of byte codes slowed performances in early versions, but advanced virtual machines with adaptive and just-in-time compilation and other techniques now typically provide performance up to 50% to 100% the speed of c++ programs.

THREADING

Lightweight processes, called threads, can easily be spun off to perform multiprocessing. It can take advantage of multiprocessors. Multithreading is included at every level in java, beginning at the syntactical level with

synchronization modifiers in the language. Multithreading is the act of executing more than one thread at once while running a single application.

PORTABILITY:

Java provides portability by compiling byte code, which is interpreted on each platform by the run-time environment, or Java Virtual Machine (JVM). The byte code is an executable program for the virtual machine, which exists only in software. The code is then interpreted and executed on the target hardware itself. All code will run on any computer for which an interpreter has been ported.

INHERITANCE

Another important aspect of an object is that each instance inherits the characteristics or properties of the original object. Some aspects of a new class may be changed, while the new class retains all the properties of the old class. The new class becomes the subclass of the original super class. A class is simply a template for future instances of an object.

USED JAVA PACKAGES

- Java swing
- Event handler
- Java utilities
- Java net features

DETAILS OF THE METHODOLOGY EMPLOYED

3. DETAILS OF METHODOLOGY EMPLOYED

3.1 ABOUT SVM

A relatively new learning approach for solving two-class pattern recognition problems. **Support vector machines** (SVMs) are a set of related supervised learning methods used for classification and regression. It is a generalized linear classifier. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.

Support Vector Machines take an inductive approach to a learning task by generalizing from training data. Machine learning initially often requires a large number of labeled training documents for the learning task. One problem is that it is often difficult to create these labeled documents. Either there is little training data available, or when sufficient data is available, there is the arduous task of having to manually categorize unlabelled documents to labeled ones. This method takes into account a particular test set during learning/training phase, and tries to minimize misclassifications of just those particular examples. It uses the information from the test sample by studying the location of those examples, building a structure based on the margin of separating hyper planes on both the training and test set data.

3.2 MODULES

- Content Extraction
- Feature Extraction
- SVM

3.2.1 CONTENT EXTRACTION

The main goal of content extraction is to separate the contents of the web page (i.e) the images, text, tables, links etc., are separated. For classifying a web page, the tables in the web page should be taken separately and only the content should be considered. Some of the images are also considered. This module aims at separating the contents and removing all the HTML tags from the document because the classification is not done based on the formatting. So all the formatting is removed and a plain text file is got as output from this module.

Content extraction is done by implementing a HTML parser for some of the required tags. For example: the <img> tag is considered so that the images are separated, <a> tag is considered so as to separate the HTML links, etc., The extracted content is passed on as input to the next module:"Feature Extraction" .

3.2.2 FEATURE EXTRACTION

The purpose of this module is to extract all the meaningful contents of the web page. This module includes stop words removal, stemming and removing redundancy. Stop words are words like conjunctions, prepositions, etc., which will not help in classifying a page. All these words are removed. We cannot classify a web page based on the stop words. All the stop words are stored in a database and accessed. Information retrieval research suggests that stop-word removal improves classification accuracy. Another reason for removal of stop word is that it avoids unnecessarily large feature vectors.

After the stop words are removed, stemming is done. The words are converted into their root words. For example: words like "write", "wrote", "written", "writing" means a single word "write". Converting all forms of a word to its root word is called stemming. Information retrieval research suggests that stemming improves classification accuracy.

The stemming algorithm employed is "Porter's Stemming Algorithm". The porter stemming algorithm is a process for removing the commoner morphological and in flexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval Systems. It is based mainly on stemming operations that remove suffixes from words, such as gerunds (motoring -> motor), plurals (cats->cat), and replacing words ending with "ator" for example with "ate" (operator->operate), etc..,
These operations are classified into rules where each of these rules deals with a specific suffix and having certain conditions to satisfy. A given word's suffix is checked against each rule in a sequential manner until it matches one, and consequently the conditions in the rule are tested on the stem that may result in a suffix removal or modification.

STOP WORDS REMOVAL

1. It removes prepositions, articles and conjunctions.
2. All the stop words are maintained in a database.
3. It removes matching stop words from the training document.
4. It retains other words for feature generation.

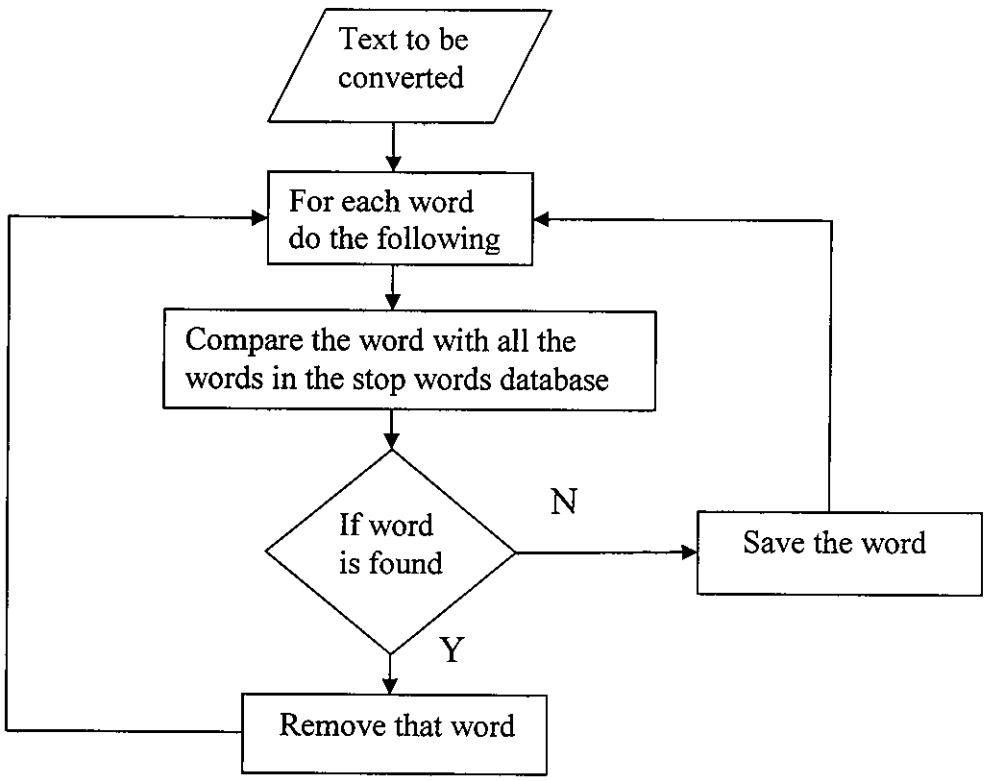

Fig.1. STOP WORDS REMOVAL

WORD STEMMING

1. It merges all non-identical words which refer to the same principle concept.
2. It reduces to their root words.
3. It reduces the number of unique terms by bringing down distinct words to their grammatical root.
4. Here the purpose is to improve the IR system performance.
5. It is done using the porter's stemming algorithm.

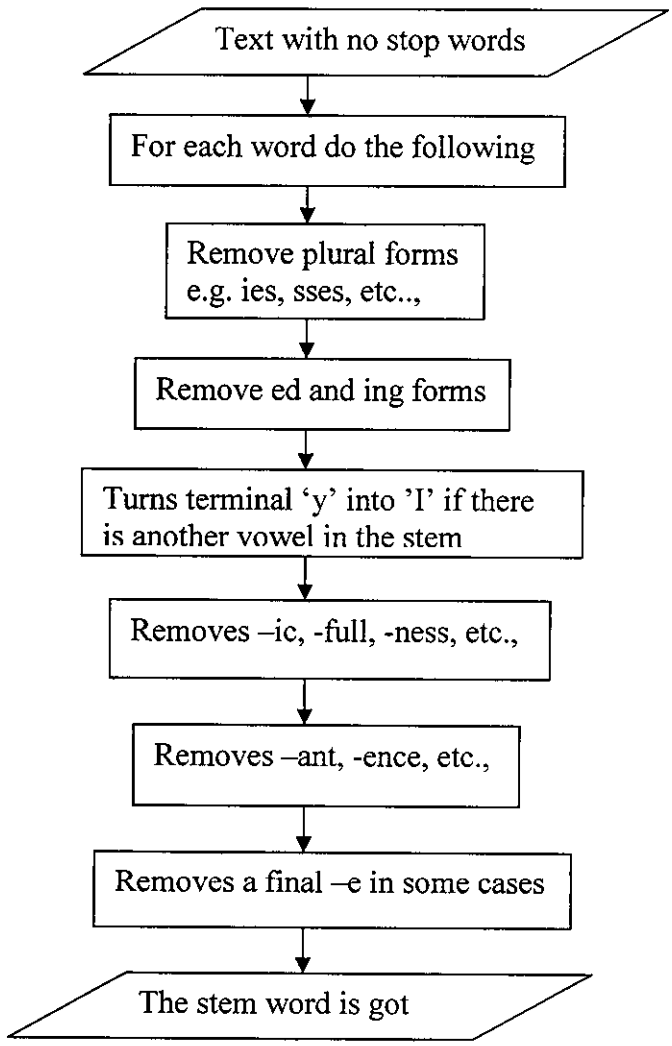

Fig.2. WORD STEMMING

3.2.3 SVM

The text file containing the words cannot be passed to SVM light. The weight of each term should be calculated and the weighted value should be passed to the classifier. The weights are calculated based on the Inverse Document Frequency (IDF) and Document Frequency (DF). For each unique word in the corpus, its inverse document frequency (IDF) was calculated. This involves processing the entire training set and keeping a count of the number of documents each word appears (its document frequency: DF). The formula for computing IDF is as follows

$$IDF(ti) = log\left(\frac{n}{DF(ti)}\right)$$

n= number of documents in the training set

DF= document frequency of the term (ti)

After pre-processing of web pages and giving weights to the entire feature space, the system is ready for the training process. Here each category is treated as a separate binary classification problem. Every training document will be processed through every category. The weights of features are given as input and these weights have to be altered till we obtain the desired result. This is done in a step-wise procedure as follows:

- For every training input calculate the output
  Output=SOP (W * input value)
- If this matches with the desired output, proceed to next training input
- Else alter w; w=w+ Δw, where Δ w=(desired –obtained)input
- Repeat until desired output is obtained for all training inputs.

3.3 OVER ALL DATAFLOW DIAGRAM

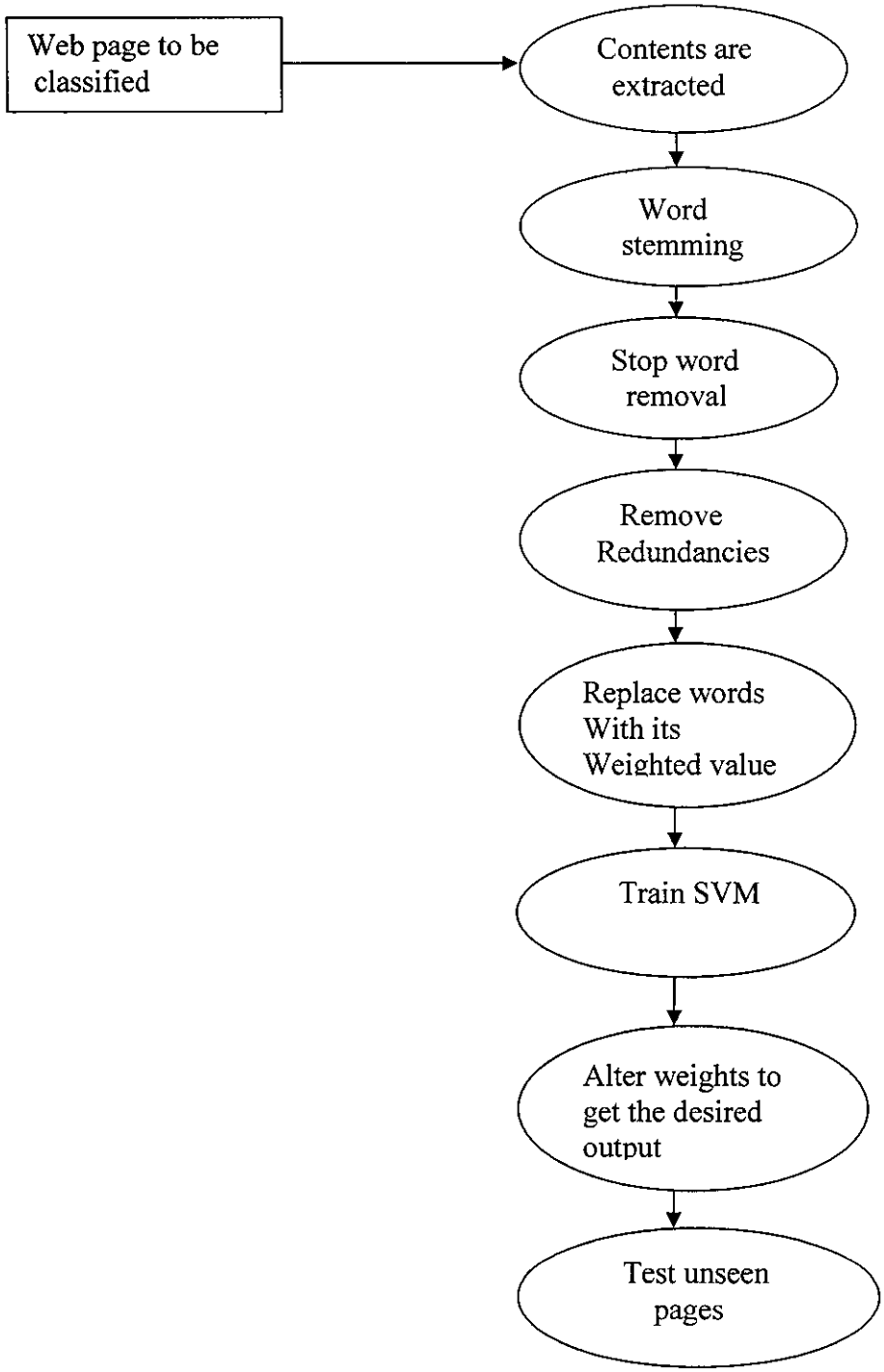

Fig.3. OVER ALL DATA FLOW DIAGRAM

PERFORMANCE EVALUATION

4. PERFORMANCE EVALUATION

4.1 TESTING

4.1.1 UNIT TESTING

A program represents the logical elements of a system. For a program to run satisfactorily, it must compile and test data correctly and tie in properly with other programs. Achieving an error free program is the responsibility of the programmer. Program testing checks for two types of errors: syntax and logical. Syntax error is a program statement that violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax errors. These errors are shown through error message generated by the computer. For logic errors the programmer must examine the output carefully.

When a program is tested, the actual output is compared with the expected output. When there is a discrepancy the sequence of instructions must be traced to determine the problem. The process is facilitated by breaking the program into self-contained portions, each of which can be checked at certain key points. The idea is to compare program values against desk-calculated values to isolate the problems.

4.1.2 FUNCTIONAL TESTING

Functional testing of an application is used to prove the application delivers correct results, using enough inputs to give an adequate level of confidence that will work correctly for all sets of inputs. The result after execution should give the accurate result.

4.1.3 WHITE BOX TESTING

White box testing, sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using white box testing method, the software engineer can derive test cases.

The white box testing should exercise all logical decisions on their true and false sides and all the logical decisions must be valid. It should execute all loops at their boundaries and within their operational bounds. This will prove that all loops are finite. It should also exercise all internal data structures to ensure their validity.

4.1.4 BLACK BOX TESTING

Black box testing also called behavioral testing, focuses on the functional requirements of the software. That is, black testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black box testing is not alternative to white box techniques. Rather it is a complementary approach that is likely to uncover a different class of errors than white box methods. Black box testing attempts to find errors in the following categories.

Black box testing is done to check for incorrect or missing functions, to check for interface errors, to check for errors in data structures or external data base access and to check for initialization and termination errors.

4.2 EVALUATION

Category assignments can be evaluated using a two-way contingency table for each category:

		Expert Judgements	
		YES is correct	NO is correct
Classifier Judgement	Assigned YES	TP	FP
	Assigned NO	FN	TN

True Positives (TP)-documents correctly assigned to this category.
False Positives (FP)-documents incorrectly assigned to this category.
False Negatives (FN)-documents incorrectly rejected from this category.
True Negatives (TN)-documents correctly rejected from this category.

Conventional performance measures are then used. These include precision, recall and accuracy.

PRECISION

Precision is the proportion of documents placed in the category that are really in the category.

$$\text{Precision} = TP / (TP+FP)$$

RECALL

Recall is the proportion of items in the category that are actually placed in the category.

$$\text{Recall} = TP / (TP+FN)$$

ACCURACY

Accuracy is the proportion of the total number of items that are correctly placed in the category.

$$\text{Accuracy} = (TP+TN)/n$$

(where $n=TP+FP+FN+TN$)

These performance measures can be misleading when examined alone. For example, a system that assigns YES to every category for every document will have a perfect recall of 100% and a very low score for precision. Quiet often a combined measure which gives a single-numbered performance measure is more appropriate. One such measure is the F_1 measure which balances precision and recall in a way that gives them equal weight.

$$F_1=2rp/(r+p)$$

The SVM^{light} system output values in percentage for accuracy, precision and recall. The computed F_1 measure will be used as a measure of performance.

PERFORMANCE EVALUATION OF PRECISION & RECALL:

Evaluation of Precision and Recall of SVM Light with respect to number of features selected. It is noted that as the number of features selected are increased, both the Precision and Recall are increased.

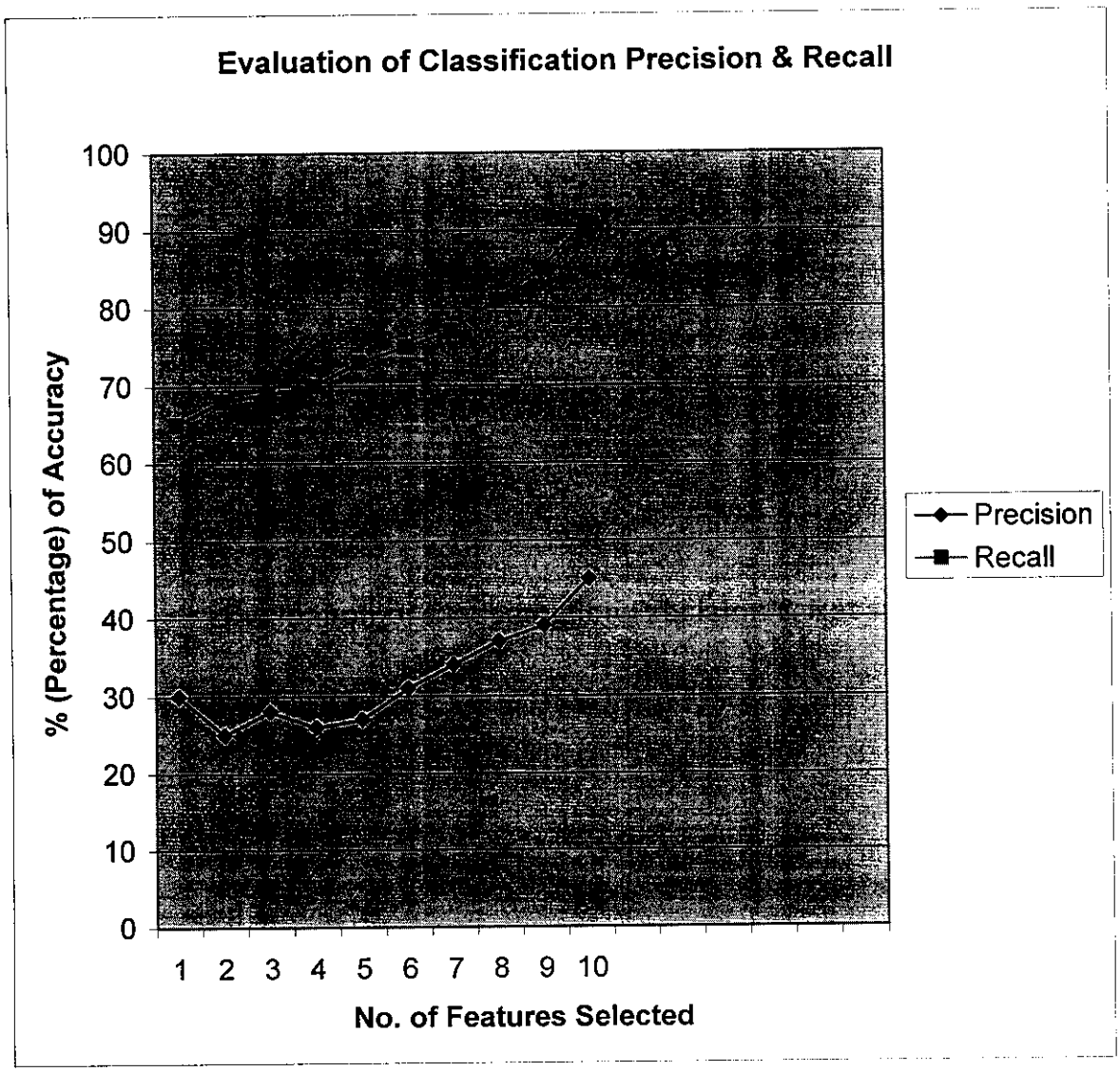

Fig.4. Evaluation of Precision and Recall

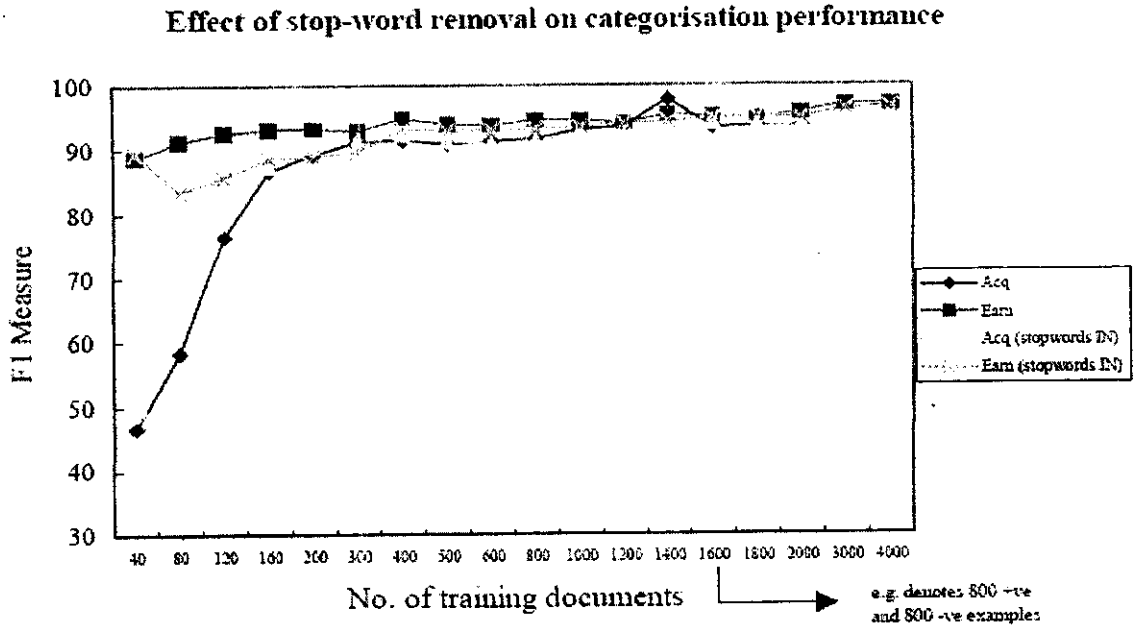

Fig.5. EFFECT OF STOP WORDS REMOVAL

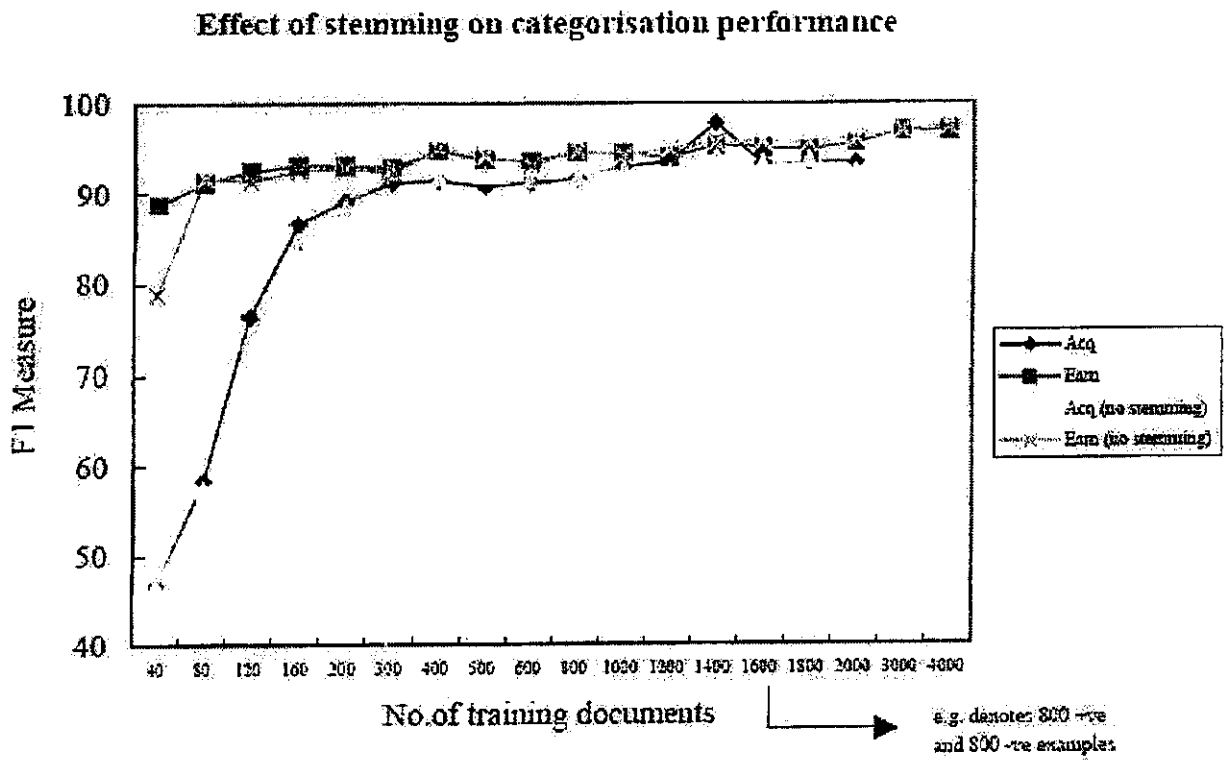

Fig.6. EFFECT OF WORD STEMMING

CONCLUSION

5. CONCLUSION

Accurate text classifiers can be learned automatically from a set of training examples which have already been classified by a human. The use of support vector machines has shown impressive results in the categorization of the training dataset. Using the SVM^{light} system, experimental results show that the larger the training set size, the better the classification accuracy.

The minimum number of training examples required to achieve an F_1performance measure of around 90 percent is around 300 training examples. This minimum level was consistent over the certain common training categories tested, although some categories achieved high levels of performance before this level. Therefore certain categories are easier to learn than others.

Support vector machines are able to generalize well in high dimensional feature spaces. The pre-processing of documents has helped in achieving a better performance compared to the other systems which categorizes without document pre-processing. Thus the performance of our proposed system has been increased.

FUTURE ENHANCEMENTS

6. FUTURE ENHANCEMENTS

The project categorizes the web pages into some pre-defined categories. But it does not support dynamic inclusion of categories. Sub categories are not included. These improvements can be done in the future enhancements of the project. The following can be enhanced:

- To increase the number of target categories
- To include sub categories
- To increase the feature space
- To obtain better accuracy values
- To include dynamic inclusion of new categories
- To include other parameters such as images also to categorize documents.

APPENDICES

7. APPENDICES

7.1 SAMPLE CODE

```
package org.htmlparser;
import java.util.*;
import java.io.*;

class MyString
{
public static String wholestr;
public static String substr;
}

public class ContentExtract
{

public static Vector vecimg=new Vector();
//public static Vector veca=new Vector();
//public static Vector vecscript=new Vector();
//public static Vector vecstring=new Vector();
public static Vector vectable=new Vector();

public static Vector finalimg=new Vector();

public static HashSet hs;

public static String wholestr;

public static String spacerem;

public static int linkcount;

public static void main(String arg[])
{
String[]
input=arg;//{"http://localhost/EBanking/amazon1.htm","http://localhost/EBanking/amazon2.htm
"};//,"http://localhost/EBanking/amazon3.htm","http://localhost/EBanking/amazon4.htm","http://
localhost/EBanking/amazon5.htm"};
String[] tags={"img","a","script","string","table"};

Vector htmltags=new Vector();
```

```
htmltags.addElement("<html><body>");
htmltags.addElement("</body></html>");
htmltags.addElement("<font color=red>");
htmltags.addElement("</font>");
htmltags.addElement(">");
htmltags.addElement("<");
htmltags.addElement("<tr><td>");
htmltags.addElement("</td></tr>");
htmltags.addElement("</table></body></html>");
htmltags.addElement("<html><body><table>");
htmltags.addElement("</a>");
String sr="<html><body><script type="+"text/javascript"+">";
htmltags.addElement(sr);
htmltags.addElement("</script></body></html>");

//////////////////////////this is for Image Extraction//////////////////////


for(int j=0;j<input.length;j++)
{

String[] temparr={input[j],"img"};
Vector vectemp=org.htmlparser.Parser.main(temparr);
vectemp.removeAll(htmltags);
vecimg.addElement(vectemp);
}

for(int i=0;i<vecimg.size();i++)
{

Vector temp=(Vector)vecimg.elementAt(i);
for(int k=0;k<temp.size();k++)
{
String str=(String)temp.elementAt(k);
finalimg.addElement(str);

}

}
//System.out.println(ContentExtract.finalimg);
//Vector remfinalimg=RemoveDubli.main(finalimg);
System.out.println("vecimg="+finalimg.size());
hs=new HashSet(finalimg);
System.out.println("HashSet="+hs.size());

Vector totalimg=new Vector(hs);
```

```
MyString[] mystr=new MyString[totalimg.size()];

TreeMap tmimg=new TreeMap();
try
{
PrintStream myout=new PrintStream("ImgFileNames.txt");

for(int x=0;x<totalimg.size();x++)
{
System.out.println();
int i=0;
String tempo=(String)totalimg.elementAt(x);
mystr[i].wholestr=tempo;
String subtempo=tempo.substring(tempo.lastIndexOf("/")+1);

String subsub=null;
System.out.println(tempo+"===="+subtempo);


if(tempo.contains("write"))
{
//System.out.println("TWO");
subsub=tempo;
//System.out.println("THREE");
}

//else
if(tempo.contains("alt")||tempo.contains("style")||tempo.contains("border")||tempo.contains("id")|
|tempo.contains("align")||tempo.contains("onmou")||tempo.contains("name")||tempo.contains("on
cli")||tempo.contains("usemap"))
else if(tempo.contains(".jpg"))
{
//System.out.println("ZERO");

String str1=tempo.substring(0,tempo.indexOf(".jpg"));

System.out.println("str1= "+str1);
String str2=str1.substring(str1.lastIndexOf("/")+1,str1.length());
System.out.println("str2= "+str2);
//subsub=subtempo.substring(0,subtempo.indexOf(""")-1);
subsub=str2+".jpg";

//        System.out.println("ONE");
}
else if(tempo.contains(".gif"))
```

```
{
//System.out.println("ZEROgif");

String str1=tempo.substring(0,tempo.indexOf(".gif"));
String str2=str1.substring(str1.lastIndexOf("/")+1,str1.length());
//subsub=subtempo.substring(0,subtempo.indexOf("")-1);
subsub=str2+".gif";
//System.out.println("ONEgif");
}
else
{
subsub=tempo;
}

myout.println(subsub);
mystr[i].substr=subsub;



tmimg.put(subsub,tempo);
i++;
}
}
catch(Exception e){}

System.out.println("TreeMap Size="+tmimg.size());

Set set=tmimg.entrySet();
Iterator itrimg=set.iterator();
try
{
PrintStream imgnames=new PrintStream("imgnames.txt");
PrintStream outimg=new PrintStream("PrimaryImages.html");
outimg.println("<html><body>");

while(itrimg.hasNext())
{
Map.Entry me=(Map.Entry)itrimg.next();
String imgtemp=(String)me.getValue();
String mytemp=(String)me.getKey();
//out.print("<tr><td>");
outimg.print("<");
outimg.print(imgtemp);
imgnames.println(mytemp);
outimg.println(">");
//out.print("</tr></td>");
```

```
}

outimg.println("</html></body>");
outimg.close();
}
catch(Exception e)
{
}
////////////////////////////////////End Of image Extraction//////////////////////////////////////////

////////////////////////////////////Start of LinkExtraction//////////////////////////////////////////


Vector veca=new Vector();
try
{
PrintStream outa=new PrintStream("PrimaryLinks.html");
outa.println("<html><body>");
outa.println("<table>");

Vector finala=new Vector();
for(int j=0;j<input.length;j++)
{

String[] temparr1={input[j],"a"};
Vector vectemp=org.htmlparser.Parser.main(temparr1);
vectemp.removeAll(htmltags);
System.out.println(vectemp);
//System.out.println("vectemp.size()="+vectemp.size());
veca.addElement(vectemp);
}

for(int i=0;i<veca.size();i++)
{
Vector temp=(Vector)veca.elementAt(i);
for(int k=0;k<temp.size();k++)
{
String str=(String)temp.elementAt(k);
finala.addElement(str);
}

}

System.out.println("finala.size= "+finala.size());
//Vector remfinala=RemoveDubli.main(finala);
HashSet hsa=new HashSet(finala);
```

```
System.out.println("HSA Size= "+hsa.size());
Vector redv=new Vector(hsa);
linkcount=redv.size();

for(int r=0;r<redv.size();r++)
{
String strredv=(String)redv.elementAt(r);
String none1="a name";
String none2="a id";
String none3="a style";
if(strredv.contains(none1)||strredv.contains(none2)||strredv.contains(none3))
{
redv.removeElementAt(r);
}
}


for(int h=0;h<redv.size();h++)
{
outa.println("<tr><td>");
outa.print("<");
String linku=(String)redv.elementAt(h);
outa.print(linku);
outa.print(">");
outa.print(linku);
outa.println("</a>");
outa.println("</td></tr>");
}
outa.println("</table>");
outa.println("</body></html>");
outa.close();
}
catch(Exception ee)
{
}

////////////////////////////////////End of LinkExtraction////////////////////////////////////


////////////////////////////////////Start for Script Extraction////////////////////////////////

Vector vecscript=new Vector();

try
{
PrintStream outscript=new PrintStream("PrimaryScript.html");
```

```
outscript.println("<html><body>");



Vector finalscript=new Vector();
for(int j=0;j<input.length;j++)
{

String[] temparr2={input[j],"script"};
Vector vectemp2=org.htmlparser.Parser.main(temparr2);
vectemp2.removeAll(htmltags);
System.out.println(vectemp2);
//System.out.println("vectemp.size()="+vectemp.size());
vecscript.addElement(vectemp2);
}

for(int i=0;i<vecscript.size();i++)
{
Vector temp=(Vector)vecscript.elementAt(i);
for(int k=0;k<temp.size();k++)
{
String str=(String)temp.elementAt(k);
finalscript.addElement(str);
}

}
System.out.println("finalscript.size= "+finalscript.size());

HashSet hsscript=new HashSet(finalscript);
System.out.println("HSScript Size= "+hsscript.size());
Vector scrv=new Vector(hsscript);

for(int h=0;h<scrv.size();h++)
{

outscript.println("<script>");
String scriptu=(String)scrv.elementAt(h);
outscript.println(scriptu);
outscript.println("</script>");

}

outscript.println("</body></html>");
outscript.close();
}
catch(Exception ee)
```

```
{
}

////////////////////////////////////End of Script Extraction//////////////////////////////

////////////////////////////////////Start of String Extraction//////////////////////////////

Vector vecstring=new Vector();

try
{
PrintStream outstring=new PrintStream("PrimaryStrings.html");
PrintStream outstring1=new PrintStream("PrimaryStrings.txt");
outstring.println("<html><body>");

Vector finalstring=new Vector();

for(int j=0;j<input.length;j++)
{

String[] temparr3={input[j],"string"};
Vector vectemp3=org.htmlparser.Parser.main(temparr3);
vectemp3.removeAll(htmltags);
System.out.println(vectemp3);
//System.out.println("vectemp.size()="+vectemp.size());
vecstring.addElement(vectemp3);
}

for(int i=0;i<vecstring.size();i++)
{
Vector tempstr=(Vector)vecstring.elementAt(i);
for(int k=0;k<tempstr.size();k++)
{
String strstr=(String)tempstr.elementAt(k);

finalstring.addElement(strstr);
}

}
System.out.println("finalstring.size= "+finalstring.size());

//Vector remfinalstring=RemoveDubli.main(finalstring);

HashSet hsstring=new HashSet(finalstring);
System.out.println("HSSTRING Size= "+hsstring.size());
Vector strinv=new Vector(hsstring);
```

```
for(int l=0;l<strinv.size();l++)
{
String hstem=(String)strinv.elementAt(l);
if(hstem.contains("img"))
{
strinv.removeElementAt(l);
}

}


for(int z=0;z<strinv.size();z++)
{
String hstr=(String)strinv.elementAt(z);
wholestr+=hstr+",";

}

System.out.println(wholestr);

StringTokenizer strtok1=new StringTokenizer(wholestr,", ");


while(strtok1.hasMoreElements())
{
String temps=strtok1.nextToken();
if(temps!=null)
{
outstring.print("<font color=red>");
outstring.print(temps);
outstring1.print(temps+" ");
outstring.println("</font>");
spacerem+=temps;
}
}

System.out.println(spacerem);


/*for(int h=0;h<strinv.size();h++)
{

//outscript.println("<script>");
String stringu=(String)strinv.elementAt(h);
```

```
outstring.println("<font color=red>");
System.out.println(stringu);
outstring.println(stringu);
outstring.println("</font>");

}*/

outstring.println("</body></html>");

}
catch(Exception ee)
{
}


System.out.println("LinkCount= "+linkcount);
////////////////////////////////End of String Extraction//////////////////////////////

//////////////////////////////Table Extraction/////////////////////////////////

Vector vectable=new Vector();

try
{
PrintStream outtable=new PrintStream("Primarytables.html");
outtable.println("<html><body>");



Vector finaltable=new Vector();
for(int j=0;j<input.length;j++)
{

String[] temparr3={input[j],"table"};
Vector vectemp3=org.htmlparser.Parser.main(temparr3);
//Vector vectemp3=UniqueTables.main();
vectemp3.removeAll(htmltags);
System.out.println(vectemp3);
//System.out.println("vectemp.size()="+vectemp.size());
vectable.addElement(vectemp3);
}

for(int i=0;i<vectable.size();i++)
{
Vector temp=(Vector)vectable.elementAt(i);
for(int k=0;k<temp.size();k++)
```

```
{
String str=(String)temp.elementAt(k);
finaltable.addElement(str);
}

}
System.out.println("finaltable.size= "+finaltable.size());

HashSet hstable=new HashSet(finaltable);
System.out.println("HSScript Size= "+hstable.size());
Vector scrt=new Vector(hstable);

for(int h=0;h<scrt.size();h++)
{

//outscript.println("<script>");
String tableu=(String)scrt.elementAt(h);
outtable.println(tableu);
//outscript.println("</script>");

}

outtable.println("</body></html>");
outtable.close();
}
catch(Exception ee)
{
}




////////////////////////////End of Table Extraction//////////////////////////




 }
}
```

7.2 SCREEN SHOTS

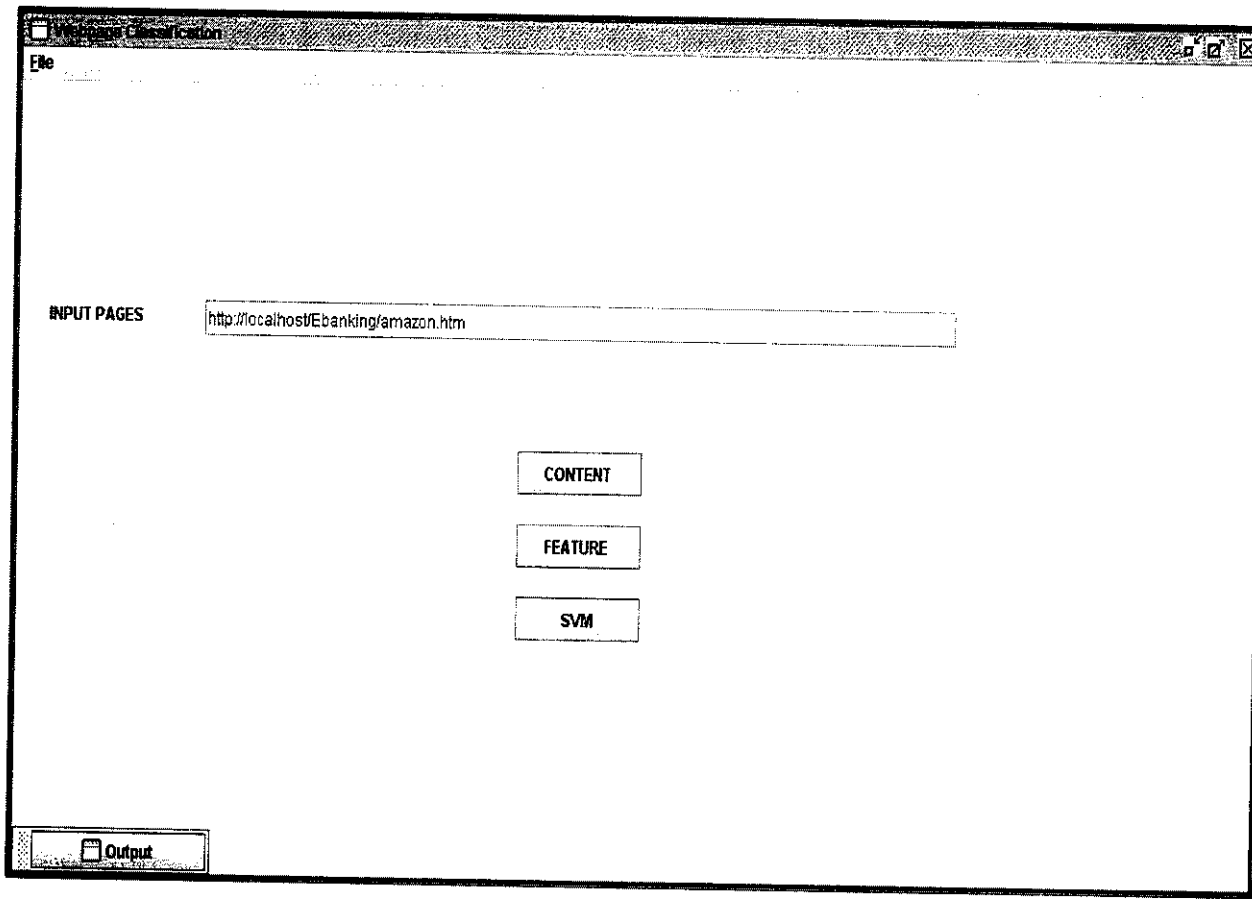

Fig.7. USER INTERFACE

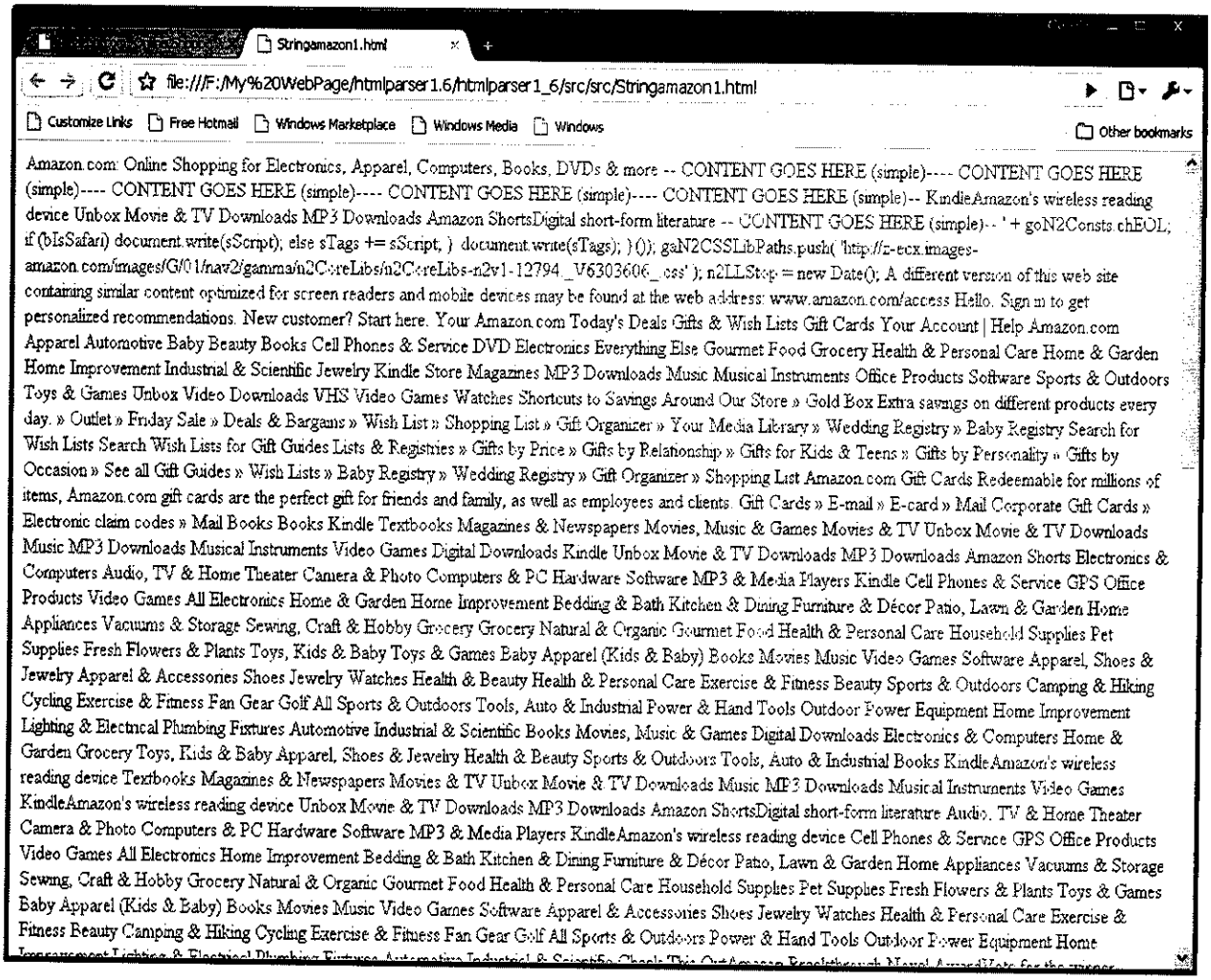

Fig.8. EXTRACTED STRINGS

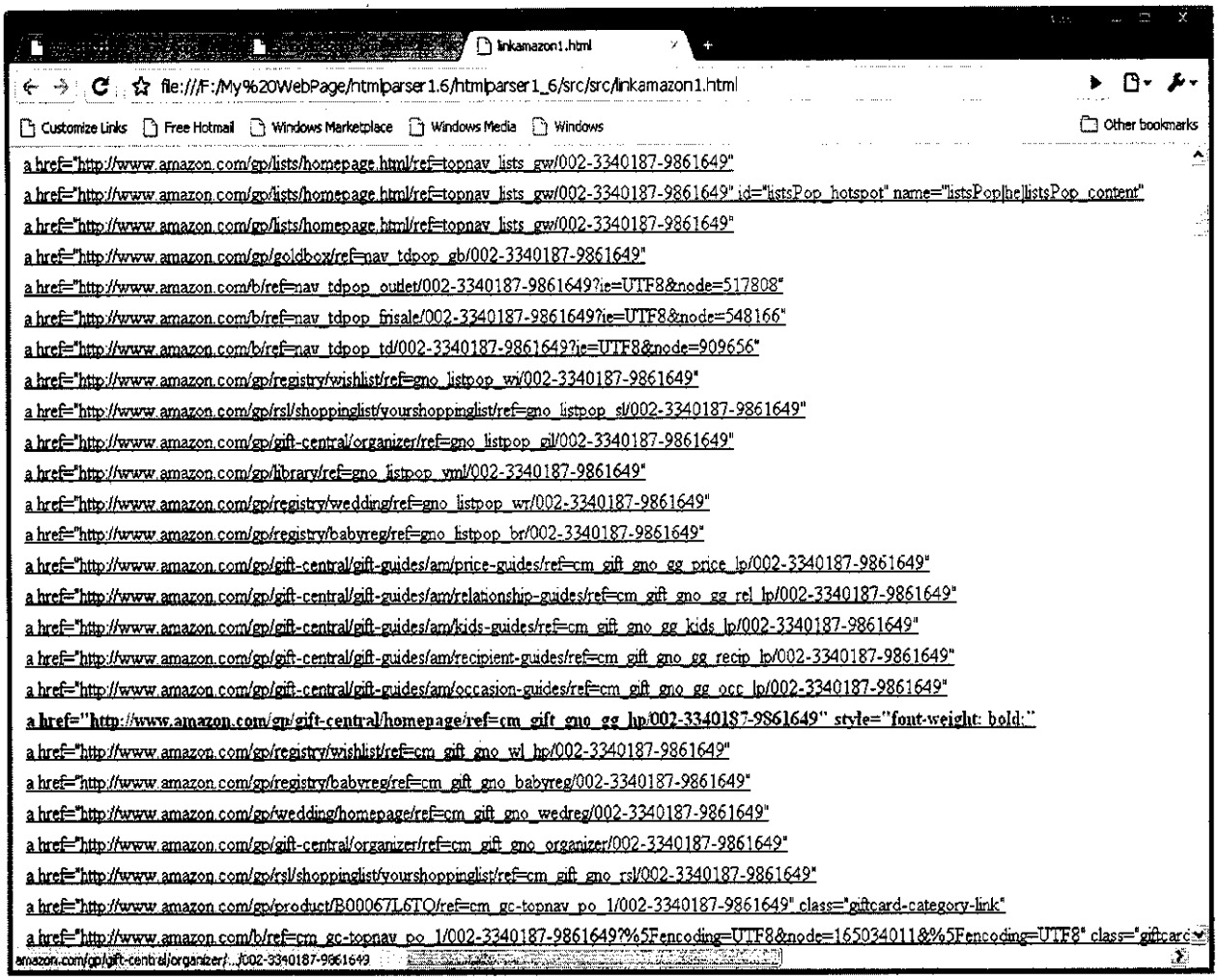

Fig.9. EXTRACTED LINKS

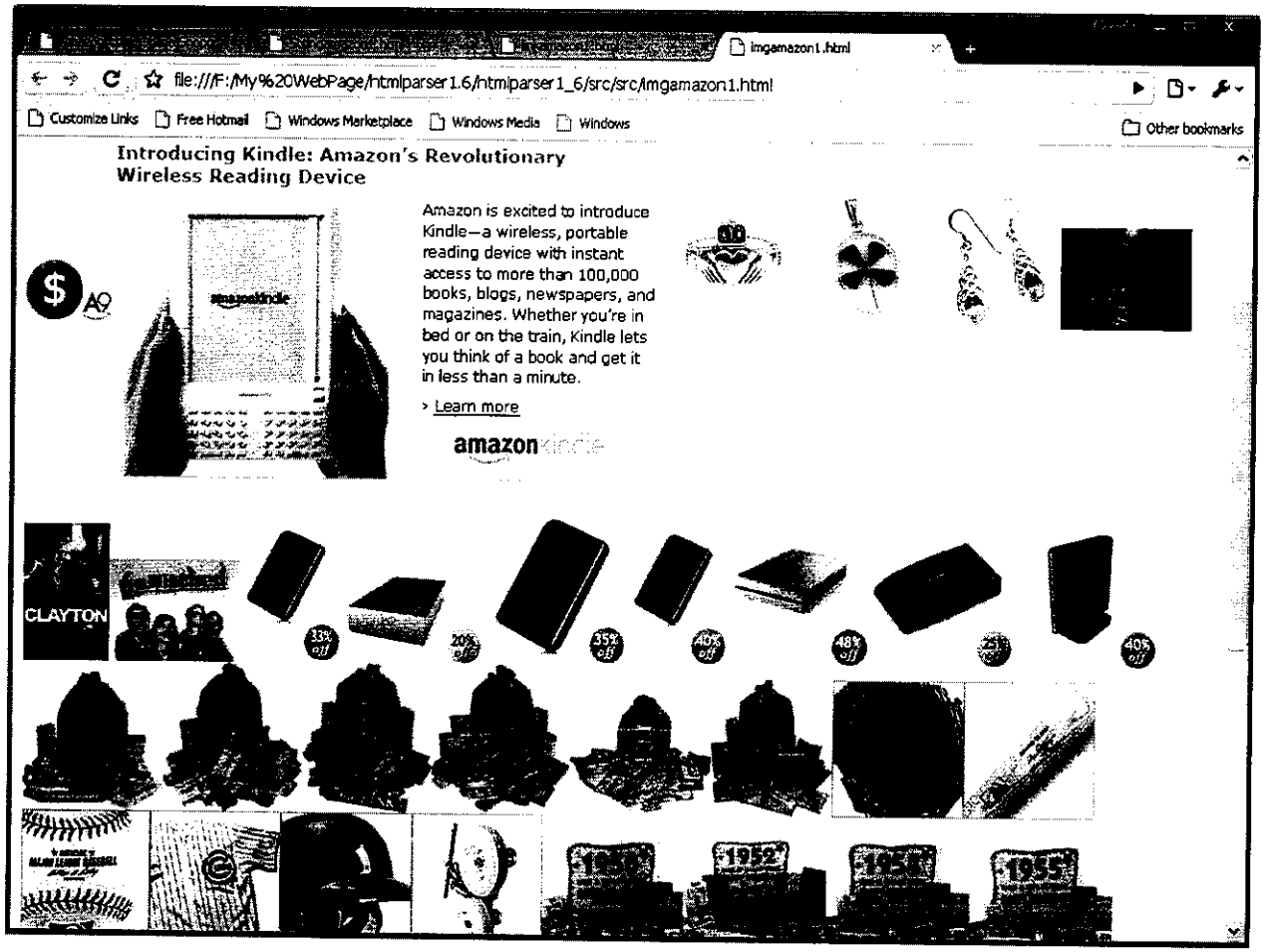

Fig.10. EXTRACTED IMAGES

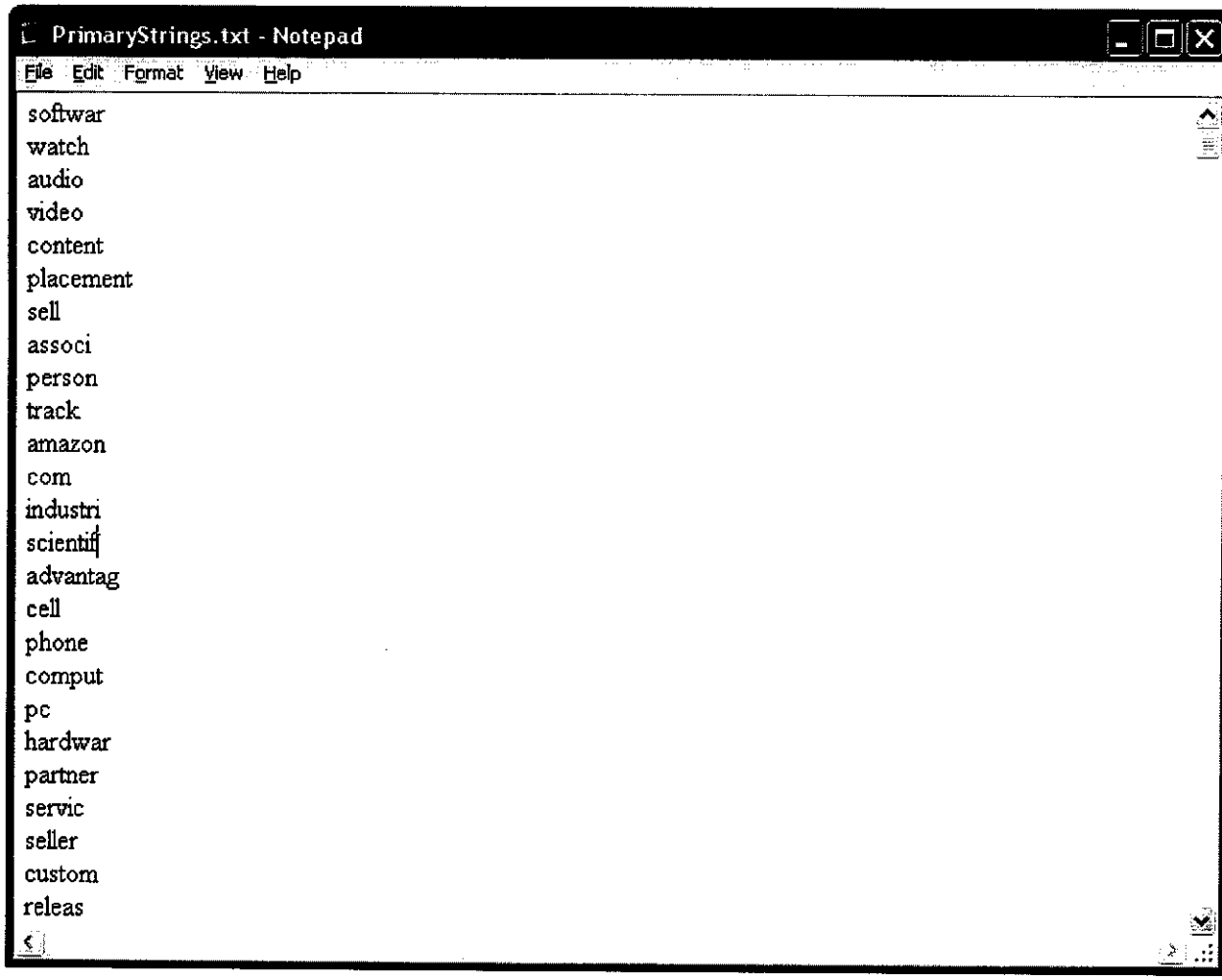

Fig.11. PRIMARY STRINGS

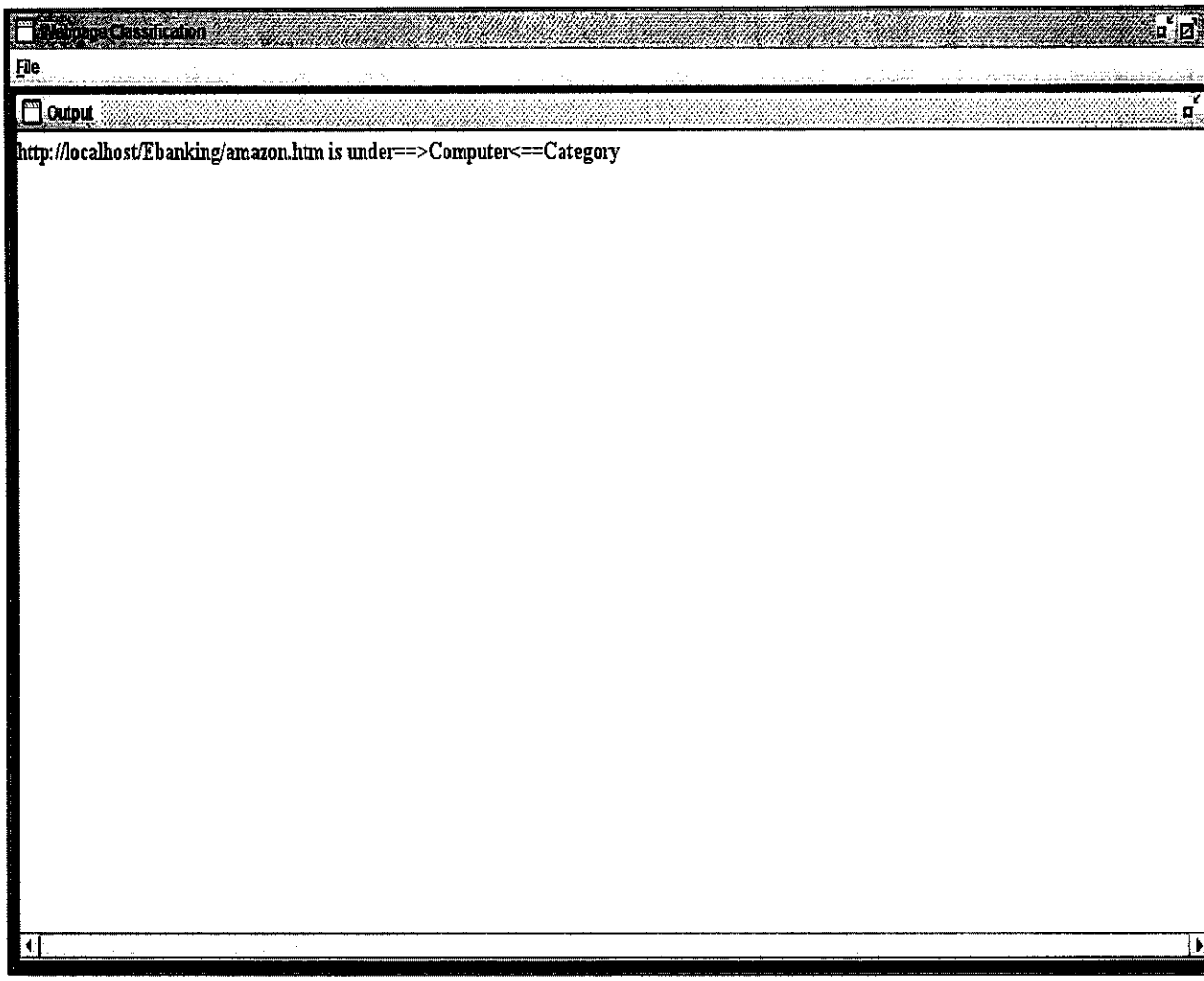

Fig.12. OUTPUT

REFERENCES

8. REFERENCES

- Ngai Tang, " Text Categorization using Support Vector Machines", 30 August 2001
- Cooley.R "Classification of news stories using Support Vector Machines" in IJCAI'99 workshop on Text Mining, Stockholm, Sweden, Aug 2 1999
- Joachims T, "Transductive Inference for Text Classification using Support Vector Machines" in International conference on Machine Learning, 1999
- Schutze H, Hull d.a and Pedersen J.O ," A comparison of classifiers and document representations for the document routing problem", in proceedings of SIGIRI-95, 18th ACM international conference on research and development on Information Retrieval. Pp229-237, Seattle, US, 1995.
- Sebastiani F, " A tutorial on Automated Text Categorization". In Analia Amandi and Ricardo Zunino, editors, proceedings of ASAI-99, 1st Argentenian Symposium on Artificial Intelligence, pages 7-35, Buenos Aires, AR,1999.
- Domonokos Tikk, Gyorgy Bir, Jae Dong yang , " A hierarchical text categorization approach and its application to FRT expansion" Hungary, 20 April 2003
- www.svms.org
- www.joachims_svm.com