

PAGE RANK USING DOCUMENT RELEVANCE MEASURE

A PROJECT REPORT

Submitted by

**A.RAVI KUMAR
B.VIGNESH**

**(11309205074)
(11309205103)**

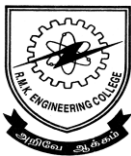
in partial fulfillment for the award of the degree

of

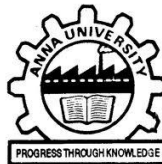
BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



RMK ENGINEERING COLLEGE, CHENNAI



ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2013

BONAFIDE CERTIFICATE

Certified that this project report on “ **MODIFIED PAGE RANK USING DOCUMENT RELEVANCE MEASURE** ” is the bonafide Work of “ **A.RAVIKUMAR (11309205074) , B.VIGNESH (11309205103)** ” who carried out the project under my supervision.

SIGNATURE

Dr.K. VIJAYA M.E., Ph.D
HEAD OF THE DEPARTMENT

Dept. of Information Technology,
RMK Engineering College,
R.S.M. Nagar,
Kavaraipettai-601206.

SIGNATURE

Mrs. **P.CHITRA M.E**
SUPERVISOR

Dept. of Information Technology,
R.M.K Engineering College,
R.S.M. Nagar,
Kavaraipettai-601206.

CERTIFICATE OF EVALUATION

College Name : RMK ENGINEERING COLLEGE

Department : INFORMATION TECHNOLOGY

Semester: 08

Title of Project	Name of the Students	Name of the Supervisor with designation
MODIFIED PAGE RANK USING DOCUMENT RELEVANCE MEASURE	A.RAVI KUMAR B.VIGNESH	Mrs.P.CHITRA Asst. Professor

The report of the project work submitted by the above students in partial fulfillment for the award of Bachelor of Technology Degree in INFORMATION TECHNOLOGY of Anna University was evaluated and confirmed to be the report of the work done by the above students and then evaluated.

Submitted the project during the viva voce held on
.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

At the outset, we would like to express our gratitude to our beloved and respected **Chairman, Thiru.R.S.Munirathnam** for his support and blessings to accomplish the project.

We would like to express our thanks to our **Vice Chairman Thiru.R.M.Kishore** for his encouragement and guidance.

We thank our **Principal, Dr. Elwin Chandramonie**, for creating the wonderful environment for us and enabling us to complete the project.

We wish to express our sincere thanks and gratitude to **Dr.K. Vijaya, M.E. Ph.D., Head, Department of Information Technology** who has been a guiding force and constant source of inspiration to us.

We express our sincere gratitude and thanks to our beloved **Project Guide Mrs. P. Chitra** and the **Project Coordinator Mr. K. Ponniyin Selvan** for having extended their fullest co-operation and guidance without which this project would not have been a success.

Our thanks to all faculty and non teaching staff members of our department for their constant support to complete this project.

ABSTRACT

The objective of this project is designing a search-engine which assures higher rank to the document that is relevant to each other on the web. Page rank distributes equal share of the rank score to its entire out-going links in which all pages are treated equal irrespective of their relevance. This project work focuses on distributing the rank score to the out-going link pages according to their relevance with the parent page.

The project covers the administrator and the end user who benefits from the search engine. The working of the entire project begins at the administrator site where the processing of web pages begin and after further processing the required data is stored in the database, when a user searches for a page using the search-engine the data available in the database is searched and only the necessary data is displayed on the clients side.

Administrator is responsible for pre-processing where the text based web pages are downloaded and stored. The html tags in a particular web page is first removed and these pages are given to another code which removes the unnecessary stop words available in the document which is further given as input for code which stems the words to their respective root words. The next process is reading each and every document and gets the words that have occurred and the number of times they have occurred and store it in the data base which is used to calculate the rank for a particular page and store it in the database.

When a user searches for a key word in the search engine the key word typed is compared with the key words available in the database and the pages available for a particular key word are listed in decreasing order of page rank as well as relevance based page rank.

TABLE OF CONTENTS

CHAPTER. NO.	TITLE	PAGE NO.
	ABSTRACT	v
1	INTRODUCTION	
1.1	OBJECTIVE AND SCOPE	1
1.2	OVERALL DESCRIPTION	1
1.3	PRODUCT PRESPECTIVE	2
1.4	PRODUCT FEATURES	2
1.5	EXISTING SYSTEM	2
1.6	PROPOSED SYSTEM	2
2	SYSTEM STUDY AND ANALYSIS	
2.1	FEASIBILITY STUDY	3
2.2	LITERATURE REVIEW	3
2.3	PROBLEM DESCRIPTION	5
2.4	HARDWARE & SOFTWARE REQUIREMENTS	5
2.5	FUNCTIONAL REQUIREMENTS	6
3	SYSTEM DESIGN	
3.1	SYSTEM ARCHITECTURE	9
3.2	OVERALL BLOCK DIAGRAM	10

3.3	DATABASE DESIGN	11
3.4	ER DIAGRAM	12
3.5	CLASS DIAGRAM	13
3.6	ACTIVITY DIAGRAM	14
3.7	SEQUENCE DIAGRAM	16
3.8	USE CASE DIAGRAM	17
4	SYSTEM IMPLEMENTATION	
4.1	MODULE DESCRIPTION	19
4.1.1	INPUT MODULE	
4.1.2	PREPROCESSING	
4.1.3	RANKING	
5	TESTING	
5.1	TEST CASES AND TEST RESULTS	21
6	CODING	
6.1	HTML TO TEXT CONVERTOR	22
6.2	STOP WORDS	23
6.3	STEMMING	33
6.4	WORD COUNT	44
6.5	MYFRAME	62
6.6	LOGIN	78
6.7	RANKING	85
6.8	WORDNET DEMO	91

7	SCREENSHOTS	
7.1	LOGIN FOR ADMINISTRATOR	95
7.2	ADMINISTRATOR FUNCTIONS	95
7.3	REMOVING HTML TAGS	96
7.4	STEMMED WORDS	97
7.5	REMOVING STOPWORDS	98
7.6	EMPTY DATABASE FOR STORING WORD COUNT VALUES	99
7.7	DATABASE UPDATED WITH WORDS AND FREQUENCY	99
7.8	COLLECTING THE WORDS WITH DIFFERENT ID'S	100
7.9	DATABASE UPDATED WITH URL AND RANK	100
7.10	DATABASE UPDATED WITH RELEVANCE BETWEEN THE EACH DOCUMENT	101
7.11	USER'S SEARCH ENGINE	102
7.12	RANK BASED DOCUMENTS URL'S	103
7.13	DOCUMENT IN THE WEB BROWSER	104
8	CONCLUSION AND FUTURE WORK	104
9	REFERENCES	104

MODIFIED PAGE RANK USING DOCUMENT RELEVANCE MEASURE

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE AND SCOPE

The main objective of the project is to improve the present page ranking method so as to facilitate the user with the better and more relative search results that concern with the search query. The change that is made to the existing page ranking method has a good scope in the field of search engine for improving the results of search query. Also with the summary generation process the user can compare with all the result related to his search requested.

1.2 OVERALL DESCRIPTION

A search engine is a web based tool that simplifies the search for a specific requirement on the web. Search engine works based on the principle of fetching the pre-processed data from the storage space based on the user's query. In the new methodology page rank is calculated based on relevance factor between two pages than calculating rank based on outgoing links. In a text based search the web-based text document is processed and rank algorithm is applied to obtain a rank for the page bases on its relevance with the parent document which is stored in the data base. When a user searches for a result with a query, then it is searched in the database and the result is displayed based on descending order of page rank. Further the user can sort out a sort summary of selected websites of his own list.

Also the user can choose the type of summary s/he wants (i.e) a short version of a brief one .

1.3 PRODUCT PRESPECTIVE

The product will demonstrate the working of a search engine by using the data available for pre-processing using the formula and demonstrate the difference between the search result obtained from exiting system and search result of the improved page ranking system. This output of the search results is used for the summary generation which is created and displayed accordingly.

1.4 PRODUCT FEATURES

The improved page rank has the advantage of ranking the pages based on relevance measure than ranking based on the outgoing links in the document. In this methodology we can see the pages listed in the order of decreasing order of rank where pages get ranked based on the relevance between them and the parent document. Using the relevance of each document and the keyword occurrence the Summary is generated for accordingly to its search keyword .

1.5 EXISTING SYSTEM

Page rank is being assigned based on link analysis algorithm that is generally used by the search engines to assign a numerical weight to each hyperlinked within. Page ranking basically identifies the relevance between pages. This algorithm arises from the method based on graph where the web pages are considered as nodes and its hyperlinks as edges of a tree structure. In the summary generation process the basic file reading with domain administrator created summaries were used .

1.6 PROPOSED SYSTEM

The proposed system is where the hit rate for a more relevant page is improved based on relevance measure which is calculated by taking the number of occurrence of terms in a particular document as a factor and calculating its relevance with the page in which its hyper link is present. The document which has more number of relevant words will be ranked as the most relevant page. Taking the same results the documents will be summarized eventually of time to time basis.

CHAPTER 2

SYSTEM STUDY AND ANALYSIS

2.1 FEASIBILITY STUDY

The current technique of calculating the rank based on outgoing links used for calculating the page rank gives equal rank to all pages connected to a page irrespective of their relevance with the parent document, so the new system brings in difference by using relevance based measure by calculating the number of occurrence of words in the document and their relevance to the parent document. So this system improves the result of search and is also feasible.

Using this currently used Summarization technic the administrator will just need to monitor the processing of the summarization and to monitor the uplink of each webpage being updated.

2.2 LITERATURE REVIEW

Chia-Chen Yen and Jih-Shih Hsu [1,2009] analyzes with the technique which determines how the page rank can be improved with different web pages. Search engines encounter many new challenges while the amount of information on the web increases rapidly. Web documents are used for various purposes, and people who rely on search engines to retrieve the desired documents. This paper proposes an associated pagerank algorithm for search engines to present quality results by scoring the relevance of web documents. The modified pagerank algorithm increases the degree of relevance than the original one, and decreases the query time efforts of topic-sensitive pagerank. In this paper, we will discuss the accuracy and efficiency of different page ranking algorithms, and propose an associated pagerank to retain the relevance between web pages and to prevent the bias of link spam. If there are three pages {A, B, C}, and A links to B and C, and C links to A. The pagerank of A will be 0.43444227, and pagerank of B and C will be 0.334638, if damping factor is 0.85 and initial pagerank is 1. This associated pagerank can reduce the complexity of topic-sensitive pagerank and survey the relevance degree between web documents. This method of page rank provides.

Zha Peng¹, Xu Xiu², Zuo Ming³ [2,2011] Analyzes the traditional PageRank algorithm of the search engine deeply. According to its topic drift and putting emphasis on old web pages, an improved PageRank algorithm is proposed, which is based on the text content analysis and the introduction of time factor. The improved algorithm effectively meets the correlation requirements and authoritative problem. It can fit all kinds of web surfers needs. Experimental results show that the modified algorithm performs better than the standard PageRank. Further more, it can effectively reduce the topic drift and enhance the satisfaction of web surfers obviously. Thus the paper analyzes the PageRank algorithm and its development in detail. We use content analysis to avoid topic drift and introduce time factor to solve the problem of putting emphasis on old webs. The experiment has shown that it can improve recall ratio and precision ratio effectively and then attain a satisfying retrieving effect.

Zhou Cailan et.al. [3,2011] analyzes the PageRank algorithm, find out its disadvantages and put forward an improved PageRank algorithm. Based on the behaviour of user clicks, this algorithm on the one hand collects the clicks of result pages and the recently click time of result pages, on the other hand set the weight of clicks and click time as formula parameters of PageRank algorithm, and verify this algorithm does improve the precision

ratio of search engines through experiments. With the development of network technology and the appearance of search engines, people can always convenient to obtain mass information which are required on the internet. Google search engine checks the link structure of the entire network by using PageRank algorithm, and distribute the value of PageRank which indicate the importance of webpage to each page, then have the hypertext-matching analysis to determine the relativity between page and user's search keywords. After the consideration of the importance of whole page and the relativity with search keywords, finally bring the most relevant and most reliable search results to front. Based on the PageRank algorithm analysis, this paper proposes an improved PageRank algorithm by combining web link analysis with user feedback.

Wenpu Xing and Ali Ghorbani[5,2004] Web mining is used to extract information from users'past behavior. Web structure mining plays an important rolein this approach. Two commonly used algorithms in web structure mining are HITS and PageRank, which are used to rank the relevant pages. Both algorithms treat all links equally when distributing rank scores. Several algorithms have been developed to improve the performance of these methods. This paper introduces the WPR algorithm, an extension to the PageRank algorithm. WPR takes into account the importance of both the inlinks and the outlinks of the pages and distributes rank scores based on the popularity of the pages. Simulation studies using the Website of Saint Thomas University show that WPR is able to identify a larger number of relevant pages to a given query compared to standard PageRank. In the current version of WPR, only the inlinks and outlinks of the pages in the reference page list are used in the calculation of the rank scores. In our future study of this method, we would like to consider the possibility of calculating the rank scores by using more than one level of reference page list.

2.3 PROBLEM DESCRIPTION:

The existing system of page rank for document based search is distributing the page rank equally to all outgoing links which could be improved by measuring the relevance of a page with its parent page.

In the Summary generation using the sentence creation technic certain contents may have lost when a single page is alone checked out , but if compared with other other page results too then the user will get to the summay of his own.

2.4 HARDWARE & SOFTWARE REQUIREMENTS

Minimum Hardware Requirements:

- 256 MB RAM
- 80 GB hard disk
- Pentium 4 processor 2.5GHz

Software Requirements:

- Java development kit 1.7
- Mysql
- wamp
- jconnector
- Command line executer
- Web Browser

2.5FUNCTIONAL REQUIREMENTS

1 OBTAINING THE DATASET

The html documents that are to be processed are downloaded as html documents using the web crawlerbased on the links.

2 REMOVAL OF MARKUP TAGS

The first step in processing is the removal of the tags that are available in the html document. Here the html document is given as input and output is written into the text file.

3 REMOVING STOP WORDS

Removing stop words facilitates easy search by reducing the number of words that no way contribute to the meaning of the document. Where the text document after the above process is given as input and meaningless articles like a, an, the, etc., are removed and output is written back again to a text file.

4 STEMMING

Stemming is the process of reducing a word to its respective root word. This facilitates the searching process by reducing number of key words that are available in the document. Here the document that is available after removing stop words is given as input and the output will be another document containing only root words.

5 IDENTIFYING TERM FREQUENCIES

Now that the document is reduced to a considerable size, it is now simple to identify the words that occur maximum number of times. The document after stemming is given as input and the output will be the words that are available in the document and the number of times that particular word has occurred. This calculation is important for calculating the relevance of a page.

6 MEASURE RELEVANCE BETWEEN DOCUMENTS

After obtained the values of term frequencies, their relevance is measured with regards to the links that were available within the webpages. This measure will be based on the number of hyperlinks the page has.

7 RANK CALCULATION:

The word count that is available now is made use of now to calculate the rank. The words that are available more in number is taken as key words and their occurrence is calculated. This is applied to the rank formula and a rank value is obtained.

8 RANKING THE DOCUMENTS

After the relevance measure based on the value of relevance measure the document are ranked, more relevance highest rank strategies.

9 ALLOCATION SPACE AND FILES FOR SUMMARIZED DATA

Each webpage stored in the system are to be helpful in data summarization of itself .Hence temporary files are created for which it can be accessed by the database.

10 UPDATING THE DATABASE:

A database which has the document-id,url of page,key words belonging to that page ,rank fields and the summarized temporary files are created and updated as and when processing is completed.

11 GETTING USER QUERIES (or) KEY WORD

Now the input key word to be searched is got from the user and when the user clicks search button comparison occurs.

12 COMPARISON

User query is compared with available key words field stored in the database and the web pages are arranged in the descending order of the respective rank.

13 RETREIVING THE RESPECTIVE PAGE

The web pages along with their url's are listed in the user's browser.

14 DATABASE UPDATION

When new pages arrive or changes are made to a rank, databaseis updated along with respective updates in respective fields. When rank is updated the keywords will also be updated if necessary.

15 ADMINISTRATOR LOGIN

Procedure for login of an administrator to perform specific calculations and updates the database as and when necessary.Also to track for new webpages being updated .

16 SUMMARIZATION

After the user gets his search results the user select's his favorable links for the summary generation and clicks for the GenerateSummary button.

17 REFRESH RANK BASED ON CHANGES

As newpages keep arriving or changing the rank is changed accordingly by recalculating the formula at regular time intervals by the administrator.

18 UPDATE DATABASE

Change the rank field of the database based on new changes.

19 SPLITTING OUTPUT SCREEN

The output window is split into two windows to show the difference in which pages are ordered based on ranks obtained through links they have andthrough relevance measure.

20 MAPPING INPUT TO RESPECTIVE WINDOWS

The output is displayed where one side of the browser has pages that are listed based on rank obtained based on outgoing links and the other side contains pages listed on basis of rank obtained through relevance measure.

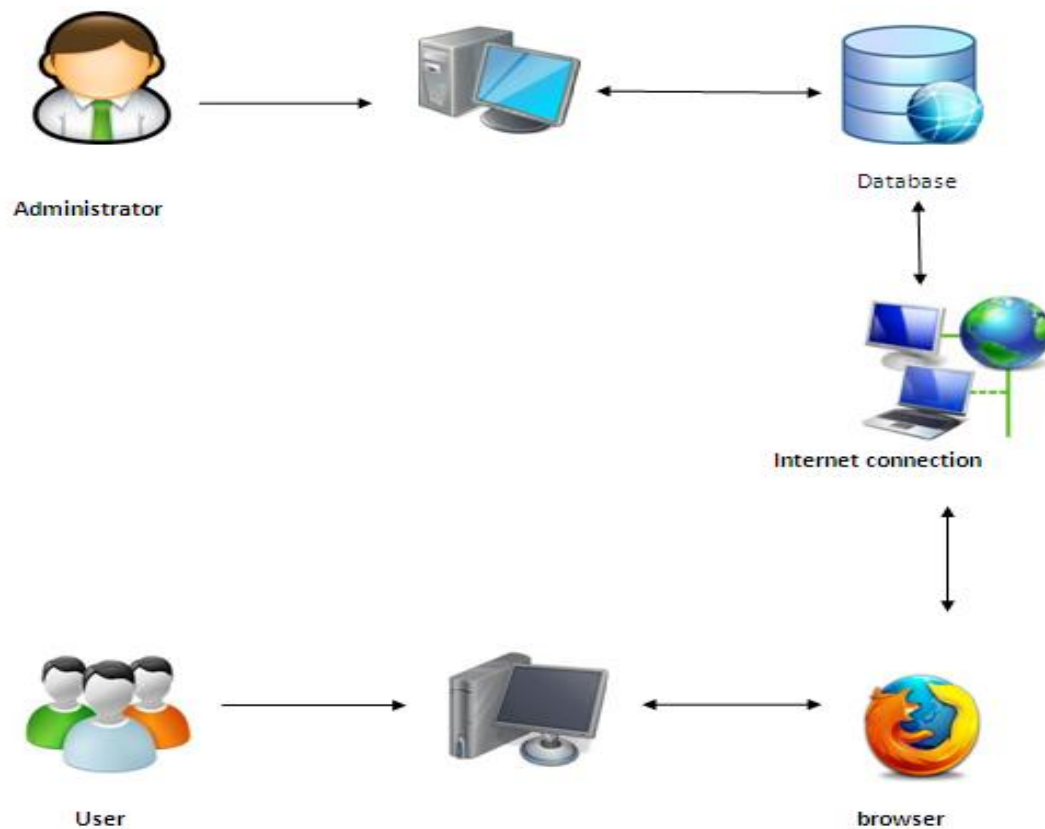
21 DISPLAYING SUMMARY ACCORDINGLY

The summary is displayed in the bulletin format and the user can see a line separating a set of points from next, which means it's the summary of the next link selected from the search result.

CHAPTER 3

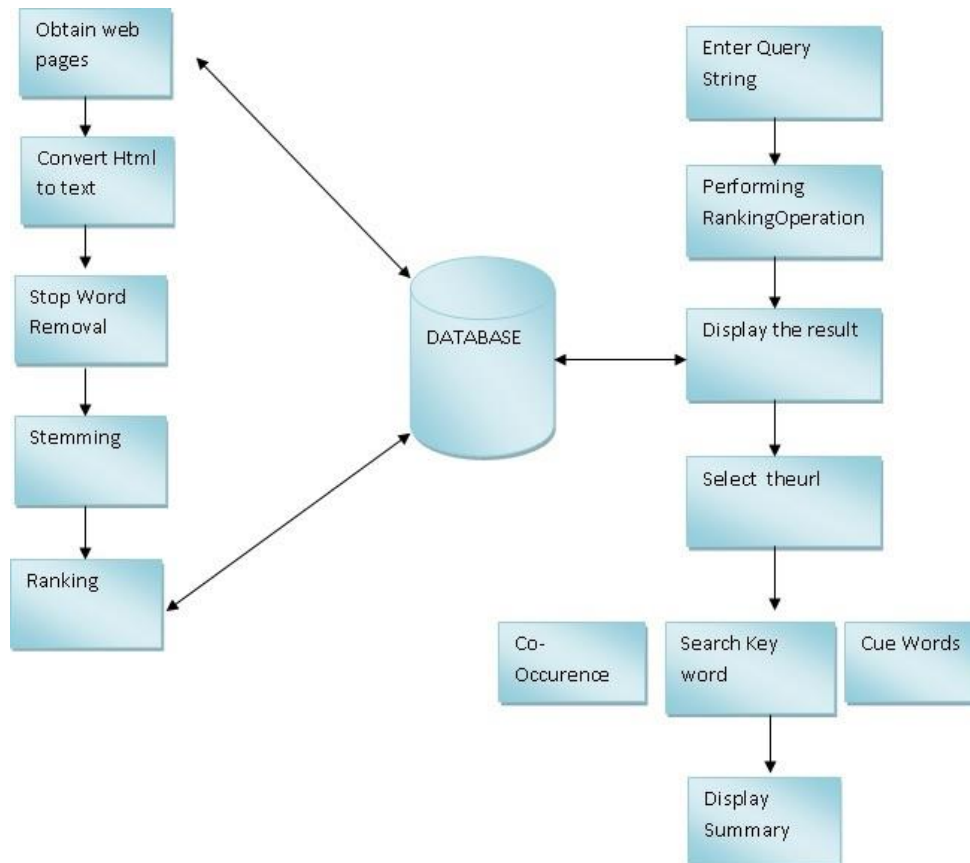
SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE



The pre-processing is done at the administrator side and the url and their respective rank is stored in the database. This database is updated after specific time interval by the administrator. The user on the other end can make use of the browser to access the search engine through internet. Based on the user query, the search engine displays the url in the descending order of rank on the browser.

3.2 OVERALL BLOCK DIAGRAM



The overall process starts with the pre-processing where the html pages are stored in the database. The html tags are removed from the web document and stored as text files. The removal of stop words is done next, where the words like a, an, the..etc., are removed. The next step is stemming where the words are stemmed to their respective root words. The next step is word count where the number of words that occur in the different document with the number of frequency is stored in the database. The normal rank is calculated based on the outbound links that are available for a page. The relevance based rank is calculated by finding the relevance between a page and the pages that are outbound links within that page and the rank along with the respective url is stored in the database. At the user side when the user types in a query the related url is displayed based

on the rank. This data is fetched from the database that contains the preprocessed data. The database will be updated periodically.

3.3 DATABASE DESIGN

Database Name: test

TABLE name: administrator

Field Name	Field Type
Username	Varchar2(30)
Password	Varchar2(30)

Table name: list

Doc id	Words	Frequency

Table name: url

Doc id	URL	Rank	Relevance	Content

Table name: relevance

Doc id	Out id	PageRelevance

3.4 ER-DIAGRAM

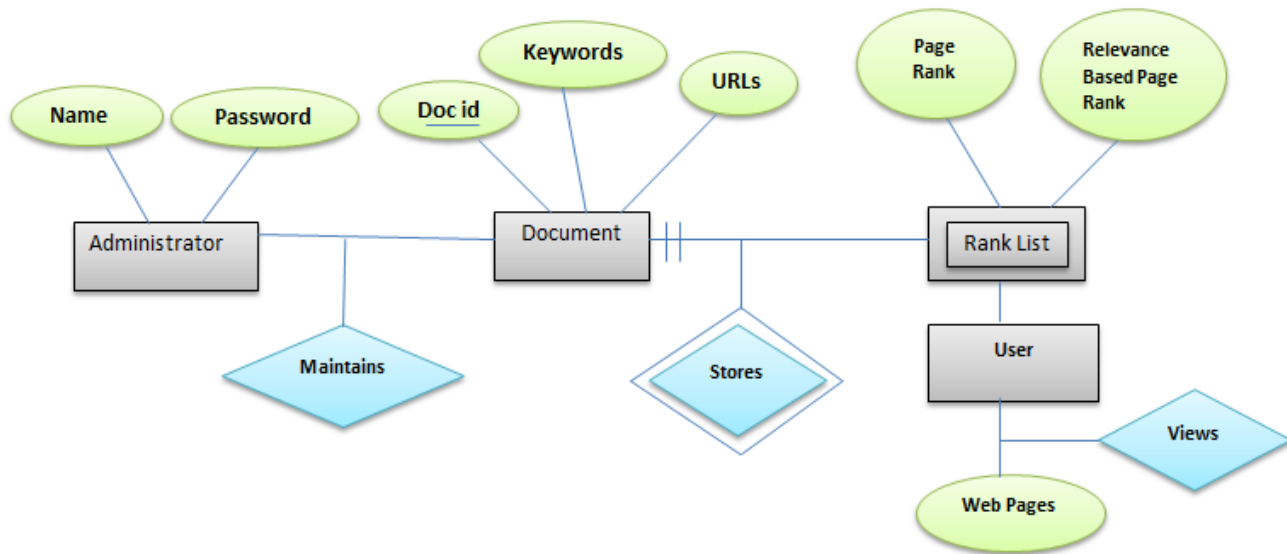


FIG-3.4 ENTITY – RELATIONSHIP DIAGRAM

This entity relationship diagram show the complete workflow of the project, the administrator is provided with a user name and password for authentication. The administrator does the preprocessing work and the data is stored in the database. This data can be made available for the user on request as a query.

3.5 CLASS DIAGRAM

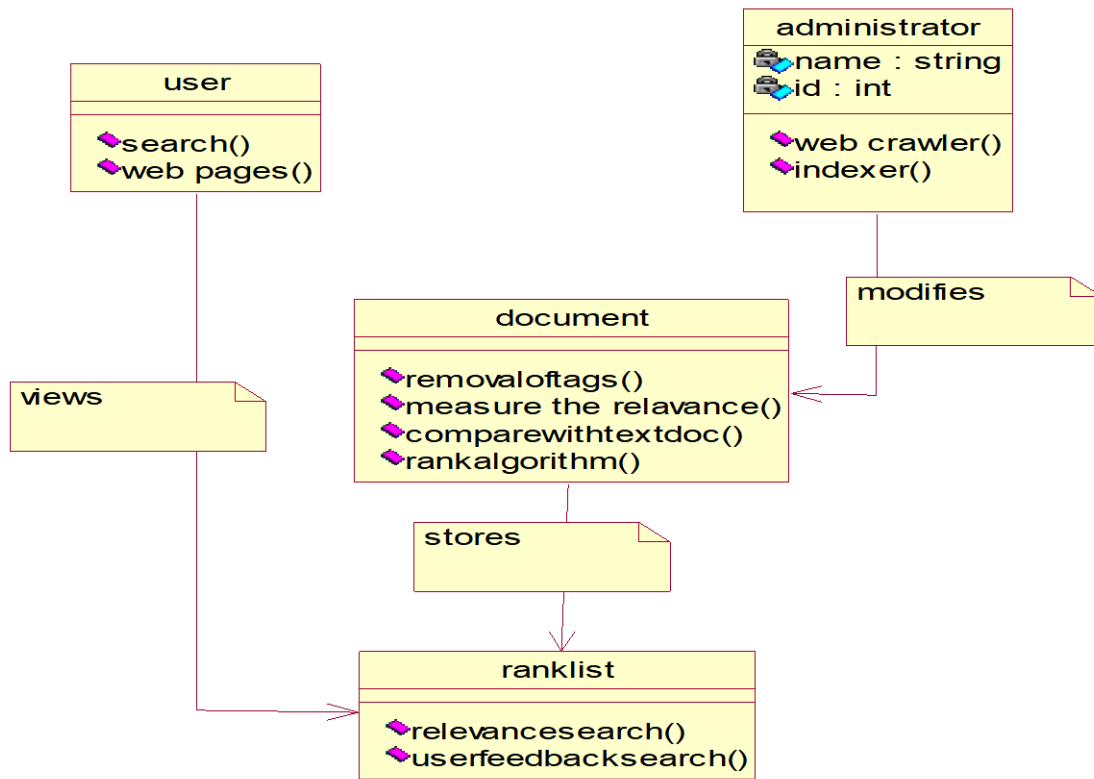


FIG-3.5 CLASS DIAGRAM

The class diagram represents the involvement of administrator and user actions. The administrator does the pre-processing and the concerned data is stored in the database. The user gets only the necessary data from it.

3.6 ACTIVITY DIAGRAM: please change the activity diagram

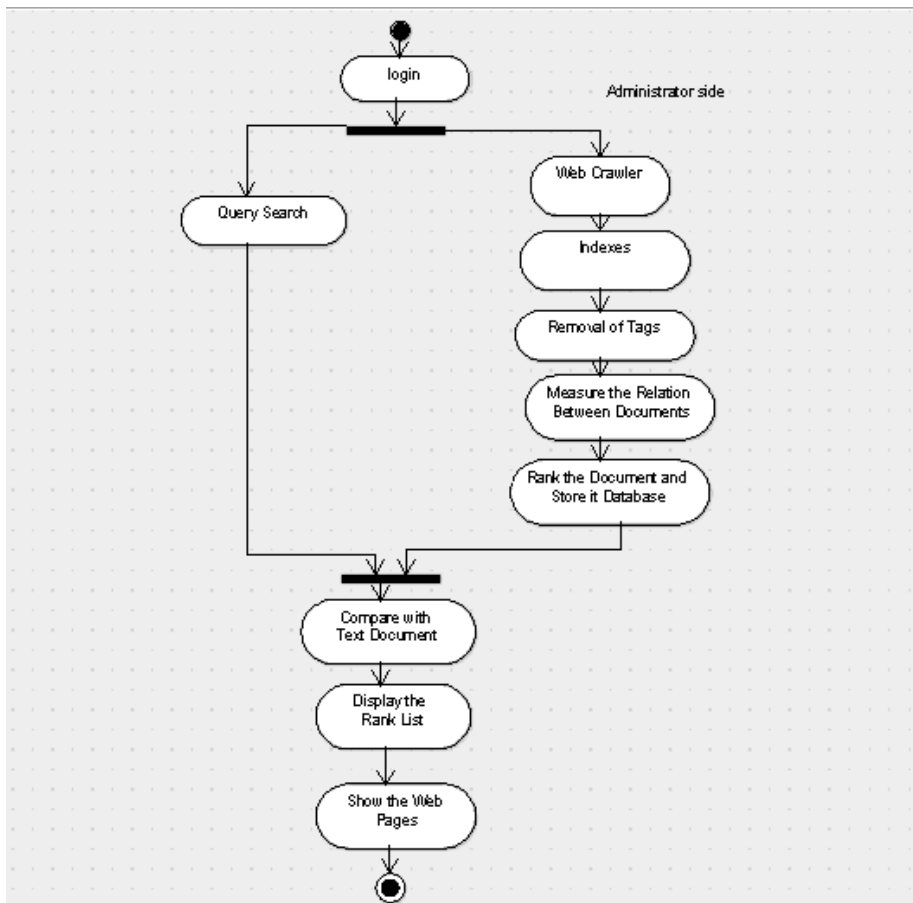


FIG-3.6 ACTIVITY DIAGRAM

The activity diagram represents the flow of activity between administrator and user. The administrator does the pre-processing and the concerned data is stored in the database. The user gets only the necessary data from it.

3.7 SEQUENCE DIAGRAM

ADMINISTRATOR 'S ACTIVITY:

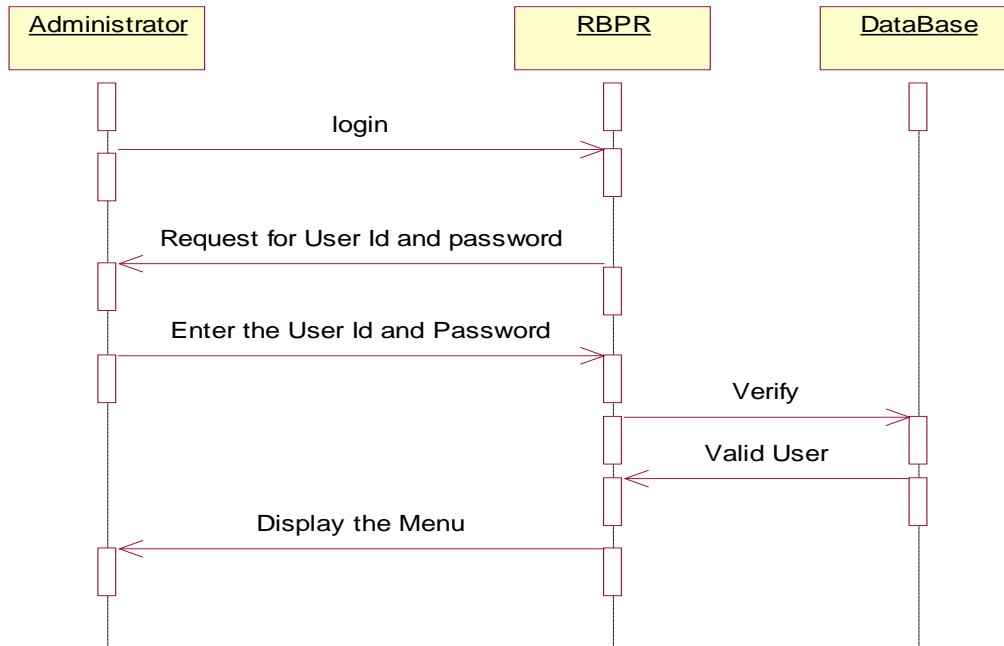


FIG-3.7.1 SEQUENCE DIAGRAM

The sequence diagram represents the sequence of activity between administrator and user. The administrator does the pre-processing and the concerned data is stored in the database. The user gets only the necessary data from it.

WEB CRAWLER ACTIVITY:

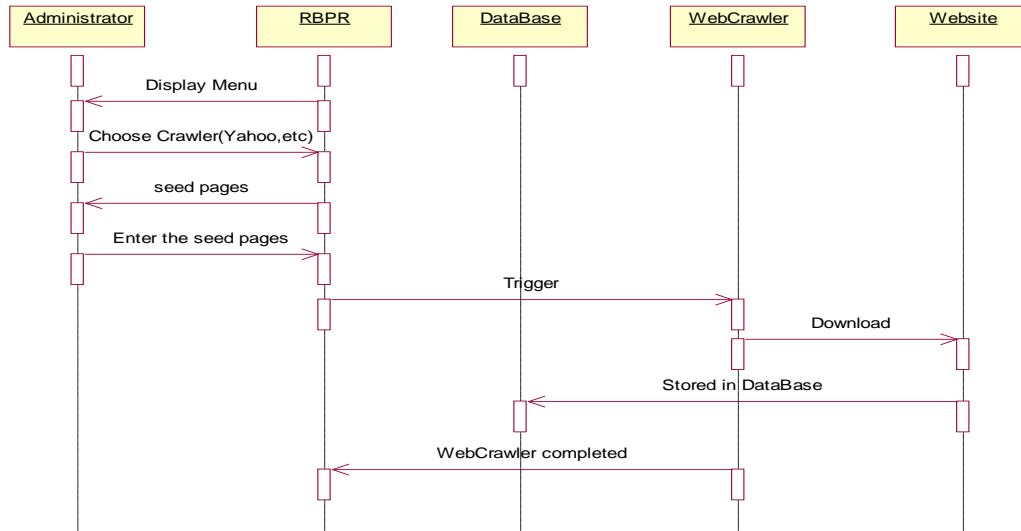


FIG-3.7.2 SEQUENCE DIAGRAM

The web crawler activity is shown above. A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner. The crawler downloads the html pages using their url's and stores them in the databases. The administrator instructs a web crawler to do this activity.

USER'S ACTIVITY :

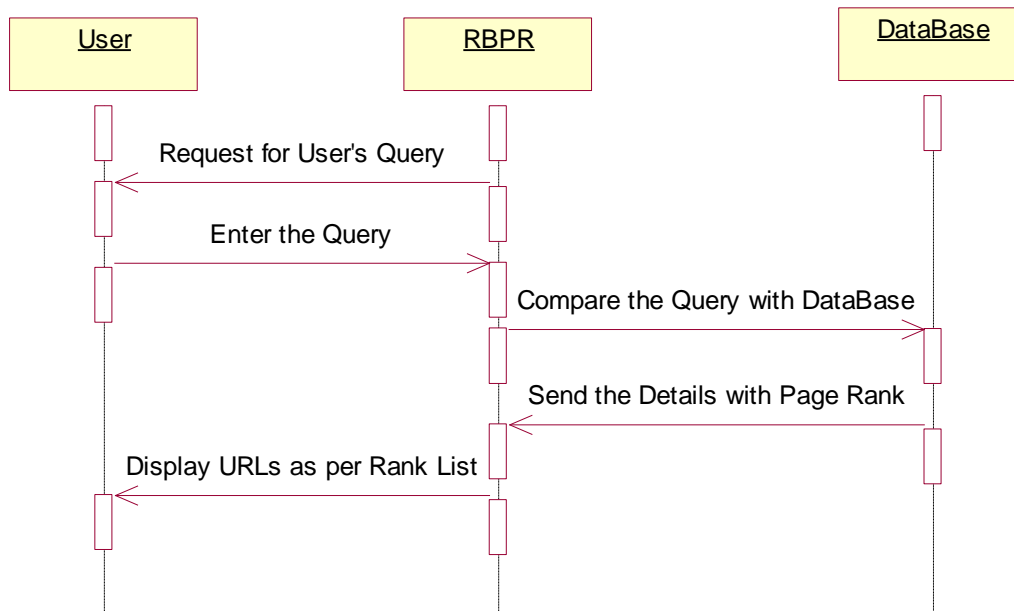


FIG-3.7.3 SEQUENCE DIAGRAM

The user enters a query in the search engine and the query is compared with the words in the database and it displays the url's as per the page rank.

3.8 USECASE DIAGRAM :

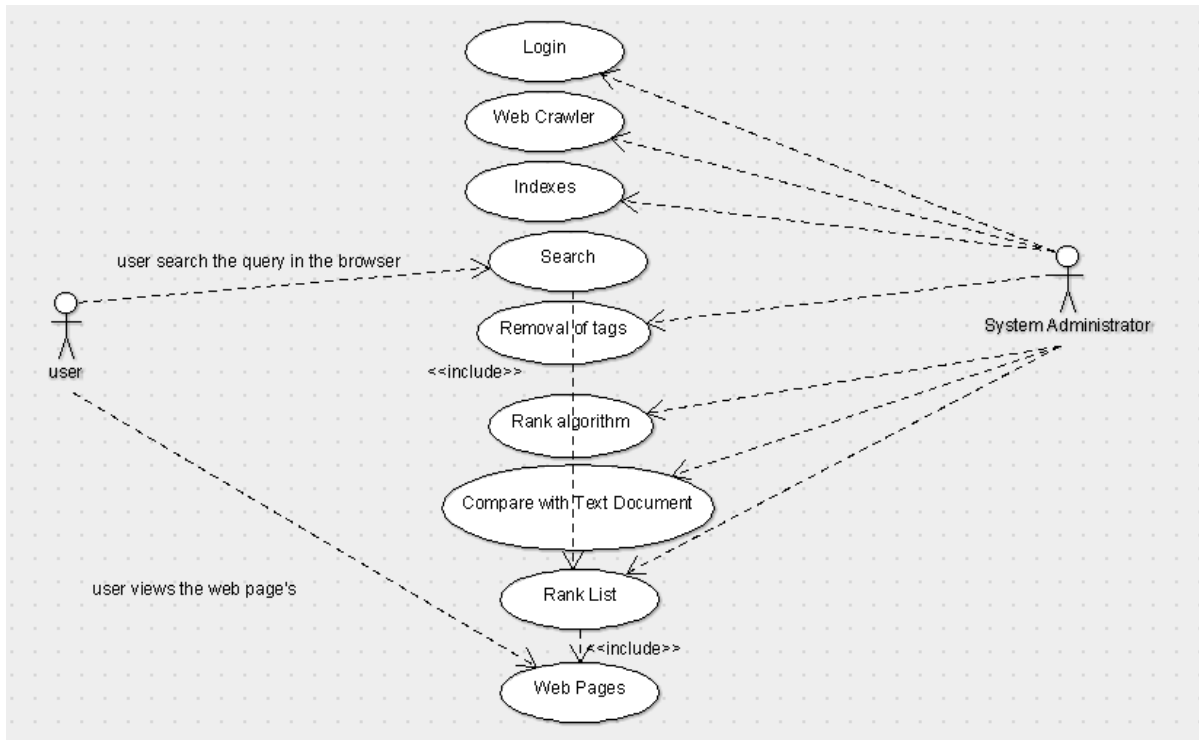


FIG-3.8 USECASE DIAGRAM

The usecase diagram shows two actors administrator and user. The administrator does the pre-processing and the concerned data is stored in the database. The user gets only the necessary data from it.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1MODULE DESCRIPTION:

4.1.1 INPUT MODULE:

WEB CRAWLER:

A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner or in an orderly fashion. This process is called Web crawling or spidering. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating html code. A Web crawler is one type of bot, or software agent. In general, it starts with a list of url's to visit, called the seeds. As the crawler visits these url's, it identifies all the hyperlinks in the page and adds them to the list of url's to visit. Url's from the frontier are recursively visited according to a set of policies. The crawler downloads the html pages using their url's and stores them in the databases.

4.1.2 PREPROCESSING:

HTML TO TEXT

The first step in preprocessing is the removal of the html tags that are available in the html documents. This conversion makes it simple for further

processing. The source folder could be located through the browsing option. The html document is given as input and the output is written into the text file.

REMOVING STOP WORDS

The next step will be removing the stop words from the available text document. The stop words consists of words such as (a, an, the, could, would etc). These words are removed because they do not contribute to the meaning of the document .

STEMMING

This process is where a word if possible is reduced to their respective root words. This process reduces time to go through every word available in the document ,where relative words could be grouped together. This also contributes to the exact value of word count available in the document.(eg: education→educate, scientists→scientists).

WORD COUNT

In relevance based rank calculation the occurrence of words in the document is taken as a factor. Hence this process finds the words that are available in the document and the occurrence of the individual words. In this way depending on the word that occurs the maximum number of times the rank may vary.

4.1.3 RANKING:

PAGE RANK

The page rank is calculated based on the outbound and inbound link.The page rank of a page is divided equally among the outbound links. If a page has rank 1 it is equally divided as 5 outbound links with 0.2 as rank for each page.

RELEVANCE BASED PAGE RANK

The relevance based page rank is calculated using wordnet API. The distance between words in the document is calculated and applied to the formula. In the formation of the most frequent set, we will filter the non topic related common words from WordNet. In the indexing time , the page similarities can be computed and associated pagerank is defined as

$$PR(V_i) = (1-d) + d \times \sum_{j \in In(V_i)} \frac{R_{ji} \times PR(V_j)}{\sum_{k \in Out(V_j)} R_{jk}}$$

R_{ji} is the relevance value of page j to page i .

If the denominator equals to zero, this means all outbound pages of pages can share the page rank value of page j . if we have the relevance value of page A for ten categories (0.3,0,0.1,0,0,0.2,0,0,0,0.05), and the relevance value of page B for ten categories (0.02,0.4,0,0,0,0.06,0,0,0,0), then we will have the relevance score of page A to page B.

The more relevant the inbound page is, the more page rank contribution the target page will get. In associated page rank, the page rank, the page rank value is not divided equally by the number of outbound links, but only relevant page can share the page rank value according to the relevance degree.

CHAPTER 5

TESTING

5.1 TEST CASES AND TEST RESULTS

White box testing : where the administrator gives the search query, it access all the internal mechanism of database and all the process of HTML TO TEXT convertor, stemmer, stop words, and word count is worked on and maximum words with frequency is accessed and displayed in order of maximum words.

Black box testing: The user gives the search key words in the search box and getting the results dynamically. Here the internal mechanism is considered. This works only on the user point of view to get the links of the search query.

CHAPTER 6

CODING

6.1 HTML TO TEXT CONVERTOR

```
package a;
import java.io.*;
import javax.swing.text.html.*;
import javax.swing.text.html.parser.*;
public class Html2Text extends HTMLToolkit.ParserCallback
{
    StringBuffer s;
    public Html2Text() {}

    public void parse(Reader in) throws IOException
    {
        s = new StringBuffer();
        ParserDelegator delegator = new ParserDelegator();
        // the third parameter is TRUE to ignore charset directive
        delegator.parse(in, this, Boolean.TRUE);
    }
    public void handleText(char[] text, int pos)
    {
        s.append(text);
    }
}
```

```

    }
    public String getText()
    {
        return s.toString();
    }
}

```

6.2 STOPWORDS

```

package a;
import java.util.*;
import java.io.*;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.util.Collections;
import java.util.Enumeraion;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Vector;

```

```

/**

```

- * Class that can test whether a given string is a stop word.
- * Lowercases all words before the test. <p/>
- * The format for reading and writing is one word per line, lines starting
- * with '#' are interpreted as comments and therefore skipped. <p/>
- * The default stopwords are based on

```

<a href="http://www.cs.cmu.edu/~mccallum/bow/rainbow/"
target="_blank">Rainbow</a>. <p/>
*
*/
public class Stopwords {

    /** The hash set containing the list of stopwords */
    protected HashSet m_Words = null;

    /** The default stopwords object (stoplist based on Rainbow) */
    protected static Stopwords m_Stopwords;

    static {
        if (m_Stopwords == null) {
            m_Stopwords = new Stopwords();
        }
    }

    /**
     * initializes the stopwords (based on
     <a href="http://www.cs.cmu.edu/~mccallum/bow/rainbow/"
     target="_blank">Rainbow</a>).
     */
    public Stopwords() {
        m_Words = new HashSet();

        //Stopwords list from Rainbow

```

add("a"); add("able"); add("about"); add("above"); add("according");
add("accordingly"); add("across"); add("actually"); add("after"); add("afterwards");
add("again"); add("against"); add("ag"); add("all"); add("allow"); add("allows");
add("almost"); add("alone"); add("along"); add("already"); add("also");
add("although"); add("always"); add("am"); add("among"); add("amongst");
add("an"); add("and"); add("another"); add("any"); add("anybody");
add("ar"); add("anyhow"); add("anyone"); add("anything"); add("anyway");
add("anyways"); add("anywhere"); add("apart"); add("appear"); add("appreciate");
add("appropriate"); add("are"); add("around"); add("as"); add("aside");
add("ask"); add("asking"); add("associated"); add("at"); add("available");
add("away"); add("awfully"); add("b"); add("be"); add("became");
add("because"); add("become"); add("becomes"); add("becoming"); add("been");
add("before"); add("beforehand"); add("behind"); add("being"); add("believe");
add("below"); add("beside"); add("besides"); add("best"); add("better");
add("between"); add("beyond"); add("both"); add("brief"); add("but");
add("by"); add("c"); add("came"); add("can"); add("cannot"); add("cant");
add("cause"); add("causes"); add("certain"); add("certainly"); add("changes");
add("clearly"); add("co"); add("com"); add("come"); add("comes");
add("concerning"); add("consequently"); add("consider"); add("considering");
add("contain"); add("containing"); add("contains"); add("corresponding");
add("could"); add("course"); add("currently"); add("d"); add("definitely");
add("described"); add("despite"); add("did"); add("different"); add("do");
add("does"); add("doing"); add("done"); add("down"); add("downwards");
add("during"); add("e"); add("each"); add("edu"); add("eg"); add("eight");
add("either"); add("else"); add("elsewhere"); add("enough"); add("entirely");
add("especially"); add("et"); add("etc"); add("even"); add("ever"); add("every");
add("everybody"); add("everyone"); add("everything"); add("everywhere");

add("ex"); add("exactly"); add("example"); add("except"); add("f"); add("far");
add("few"); add("fifth"); add("first"); add("five"); add("followed");
add("following"); add("follows"); add("for"); add("former"); add("formerly");
add("forth"); add("four"); add("from"); add("further"); add("furthermore");
add("g"); add("get"); add("gets"); add("getting"); add("given"); add("gives");
add("go"); add("goes"); add("going"); add("gone"); add("got"); add("gotten");
add("greetings"); add("h"); add("had"); add("happens"); add("hardly");
add("has"); add("have"); add("having"); add("he"); add("hello"); add("help");
add("hence"); add("her"); add("here"); add("hereafter"); add("hereby");
add("herein"); add("hereupon"); add("hers"); add("herself"); add("hi");
add("him"); add("himself"); add("his"); add("hither"); add("hopefully");
add("how"); add("howbeit"); add("however"); add("i"); add("ie"); add("if");
add("ignored"); add("immediate"); add("in"); add("inasmuch"); add("inc");
add("indeed"); add("indicate"); add("indicated"); add("indicates"); add("inner");
add("insofar"); add("instead"); add("into"); add("inward"); add("is"); add("it");
add("its"); add("itself"); add("j"); add("just"); add("k"); add("keep");
add("keeps"); add("kept"); add("know"); add("knows"); add("known");
add("l"); add("last"); add("lately"); add("later"); add("latter"); add("latterly");
add("least"); add("less"); add("lest"); add("let"); add("like"); add("liked");
add("likely"); add("little"); add("ll"); add("look"); add("looking"); add("looks");
add("ltd"); add("m"); add("mainly"); add("many"); add("may"); add("maybe");
add("me"); add("mean"); add("meanwhile"); add("merely"); add("might");
add("more"); add("moreover"); add("most"); add("mostly"); add("much");
add("must"); add("my"); add("myself"); add("n"); add("name"); add("namely");
add("nd"); add("near"); add("nearly"); add("necessary"); add("need");
add("needs"); add("neither"); add("never"); add("nevertheless"); add("new");
add("next"); add("nine"); add("no"); add("nobody"); add("non"); add("none");

add("noone"); add("nor"); add("normally"); add("not"); add("nothing");
 add("novel"); add("now");
 add("nowhere"); add("o"); add("obviously"); add("of"); add("off");
 add("often"); add("oh"); add("ok"); add("okay"); add("old"); add("on");
 add("once"); add("one"); add("ones"); add("only"); add("onto"); add("or");
 add("other"); add("others"); add("otherwise"); add("ought"); add("our");
 add("ours"); add("ourselves"); add("out"); add("outside"); add("over");
 add("overall"); add("own"); add("p"); add("particular"); add("particularly");
 add("per"); add("perhaps"); add("placed"); add("please"); add("plus");
 add("possible"); add("presumably"); add("probably"); add("provides"); add("q");
 add("que"); add("quite"); add("qv"); add("r"); add("rather"); add("rd");
 add("re"); add("really"); add("reasonably"); add("regarding"); add("regardless");
 add("regards"); add("relatively"); add("respectively"); add("right"); add("s");
 add("said"); add("same"); add("saw"); add("say"); add("saying"); add("says");
 add("second"); add("secondly"); add("see"); add("seeing"); add("seem");
 add("seemed"); add("seeming"); add("seems"); add("seen"); add("self");
 add("selves"); add("sensible"); add("sent"); add("serious"); add("seriously");
 add("seven"); add("several"); add("shall"); add("she"); add("should");
 add("since");add("six"); add("so"); add("some"); add("somebody"); add("at");
 add("somehow"); add("someone"); add("something"); add("sometime");
 add("sometimes"); add("somewhat"); add("somewhere"); add("soon"); add("sorry");
 add("specified"); add("specify"); add("specifying"); add("au"); add("still");
 add("sub"); add("such"); add("sup"); add("sure"); add("t"); add("take");
 add("taken"); add("tell"); add("tends"); add("th"); add("than"); add("thank");
 add("thanks"); add("thanx"); add("that"); add("thats"); add("the"); add("their");
 add("theirs"); add("them"); add("themselves"); add("then"); add("thence");
 add("there"); add("thereafter"); add("thereby"); add("therefore"); add("therein");

```

add("theres");    add("thereupon");    add("these");    add("they");    add("think");
add("third");    add("this");    add("thorough");    add("thoroughly");    add("those");
add("though");    add("three");    add("through");    add("throughout");    add("thru");
add("thus");    add("to");    add("together");    add("too");    add("took");    add("toward");
add("towards");    add("tried");    add("tries");    add("truly");    add("try");
add("trying");    add("twice");    add("two");    add("u");    add("un");    add("under");
add("unfortunately");    add("unless");    add("unlikely");    add("until");    add("unto");
add("up");    add("upon");    add("us");    add("use");    add("used");    add("useful");
add("uses");    add("using");    add("usually");    add("uucp");    add("v");
add("value");add("various");    add("ve");    add("very");    add("via");    add("viz");
add("vs");    add("w");    add("want");    add("wants");    add("was");    add("way");
add("we");    add("welcome");    add("well");    add("went");    add("were");    add("what");
add("whatever");    add("when");    add("whence");    add("whenever");    add("where");
add("whereafter");    add("whereas");add("whereby");add("wherein");    add("whereupon");
add("wherever");    add("whether");    add("which");    add("while");add("whither");
add("who");    add("whoever");    add("whole");    add("whom");    add("whose");
add("why");    add("will");    add("willing");    add("wish");    add("with");
add("within");    add("without");    add("wonder");    add("would");    add("would");
add("x");    add("y");    add("yes");    add("yet");    add("you");    add("your");
add("yours");

    add("yourself");    add("yourselves");    add("z");    add("zero");    }

/**
 * removes all stopwords
 */

public void clear() {
    m_Words.clear();
}

```

```

public void add(String word)
{
    if (word.trim().length() > 0)
        m_Words.add(word.trim().toLowerCase());
}

public boolean remove(String word)
{
    return m_Words.remove(word);
}

public boolean is(String word)
{
    return m_Words.contains(word.toLowerCase());
}

public Enumeration elements()
{
    Iterator  iter;
    Vector    list;
    iter = m_Words.iterator();
    list = new Vector();
    while (iter.hasNext())
        list.add(iter.next());
    // sort list
    Collections.sort(list);
return list.elements();
}

public String toString()

```



```

{
    Enumeration  enm;
    StringBuffer result;

    result = new StringBuffer();
    enm  = elements();
    while (enm.hasMoreElements())
    {
        result.append(enm.nextElement().toString());
        if (enm.hasMoreElements())
            result.append(",");
    }
    return result.toString();
}

public static boolean isStopword(String str)
{
    return m_Stopwords.is(str.toLowerCase());
}

public static void main(String[] args) throws Exception {

    String s=args[0];
    FileInputStream fstream = new FileInputStream(s);
    Scanner scanner=new Scanner(s);
    DataInputStream in = new DataInputStream(fstream);
    BufferedReader br = new BufferedReader(new InputStreamReader(in));

```

```

String strLine;
Writer output = null;
File file = new File("write.txt");
output = new BufferedWriter(new FileWriter(file));

while ((strLine = br.readLine()) != null)
{
    String[] arr=strLine.split(" ");
    int len=arr.length;
    // words to process?
    Vector words = new Vector();
    for (int i = 0; i < len; i++)
    {
        if (arr[i].trim().length() > 0)
            words.add(arr[i].trim());
    }

    Stopwords stopwords = new Stopwords();
    Enumeration enm = stopwords.elements();
    int i = 0;
    while (enm.hasMoreElements())
    {
        enm.nextElement();
        i++;
    }

    String abc;
    // check words for being a stopword
    if (words.size() > 0)
    {
        for ( i = 0; i < words.size(); i++)
            {if(stopwords.is(words.get(i).toString())!=true)
            {
                abc=words.get(i).toString();

```

```

        output.write(abc);
        output.write(" ");
    } } } } output.close(); } }

```

6.3 STEMMING

```

package a;

public class Stemmer
{
    private char[] b;
    private int i, /* offset into b */
               i_end, /* offset to end of stemmed word */
               j, k;
    private static final int INC = 50;
        /* unit of size whereby b is increased */
    public Stemmer()
    { b = new char[INC];
      i = 0;
      i_end = 0;
    }

    /**
     * Add a character to the word being stemmed. When you are finished
     * adding characters, you can call stem(void) to stem the word.
     */
    public void add(char ch)
    { if (i == b.length)
      { char[] new_b = new char[i+INC];

```

```

        for (int c = 0; c < i; c++) new_b[c] = b[c];
        b = new_b;
    }
    b[i++] = ch;
}

public void add(char[] w, int wLen)
{
    if (i+wLen >= b.length)
    {
        char[] new_b = new char[i+wLen+INC];
        for (int c = 0; c < i; c++) new_b[c] = b[c];
        b = new_b;
    }
    for (int c = 0; c < wLen; c++) b[i++] = w[c];
}

public String toString() { return new String(b,0,i_end); }
public int getResultLength() { return i_end; }
public char[] getResultBuffer() { return b; }

private final boolean cons(int i)
{
    switch (b[i])
    {
        case 'a': case 'e': case 'i': case 'o': case 'u': return false;
        case 'y': return (i==0) ? true : !cons(i-1);
        default: return true;
    }
}

private final int m()

```

```

{ int n = 0;
  int i = 0;
  while(true)
  { if (i > j) return n;
    if (! cons(i)) break; i++;
  }
  i++;
  while(true)
  { while(true)
    { if (i > j) return n;
      if (cons(i)) break;
      i++;
    }
    i++;
    n++;
    while(true)
    { if (i > j) return n;
      if (! cons(i)) break;
      i++;
    }
    i++;
  }
}

```

/* vowelinstem() is true <=> 0,...j contains a vowel */

private final boolean vowelinstem()

```

{ int i; for (i = 0; i <= j; i++) if (! cons(i)) return true;
  return false;
}

```

/* doublec(j) is true <=> j,(j-1) contain a double consonant. */

```

private final boolean doublec(int j)

```

```

{ if (j < 1) return false;
  if (b[j] != b[j-1]) return false;
  return cons(j);
}

```

/* cvc(i) is true <=> i-2,i-1,i has the form consonant - vowel - consonant
and also if the second c is not w,x or y. this is used when trying to
restore an e at the end of a short word. e.g.

cav(e), lov(e), hop(e), crim(e), but
snow, box, tray.

*/

```

private final boolean cvc(int i)

```

```

{ if (i < 2 || !cons(i) || cons(i-1) || !cons(i-2)) return false;
  { int ch = b[i];
    if (ch == 'w' || ch == 'x' || ch == 'y') return false;
  }
  return true;
}

```

```

private final boolean ends(String s)
{
    int l = s.length();
    int o = k-l+1;
    if (o < 0) return false;
    for (int i = 0; i < l; i++) if (b[o+i] != s.charAt(i)) return false;
    j = k-l;
    return true;
}

/* setto(s) sets (j+1)...k to the characters in the string s, readjusting
   k. */

private final void setto(String s)
{
    int l = s.length();
    int o = j+1;
    for (int i = 0; i < l; i++) b[o+i] = s.charAt(i);
    k = j+l;
}

private final void r(String s) { if (m() > 0) setto(s); }

private final void step1()
{
    if (b[k] == 's')
    {
        if (ends("sses")) k -= 2; else
            if (ends("ies")) setto("i"); else
                if (b[k-1] != 's') k--;
    }

    if (ends("eed")) { if (m() > 0) k--; } else
    if ((ends("ed") || ends("ing")) && vowelinstem())
    {
        k = j;
    }
}

```

```

    if (ends("at")) setto("ate"); else
    if (ends("bl")) setto("ble"); else
    if (ends("iz")) setto("ize"); else
    if (doublec(k))
    { k--;
      { int ch = b[k];
        if (ch == 'l' || ch == 's' || ch == 'z') k++;
      }
    }
    else if (m() == 1 && cvc(k)) setto("e");
  }
}
/* step2() turns terminal y to i when there is another vowel in the stem. */

```

```

private final void step2() { if (ends("y") && vowelinstem()) b[k] = 'i'; }

```

```

/* step3() maps double suffices to single ones. so -ization ( = -ize plus
-ation) maps to -ize etc. note that the string before the suffix must give
m() > 0. */

```

```

private final void step3() { if (k == 0) return; /* For Bug 1 */ switch (b[k-1])
{
  case 'a': if (ends("ational")) { r("ate"); break; }
             if (ends("tional")) { r("tion"); break; }
             break;
  case 'c': if (ends("enci")) { r("ence"); break; }
             if (ends("anci")) { r("ance"); break; }
}

```



```

        break;
    case 'e': if (ends("izer")) { r("ize"); break; }
        break;
    case 'l': if (ends("bli")) { r("ble"); break; }
        if (ends("alli")) { r("al"); break; }
        if (ends("entli")) { r("ent"); break; }
        if (ends("eli")) { r("e"); break; }
        if (ends("ousli")) { r("ous"); break; }
        break;
    case 'o': if (ends("ization")) { r("ize"); break; }
        if (ends("ation")) { r("ate"); break; }
        if (ends("ator")) { r("ate"); break; }
        break;
    case 's': if (ends("alism")) { r("al"); break; }
        if (ends("iveness")) { r("ive"); break; }
        if (ends("fulness")) { r("ful"); break; }
        if (ends("ousness")) { r("ous"); break; }
        break;
    case 't': if (ends("aliti")) { r("al"); break; }
        if (ends("iviti")) { r("ive"); break; }
        if (ends("biliti")) { r("ble"); break; }
        break;
    case 'g': if (ends("logi")) { r("log"); break; }
} }
private final void step4() { switch (b[k])
{
    case 'e': if (ends("icate")) { r("ic"); break; }

```

```

        if (ends("ative")) { r(""); break; }
        if (ends("alize")) { r("al"); break; }
        break;
    case 'i': if (ends("iciti")) { r("ic"); break; }
        break;
    case 'l': if (ends("ical")) { r("ic"); break; }
        if (ends("ful")) { r(""); break; }
        break;
    case 's': if (ends("ness")) { r(""); break; }
        break;
} }

```

```
private final void step5()
```

```

{  if (k == 0) return;
    switch (b[k-1])
    {  case 'a': if (ends("al")) break; return;
        case 'c': if (ends("ance")) break;
            if (ends("ence")) break; return;
        case 'e': if (ends("er")) break; return;
        case 'i': if (ends("ic")) break; return;
        case 'l': if (ends("able")) break;
            if (ends("ible")) break; return;
        case 'n': if (ends("ant")) break;
            if (ends("ement")) break;
            if (ends("ment")) break;
            /* element etc. not stripped before the m */
            if (ends("ent")) break; return;
    }
}

```

```

    case 'o': if (ends("ion") && j >= 0 && (b[j] == 's' || b[j] == 't')) break;
               /* j >= 0 fixes Bug 2 */
               if (ends("ou")) break; return;
               /* takes care of -ous */
    case 's': if (ends("ism")) break; return;
    case 't': if (ends("ate")) break;
               if (ends("iti")) break; return;
    case 'u': if (ends("ous")) break; return;
    case 'v': if (ends("ive")) break; return;
    case 'z': if (ends("ize")) break; return;
    default: return;
}
if (m() > 1) k = j;
}

```

/* step6() removes a final -e if m() > 1. */

```

private final void step6()
{ j = k;
  if (b[k] == 'e')
  { int a = m();
    if (a > 1 || a == 1 && !cvc(k-1)) k--;
  }
  if (b[k] == 'l' && doublec(k) && m() > 1) k--;
}

```

/** Stem the word placed into the Stemmer buffer through calls to add().

* Returns true if the stemming process resulted in a word different
* from the input. You can retrieve the result with
* getResultLength()/getResultBuffer() or toString().

*/

```
public void stem()
```

```
{ k = i - 1;
```

```
  if (k > 1) { step1(); step2(); step3(); step4(); step5(); step6(); }
```

```
  i_end = k+1; i = 0;
```

```
  } }
```

6.4 WORDCOUNT:

```
import java.io.*;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
```

```
import java.sql.Statement;
```

```
import java.util.*;
```

```
public class wordcount {
```

```
    static Connection conn;
```

```
    static TextReader in;
```

```
    static PrintWriter out;
```

```
    static class WordData {
```

```
        String word;
```

```
        int count;
```

```

WordData(String w) {
    // Constructor for creating a WordData object when
    // we encounter a new word.
    word = w;
    count = 1; // The initial value of count is 1.
}
} // end class WordData

static class CountCompare implements Comparator {
    // A comparator for comparing objects of type WordData
    // according to their counts. This is used for
    // sorting the list of words by frequency.
    public int compare(Object obj1, Object obj2) {
        WordData data1 = (WordData) obj1;
        WordData data2 = (WordData) obj2;
        return data2.count - data1.count;
        // The return value is positive if data2.count > data1.count.
        // I.E., data1 comes after data2 in the ordering if there
        // were more occurrences of data2.word than of data1.word.
        // The words are sorted according to decreasing counts.
    }
} // end class CountCompare

// public static void main(String[] args){
wordcount(String s1,String s2) // constructor
{

```

```

        String args[]={s1,s2};
        openFiles(args); // Opens input and output streams, using file

        TreeMap words; // TreeMap in which keys are words and values
        words = new TreeMap();

        readWords(in, words);
        // Reads words from the input stream and
        // stores data about them in words.

        List wordsByCount;
        wordsByCount = new ArrayList(words.values());
        Collections.sort(wordsByCount, new CountCompare());
        System.out.println("Words found in the file named \"" + args[0] + "\".\n");
        System.out.println("The number of times that the word occurred in the");
        System.out.println("file is given in parentheses after the word.\n\n");
        System.out.println("The words from the file in alphabetical order:\n");
        System.out.println("\n\nThe words in order of frequency:\n");
        printWords(wordsByCount); // Prints words by frequency count.
        System.out.println(words.size() + " distinct words were found.");

    } // end constructor .....main()

    static void printWords(Collection wordData) {
        //PrintWriter outputStream, Collection wordData) {

        // wordData must contain objects of type WordData. The words

```

```

        // and counts in these objects are written to the output stream.
        Iterator iter = wordData.iterator();
        while (iter.hasNext()) {
            WordData data = (WordData) iter.next();
            System.out.println("  " + data.word + " " + data.count );
        }
    } // end printWords()

```

```

static void openFiles(String[] args) {
    if (args.length != 2) {
        System.out.println("Error: Please specify file names on command line.");
        System.exit(1);
    }
    try {
        in = new TextReader(new FileReader(args[0]));
    }
    catch (IOException e) {
        System.out.println("Error: Can't open input file " + args[0]);
        System.exit(1);
    }
    try {
        out = new PrintWriter(new FileWriter(args[1]));
    }
    catch (IOException e) {
        System.out.println("Error: Can't open output file " + args[1]);
        System.exit(1);
    }
}

```

```

    }
} // end openFiles()

static void readWords(TextReader inStream, Map words) {
    try {
        while (true) {
            while (! inStream.eof() && ! Character.isLetter(inStream.peek()))
                inStream.getAnyChar(); // Skip past non-letters.
            if (inStream.eof())
                break; // Exit because there is no more data to read.
            String word = inStream.getAlpha(); // Read one word from stream.
            word = word.toLowerCase();
            WordData data = (WordData)words.get(word);
            // Check whether the word is already in the Map. If not,
            // the value of data will be null. If it is not null, then
            // it is a WordData object containing the word and the
            // number of times we have encountered it so far.
            if (data == null) {
                // We have not encountered word before. Add it to
                // the map. The initial frequency count is
                // automatically set to 1 by the WordData constructor.
                words.put(word, new WordData(word) );
            }
            else {
                // The word has already been encountered, and data is
                // the WordData object that holds data about the word.
                // Add 1 to the frequency count in the WordData object.

```



```

        data.count = data.count + 1;
    }
}
}
catch (TextReader.Error e) {
    System.out.println("An error occurred while reading the data.");
    System.out.println(e.toString());
    System.exit(1);
}

} // end readWords()
} // end of class wordcount

```

class TextReader extends FilterReader

```

{           // ***** Exception Classes
*****

```

public static class Error extends RuntimeException

{ /* Represents any exception that occurs in the AsciiInputStream Class. In fact, only general Exceptions are translated directly into TextReader.Error. Other exceptions throw objects belonging to one of the following subclasses of TextReader.Error. */

Error(String errorMessage)

```

{ super(errorMessage);
}
}

```

public static class FormatError extends Error

```

{ // Illegal data: an illegal number or an illegal boolean value
FormatError(String errorMessage)
{ super(errorMessage); }
}

public static class EOFError extends Error
{ // attempt to read past end of stream
EOFError(String errorMessage) { super(errorMessage); }
}

public TextReader(BufferedReader s) { super(s); }
public TextReader(Reader s) { super(new BufferedReader(s)); }
public TextReader(InputStream s) { super(new BufferedReader(new
InputStreamReader(s))); }

public void IOCheck(boolean throwExceptions)
{ // call IOCheck(false) to turn
throwExceptionOnError = throwExceptions;

// off exceptions, then check
}

// for errors by calling checkError()
public boolean error()
{
// returns true if the most recent input operation on
return errorFlag;
// this TextReader produced an error. An error
}

```

```

// message can be retrieved by calling getErrorMessage()
public String getErrorMessage() {
// if the most recent operation on
return errorFlag ? errorMessage : null;
// this TextReader produced an error, this
} //*****InputMethods *****
public char peek()
{ errorFlag = false; return lookChar();
}
public char getAnyChar() { errorFlag = false; return readChar();
}
public char getChar()
{ errorFlag = false; skipWhiteSpace(); return readChar();
}
public byte getByte() { errorFlag = false; return (byte)readInteger(-128L,127L);

}
public short getShort() { errorFlag = false; return (short)readInteger(-32768L,32767L);
}
public int getInt()
{ errorFlag = false;
return (int)readInteger((long)Integer.MIN_VALUE, (long)Integer.MAX_VALUE);
}
public long getLong() { errorFlag = false; return readInteger(Long.MIN_VALUE,
Long.MAX_VALUE);
}
public float getFloat() { errorFlag = false; return readFloat();
}

```

```

}
public double getDouble() { errorFlag = false; return readDouble();
}
public boolean getBoolean() { errorFlag = false; return readBoolean();
}
public String getWord() { errorFlag = false; return readWord();
}
public String getAlpha() { errorFlag = false; return readAlpha();
}
public String getln() { errorFlag = false; return readLine();
}
public char getlnChar() { char x=getChar(); dropLine(); return x;
}
public byte getlnByte() { byte x=getByte(); dropLine(); return x;
}
public short getlnShort() { short x=getShort(); dropLine(); return x;
}
public int getlnInt() { int x=getInt(); dropLine(); return x;
}
public long getlnLong() { long x=getLong(); dropLine(); return x;
}
public float getlnFloat() { float x=getFloat(); dropLine(); return x;
}
public double getlnDouble() { double x=getDouble(); dropLine(); return x;
}
public boolean getlnBoolean() { boolean x=getBoolean(); dropLine(); return x;
}

```

```

public String getlnWord() { String x=getWord(); dropLine(); return x;
}
public String getlnAlpha() { String x=getAlpha(); dropLine(); return x;
}
public boolean eoln() {
char ch = lookChar();
while (!EOF && !errorFlag && (ch == ' ' || ch == '\t'))
{ readChar(); ch = lookChar();
}
return (ch == '\n' || EOF);
}
public boolean eof()
{ char ch = lookChar();
while (!EOF && !errorFlag && (ch == ' ' || ch == '\n' || ch == '\t'))
{ readChar(); ch = lookChar(); }
return EOF;
}
public void skipWhiteSpace()
{ char ch = lookChar();
while (!errorFlag && (ch == ' ' || ch == '\n' || ch == '\t'))
{ readChar(); ch = lookChar();
}
}
public void skipNonLetters()
{ char ch = lookChar();
while (!errorFlag && !EOF && !Character.isLetter(ch))
{ readChar(); ch = lookChar();
}
}

```

```

}
}
public void close()
{ errorFlag = false;
try { in.close(); }
catch (IOException e) { errorFlag = true; errorMessage = e.toString(); } }
// *****private implementation stuff *****

private int lookAhead = -1; // one-character buffer; -1 indicates the buffer is empty
private boolean errorFlag = false; // set to true if most recent operation produced an error
private String errorMessage = ""; // error message for the most recent error
private boolean EOF = false; // has the end-of-stream been encountered?
private boolean throwExceptionOnError = true; // determines which type of error-
handling is used

// Three utility routines for throwing exceptions.
private void doError(String message)
{ errorFlag = true; errorMessage = message;
if (throwExceptionOnError) throw new Error(message);
}
private void doFormatError(String message)
{ errorFlag = true;
errorMessage = message;
if (throwExceptionOnError) throw new FormatError(message);
}
private void doEOFError(String message)
{ errorFlag = true;
errorMessage = message;
if (throwExceptionOnError) throw new FormatError(message);
}

```

```

}
// The remaining methods are basic methods for reading the various types of
// data from the stream
private char readChar()
{ char ch = lookChar();
if (EOF) doEOFError("Attempt to read past end-of-data in input stream.");
lookAhead = -1;
return ch;
}
private boolean possibleLineFeedPending = false; // for dealing with \r\n pairs
private char lookChar()
{ if (lookAhead != -1)
{ if (lookAhead == '\r') return '\n';
else return (char)lookAhead;
} if (EOF) return '\0';
try { int n = in.read();
if (n == '\n' && possibleLineFeedPending)
{
// ignore \n of \r\n pair
n = in.read();
}
possibleLineFeedPending = (n == '\r');
lookAhead = n;
if (lookAhead == -1)
{ EOF = true; return '\0';
}
} catch (IOException e)

```

```

    { doError(e.getMessage()); }
    if (lookAhead == '\r')
// represent all eoln's with \n
    lookAhead = '\n';
    return (char)lookAhead;
}

private void dropLine()
{ while (!errorFlag)
  { if (lookChar() == '\0') return;
    if (readChar() == '\n') return; }
}

private String readLine()
{ StringBuffer s = new StringBuffer(100);
  char ch = readChar();
  while (!errorFlag && ch != '\n')
  { s.append(ch); ch = lookChar();
    if (ch == '\0') break; ch = readChar();
  }
  return s.toString();
}

private String readWord()
{ skipWhiteSpace();
  if (errorFlag) return null;
  StringBuffer s = new StringBuffer(50);
  char ch = lookChar();
  if (EOF) { doEOFError("Attempt to read past end-of-data");
    return null;
  }

```



```

}
while (!errorFlag && !EOF && ch != '\n' && ch != ' ' && ch != '\t')
{
    s.append(readChar());
ch = lookChar(); }
return s.toString();
}

private String readAlpha()
{ skipNonLetters();
if (errorFlag) return null;
StringBuffer s = new StringBuffer(50);
char ch = lookChar();
if (EOF) { doEOFError("Attempt to read past end-of-data"); return null;
}
while (!errorFlag && Character.isLetter(ch))
{ s.append(readChar());
ch = lookChar(); }
return s.toString();
}

public float readFloat()
{ double d = readDouble();
if (errorFlag) return Float.NaN;
if (Math.abs(d) > Float.MAX_VALUE) doFormatError("Input number outside of legal
range for values of type float");
return (float)d;
}

public double readDouble()
{ double x = Double.NaN;

```

```

StringBuffer s=new StringBuffer(50);
skipWhiteSpace();
char ch = lookChar();
if (ch == '-' || ch == '+') { s.append(readChar()); skipWhiteSpace(); ch = lookChar(); }
if ( (ch < '0' || ch > '9') && (ch != '.') )
{ if (EOF) doEOFError("Expecting a floating-point number and found end-of-data");
else doFormatError("Expecting a floating-point number and found \"\" + ch + "\"\" );
return Double.NaN;
}
boolean digits = false;
while (ch >= '0' && ch <= '9')
{ s.append(readChar());
ch = lookChar();
digits = true;
}
if (ch == '.')
{ s.append(readChar()); ch = lookChar();
while (ch >= '0' && ch <= '9')
{ s.append(readChar());
ch = lookChar();
digits = true;
}
}
if (!digits)
{ doFormatError("No digits found in floating-point input.");
return Double.NaN; }
if (ch == 'E' || ch == 'e')

```

```

    { s.append(readChar()); ch = lookChar();
    if (ch == '-' || ch == '+')
    { s.append(readChar()); ch = lookChar(); }
    if ( (ch < '0' || ch > '9') && (ch != '.') )
    { if (EOF) doEOFError("Expecting exponent for a floating-point number and found end-
of-data");
    else doFormatError("Expecting exponent for a floating-point number and found \"\" + ch
+ \"\"); return Double.NaN;
    }
    while (ch >= '0' && ch <= '9')
    { s.append(readChar());
    ch = lookChar();
    }
    }
    String str = s.toString();
    Double d;
    try { d = new Double(str);
    x = d.doubleValue(); }
    catch (NumberFormatException e)
    { x = Double.NaN; }
    if (Double.isNaN(x) || Double.isInfinite(x))
    { doFormatError("Illegal floating point number");
    return Double.NaN;
    }
    return x;
}

```

```

public boolean readBoolean()
{
    boolean ans = false;
    String s = getWord();
    if (errorFlag) return false;
    if ( s.equalsIgnoreCase("true") || s.equalsIgnoreCase("t") || s.equalsIgnoreCase("yes") ||
        s.equalsIgnoreCase("y") || s.equals("1") )
    {
        ans = true;
    }
    else if ( s.equalsIgnoreCase("false") || s.equalsIgnoreCase("f") ||
        s.equalsIgnoreCase("no") || s.equalsIgnoreCase("n") || s.equals("0") )
    {
        ans = false;
    }
    else doFormatError("Illegal input for value of type boolean: \"" + s + "\""); return ans;
}

private long readInteger(long min, long max)
{
    // read long integer, limited to specified range
    skipWhiteSpace();
    if (errorFlag) return 0;
    char sign = '+';
    if (lookChar() == '-' || lookChar() == '+')
    {
        sign = getChar(); skipWhiteSpace();
    }
    long n = 0;
    char ch = lookChar();
    if (ch < '0' || ch > '9')
    {
        if (EOF) doEOFError("Expecting an integer and found end-of-data");
        else doFormatError("Expecting an integer and found \"" + ch + "\""); return 0;
    }
}

```

```

while (!errorFlag && ch >= '0' && ch <= '9')
{ readChar(); n = 10*n + (int)ch - (int)'0';
if (n < 0) { doFormatError("Integer value outside of legal range");
return 0; }
ch = lookChar();
}
if (sign == '-') n = -n; if (n < min || n > max)
{ doFormatError("Integer value outside of legal range"); return 0; }
return n; }

// Override the read() methods from FilterReader so that they will work if a
// TextReader is wrapped inside another file. (This is necessary to take
// lookAhead into account.)
public int read() throws IOException
{ if (lookAhead == -1) return super.read();
else { int x = lookAhead;
lookAhead = -1; return x;
}
}

public int read(char[] buffer, int offset, int count) throws IOException
{ if (lookAhead == -1 || count <= 0)
return super.read(buffer,offset,count);
else if (count == 1) { buffer[offset] = (char)lookAhead; lookAhead = -1; return 1; }
else
{ buffer[offset] = (char)lookAhead; lookAhead = -1; int ct =
super.read(buffer,offset+1,count-1); return ct + 1; }
} } // end of class TextReader

```

6.5 MYFRAME:

```
import a.*;
import java.sql.Statement;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.util.Enumeration;
import java.util.Vector;
import examples.*;
public class myFrame extends JFrame
{
    final JTextField text=new JTextField(25);
    final JTextArea textArea = new JTextArea(15,22); JFileChooser chooser;
    public myFrame()
    {

        Toolkit kit = Toolkit.getDefaultToolkit();
        Dimension screenSize = kit.getScreenSize();
```

```

int Height = screenSize.height;
int Width = screenSize.width;
setTitle("ADMINISTRATOR FUNCTIONS");
setLocation(Height/3,Width/5);
setVisible(true);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//creating buttons and tool tip

JButton b=new JButton("HTML TO TEXT");
JButton b1=new JButton("REMOVE STOP WORDS");
JButton b2=new JButton("STEMMING");
JButton b3=new JButton("WORD COUNT");
JButton b4=new JButton("browse");
b.setToolTipText("Converts html file to Text file");
b1.setToolTipText("REMOVES STOP WORDS FROM THE
DOCUMENT");
b2.setToolTipText("STEMS THE WORDS IN THE DOCUMENT");
b3.setToolTipText("COUNTING THE WORDS");
b4.setToolTipText("BROWSE THE FILES");

setLayout(new BorderLayout());
ActionListener l1=new OperatorListener()

/*panel*/
JPanel panel1=new JPanel();
JPanel panel2=new JPanel();

```

```

JPanel panel3=new JPanel();
JProgressBar pb=new JProgressBar(0,100);
//text field label
    panel2.add(new JLabel("Enter The Folder Path", SwingConstants.LEFT));
panel2.add(text);
panel2.add(b4);
//add Button to pannel
panel1.add(b);
panel1.add(b1);
panel1.add(b2);
panel1.add(b3);

//add panel to frame

add(panel1);
add(panel2);
add(panel1,BorderLayout.SOUTH );
add(panel3, BorderLayout.EAST);
add(panel2, BorderLayout.NORTH);

//add action listener to button
b.addActionListener(l1);
b1.addActionListener(l1);

b2.addActionListener(l1);
b3.addActionListener(l1);
b4.addActionListener(l1);

```



```

//adding scroll pane to text area

JScrollPane bar=new JScrollPane(textArea);

add(bar, BorderLayout.CENTER);
pack();

}
class OperatorListener implements ActionListener
{
    private String path;

    public void actionPerformed(ActionEvent ac)
    {
        String input=ac.getActionCommand();
        if(input.equals("browse"))
        {
            myFrame myf=new myFrame();
            chooser = new JFileChooser();
            chooser.setCurrentDirectory(new java.io.File("."));
            String choosertitle ="string here";
            chooser.setDialogTitle(choosertitle );
            chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
            chooser.setAcceptAllFileFilterUsed(false);

            if(chooser.showOpenDialog(myf)== JFileChooser.APPROVE_OPTION) {
                System.out.println("getCurrentDirectory(): " + chooser.getCurrentDirectory());
            }
        }
    }
}

```

```

        System.out.println("getSelectedFile() : " + chooser.getSelectedFile());
        String address=chooser.getSelectedFile().toString();
        text.setText(address);
    }
    else {
        System.out.println("No Selection ");
    }
    myf.dispose();
}

else if(input.equals("HTML TO TEXT"))
    {
        path=text.getText();
        if(path.length()<1)
        {
JOptionPane.showMessageDialog(null,"PLEASE ENTER THE FOLDER'S
PATH","dialog",JOptionPane.ERROR_MESSAGE);
        }
    else{
        try{
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "root");
            Statement statement = con.createStatement();
            statement.executeUpdate("DELETE FROM url");
            con.close();
        }
        catch(Exception e)
        {

```

```

System.out.println("error deleting table");
}
File fo=new File(path);
textArea.append("Searching in..." + fo.getName());
System.out.println("Searching in..." + fo.getName());
String internalNames[] = fo.list();
for(int i=0; i<internalNames.length; i++)
{
    try{
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "root");
        Statement statement = con.createStatement();
        textArea.append("\nProcessing..." + internalNames[i]);
        try {
            // the HTML to convert
            String name=path+"\\ "+internalNames[i];
            int s=i+1;
            String name1=name;
            System.out.println(name1);
            String name2=name1.replace("\\", "-");
            String name3=name2.replace(" ", "*");
            System.out.println(name3);
            statement.executeUpdate("INSERT INTO url (id,url)VALUES ('"+s+"','"+name3+"')");
            FileReader in = new FileReader(name);
            Html2Text parser = new Html2Text();
            parser.parse(in);
            in.close();

```

```

    FileWriter fstream=new FileWriter("C:\\just try\\test\\text"+i+".txt");
    BufferedWriter out=new BufferedWriter(fstream);
    out.write(parser.getText());
    out.close();
}
catch (Exception e)      {
    e.printStackTrace();
}
con.close();
}
catch(Exception e){
    System.out.println("error:"+e);
}
}
JOptionPane.showMessageDialog(null,"Tags removed from HTML files");
}}
else if(input.equals("REMOVE STOP WORDS"))
{
    String path="c:/just try/test";
    File fs=new File(path);
    textArea.append("Searching in..." + fs.getName());
    System.out.println("Searching in..." + fs.getName());
    String internalNames[] = fs.list();
    for(int i=0; i<internalNames.length; i++)
    {
        textArea.append("\nProcessing..." + internalNames[i]);
        try{
            String fname=path+"/"+internalNames[i];

```

```

FileInputStream fstream = new FileInputStream(fname);
DataInputStream in = new DataInputStream(fstream);
BufferedReader br = new BufferedReader(new InputStreamReader(in));
    String strLine;
    Writer output = null;
    File file = new File("C:\\just try\\test2\\stop_word_removed"+i+".txt");
    output = new BufferedWriter(new FileWriter(file));
while ((strLine = br.readLine()) != null)
{
    String[] arr=strLine.split(" ");
        int len=arr.length;
        // words to process?
    Vector words = new Vector();
    for (int j = 0; j < len; j++)
    {
        if (arr[j].trim().length() > 0)
            words.add(arr[j].trim());
    }
    Stopwords stopwords = new Stopwords();
    Enumeration enm = stopwords.elements();
    int j = 0;
    while (enm.hasMoreElements())
    {
        enm.nextElement();
        j++;
    }
}

```

```

String abc;
// check words for being a stopword
if (words.size() > 0)
{
    for ( j = 0; j < words.size(); j++) {
        if(stopwords.is(words.get(j).toString())!=true) {
            abc=words.get(j).toString();
            output.write(abc);
            output.write(" ");
        }
    }
    output.close();
}
catch (Exception e) {
    e.printStackTrace();
}
}

JOptionPane.showMessageDialog(null,"Stop words removed from files");
}
//elseif ends here

else if(input.equals("STEMMING"))
{
    String path="c:/just try/test2";
    File fp=new File(path);
    textArea.append("Searching in..." + fp.getName());
}

```

```

    System.out.println("Searching in..." + fp.getName());
    String internalNames[] = fp.list();
    for(int i=0; i<internalNames.length; i++) {
        textArea.append("\nProcessing..." + internalNames[i]);
    try{
        Writer output2 = null;
        File file2 = new File("c:\\just try\\test3\\stemmed"+i+".txt");
        output2 = new BufferedWriter(new FileWriter(file2));
        char[] w = new char[501];
        Stemmer s = new Stemmer();
        for (int j = 0; j < 1; j++)
            try{
                String stem=path+"/"+internalNames[i];
                FileInputStream in1 = new FileInputStream(stem);
            try{
                while(true) {
                    int ch = in1.read();
                    if (Character.isLetter((char) ch)){
                        int k = 0;
                        while(true){
                            ch = Character.toLowerCase((char) ch);
                            w[k] = (char) ch;
                            if (k < 500) k++;
                            ch = in1.read();
                            if (!Character.isLetter((char) ch)){
                                /* to test add(char ch) */
                                for (int c = 0; c < k; c++) s.add(w[c]);

```

```

/* or, to test add(char[] w, int j) */
    /* s.add(w, j); */
    s.stem();
    { String u;
/* and now, to test toString() : */
    u = s.toString();
/* to test getResultBuffer(), getResultLength() : */
    /* u = new String(s.getResultBuffer(), 0, s.getResultLength()); */
//System.out.print(u);
    output2.write(u);
    }
    break;
}}
}
if (ch < 0) break;
    output2.write((char)ch);
// System.out.print((char)ch);
}
}
catch (IOException e)
{ System.out.println("error reading ");
    break;
}
}
catch (FileNotFoundException e)
{ System.out.println("file not found");
    break; }

```



```

output2.close();

    }
    catch (IOException e)
    { System.out.println("error ");

    }

}
OptionPane.showMessageDialog(null,"All words in the given document's were
stemmed");
}
else if(input.equals("WORD COUNT"))
{
    try{
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");
        Statement statement = con.createStatement();
        statement.executeUpdate("DELETE from list");
        con.close();
    }
    catch(Exception e)
    {
        System.out.println("cannot clear rows"+e);
    }
    String path="c:/just try/test3";

```

```

File fc=new File(path);
textArea.append("Searching in..." + fc.getName());
System.out.println("Searching in..." + fc.getName());
String internalNames[] = fc.list();
try{
    Connection con =DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "root");
    Statement statement = con.createStatement();
    statement.executeUpdate("DELETE FROM list1;");
        con.close();        }
    catch(Exception e)
    {
        System.out.println("cannot delete list1:"+e);
    }
for(int i=0; i<internalNames.length; i++)
    {
        textArea.append("\nProcessing..." +internalNames[i]);
        String w1="c:/just try/test3/" +internalNames[i];
        String w2="Final"+i+".txt";
        wordcount obj=new wordcount(w1,w2,i);
    } try{
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "root");
Statement statement = con.createStatement();
ResultSet rs = statement.executeQuery("SELECT distinct words FROM list order by
words");
String[] words=new String[100000];

```

```

    int i=0;
while(rs.next())
    {
        words[i]=rs.getString("words");
        i=i+1;
    }
    int j=0;
while(j<i)
{
    String name="";
String w=words[j];
rs=statement.executeQuery("select id from list where words='"+w+"'");
while(rs.next()) {
    name=name+rs.getString("id");
    name = name+", ";
    }
    statement.executeUpdate("insert into list1 values '"+name+"','"+w+"'");
        // con.close();
        j=j+1;
    }
    con.close();
}
catch(Exception e)
{
    System.out.println("cannot create view :"+e);
}
// Ranking rank=new Ranking();

```

```

{
try{
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "root");
Statement statement = con.createStatement();
Ranking ranking = new Ranking();
double r=0;
String pid;
for(int i=0;i<9;i++)
{
    pid=Integer.toString(i+1);
    r=ranking.rank(pid);
    System.out.println("rank of "+pid+" = "+r);
    statement.executeUpdate("update url set rank='"+r+"' where id='"+pid+"'");
}
con.close();
}
catch(Exception e){
System.out.println("error occured in insert the rank:");
}
} //ends here
try{
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "root");
Statement statement = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
statement.executeUpdate("DELETE FROM relevance");

```

```

System.out.println(WordNetDemo.method(1,4));
System.out.println(WordNetDemo.method(1,5));
System.out.println(WordNetDemo.method(2,5));
System.out.println(WordNetDemo.method(2,6));
System.out.println(WordNetDemo.method(3,5));
System.out.println(WordNetDemo.method(3,6));
System.out.println(WordNetDemo.method(4,7));
System.out.println(WordNetDemo.method(5,7));
System.out.println(WordNetDemo.method(5,8));
System.out.println(WordNetDemo.method(6,8));
con.close();
}
catch(Exception e) {
    System.out.println("error"+e);
}
JOptionPane.showMessageDialog(null,"ALL Necessary Words Have Been Stored In
DataBase");
}
}
}
}

```

6.6 LOGIN:

```
import a.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
import java.util.*;
import java.io.*;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.Collections;
import java.util.Enumeration;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Vector;
import javax.swing.text.html.*;
import javax.swing.text.html.parser.*;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```

import java.sql.Connection;
import java.sql.SQLException;
public class login extends JFrame
{
    // Variables declaration
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JTextField jTextField1;
    private JPasswordField jPasswordField1;
    private JButton jButton1;
    private JPanel contentPane;
    // End of variables declaration

    public login()
    {
        super();
        create();
        this.setVisible(true);
    }
    private void create()
    {
        jLabel1 = new JLabel();
        jLabel2 = new JLabel();
        jTextField1 = new JTextField();
        jPasswordField1 = new JPasswordField();
        jButton1 = new JButton();
        contentPane = (JPanel)this.getContentPane();
    }
}

```

```

// jLabel1
jLabel1.setHorizontalAlignment(SwingConstants.LEFT);
jLabel1.setForeground(new Color(0, 0, 255));
jLabel1.setText("username:");

// jLabel2
jLabel2.setHorizontalAlignment(SwingConstants.LEFT);
jLabel2.setForeground(new Color(0, 0, 255));
jLabel2.setText("password:");

// jTextField1
jTextField1.setForeground(new Color(0, 0, 255));
jTextField1.setSelectedTextColor(new Color(0, 0, 255));
jTextField1.setToolTipText("Enter your username");
jTextField1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        jTextField1_actionPerformed(e);
    }

});

//
// jPasswordField1
//
jPasswordField1.setForeground(new Color(0, 0, 255));
jPasswordField1.setToolTipText("Enter your password");

```



```

jPasswordField1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        jPasswordField1_actionPerformed(e);
    }
});

// jButton1
jButton1.setBackground(new Color(204, 204, 204));
jButton1.setForeground(new Color(0, 0, 255));
jButton1.setText("Login");
jButton1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        jButton1_actionPerformed(e);
    }
});

//
// contentPane
//
contentPane.setLayout(null);
contentPane.setBorder(BorderFactory.createEtchedBorder());
contentPane.setBackground(new Color(204, 204, 204));
addComponent(contentPane, jLabel1, 5,10,106,18);
addComponent(contentPane, jLabel2, 5,47,97,18);
addComponent(contentPane, jTextField1, 110,10,183,22);
addComponent(contentPane, jPasswordField1, 110,45,183,22);

```

```

        addComponent(contentPane, jButton1, 150,75,83,28);
        //
        // login
        //
        this.setTitle("Login To Members Area");
        this.setLocation(new Point(76, 182));
        this.setSize(new Dimension(335, 141));
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        this.setResizable(false);
    }

    /** Add Component Without a Layout Manager (Absolute Positioning) */
    private void addComponent(Container container,Component c,int x,int y,int
width,int height)
    {
        c.setBounds(x,y,width,height);
        container.add(c);
    }
    private void jTextField1_actionPerformed(ActionEvent e)
    {
    }
    private void jPasswordField1_actionPerformed(ActionEvent e)
    {
    }
    private void jButton1_actionPerformed(ActionEvent e)
    {

```

```

        String value1=jTextField1.getText();
        String value2=jPasswordField1.getText();
        try{
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
        DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");
        Statement st=con.createStatement();
        ResultSet rs=st.executeQuery("select * from administrator where
username='"+value1+"' and password='"+value2+"'");
        String uname="",pass="";
        if(rs.next()){
            uname=rs.getString("username");
            pass=rs.getString("password");
        }
        if (value1.equals(uname) && value2.equals(pass))
        {
            this.setVisible(false);
            myFrame j=new myFrame();
            j.setVisible(true);

        }
        else{
        JOptionPane.showMessageDialog(null,"Incorrect login or
password","Error",JOptionPane.ERROR_MESSAGE);

        jTextField1 .setText("");
        jPasswordField1.setText("");

```

```

    }
    }
    catch(Exception e1){ }
}

    public static void main(String[] args)
    {
        JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);
        try
        {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAn
dFeel");
        }
        catch (Exception ex){
            System.out.println("Failed loading L&F: ");
            System.out.println(ex);
        }
        new login();
    }
}

```

6.7 RANKING:

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

```

```

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import Jama.Matrix;

/*
 * Developed by Nima Goodarzi
 * Website: http://www.javadev.org
 * Email: nima@javadev.org
 */
public class Ranking {
    private final double DAMPING_FACTOR = 0.85;
    private List params = new ArrayList();
    //public static void main(String[] args) {
    /*
     *
     * Solve the equation of  $ax=b$ , which : a is the generated matrix based on
     *
     * the parameter constants. x is the page ranks matrix. b is a  $n*1$  matrix
     *
     * which all the values are equal to the damping factor.
     */

    public double rank(String pageId) {
        generateParamList(pageId);
    }

```

```

    Matrix a = new Matrix(generateMatrix());
    double[][] arrB = new double[params.size()][1];
    for (int i = 0; i < params.size(); i++) {
        arrB[i][0] = 1 - DAMPING_FACTOR;
    }
    Matrix b = new Matrix(arrB);
    // Solve the equation and get the page ranks
    Matrix x = a.solve(b);
    int ind = 0;
    int cnt = 0;
    for (Iterator it = params.iterator(); it.hasNext();) {
        String curPage = (String) it.next();
        if (curPage.equals(pageId))
            ind = cnt;
        cnt++;
    }
    double y=x.toArray()[ind][0];
        return y;
    }
    /*
    * This method generates the matrix of the linear equations. The generated
    * matrix is a n*n matrix where n is number of the related pages.
    */

```

```

private double[][] generateMatrix() {
    double[][] arr = new double[params.size()][params.size()];
    for (int i = 0; i < params.size(); i++) {

```

```

        for (int j = 0; j < params.size(); j++) {
            arr[i][j] = getMultiFactor((String) params.get(i),
                (String) params.get(j));
        }
    }

    return arr;
}

/*
 * This method returns the constant of the given variable in the linear
 * equation.
 */

private double getMultiFactor(String sourceId, String linkId) {
    if (sourceId.equals(linkId))
        return 1;
    else {
        String[] inc = getInboundLinks(sourceId);
        for (int i = 0; i < inc.length; i++) {
            if (inc[i].equals(linkId)) {
                return -1
                    * (DAMPING_FACTOR / getOutboundLinks(linkId).length);
            }
        }
    }
    return 0;
}

/*

```

* This method returns list of the related pages. This list is also the
* parameters in the linear equation.

*/

```
private void generateParamList(String pageId) {  
    // Add the starting page.  
    if (!params.contains(pageId))  
        params.add(pageId);  
    // Get list of the inbound pages  
    String[] inc = getInboundLinks(pageId);  
    // Add the inbound links to the params list and do same for inbound  
    // links  
    for (int i = 0; i < inc.length; i++) {  
        if (!params.contains(inc[i]))  
            generateParamList(inc[i]);  
    }  
}
```

/*

* Return list of the inbound links to a given page.

*/

```
private String[] getInboundLinks(String pageId) {  
  
    // This simulates a simple page collection  
    Map map = new HashMap();  
    map.put("2", new String[] { "1" });  
    map.put("3", new String[] { "1" });  
    map.put("4", new String[] { "1" });
```



```

map.put("5", new String[] { "1" });
map.put("6", new String[] { "2" });
map.put("7", new String[] { "2" });
map.put("8", new String[] { "2" });
map.put("9", new String[] { "3" });
map.put("10", new String[] { "4" });
map.put("11", new String[] { "4" });
map.put("12", new String[] { "4" });
map.put("13", new String[] { "5" });
map.put("14", new String[] { "5" });
map.put("15", new String[] { "5" });
map.put("16", new String[] { "9" });
map.put("1", new String[] { "15" });
map.put("11", new String[] { "14" });
map.put("7", new String[] { "1" });
/*map.put("4", new String[] { "1" });
map.put("5", new String[] { "1" });
map.put("5", new String[] { "2" });
map.put("5", new String[] { "3" });
map.put("7", new String[] { "5" });
map.put("8", new String[] { "5" });
map.put("6", new String[] { "3" });
map.put("7", new String[] { "4" });*/
return (String[]) map.get(pageId);
}

```

/*

* Returns list of the outbound links from a page.

*/

```
private String[] getOutboundLinks(String pageId) {  
    // This simulates a simple page collection  
    Map<String, String[]> map = new HashMap();  
    map.put("1", new String[] { "2", "3","4","5" });  
    map.put("2", new String[] { "6", "7" ,"8" });  
    map.put("3", new String[] { "9" });  
    map.put("4", new String[] { "10", "11" ,"12" });  
    map.put("5", new String[] { "13", "14" ,"15" });  
    map.put("9", new String[] { "15" });  
    map.put("15", new String[] { "1" });  
    map.put("14", new String[] { "11" });  
    map.put("1", new String[] { "7" });  
    /*map.put("1", new String[] { "4","5"});  
    map.put("2", new String[] { "5" });  
    map.put("3", new String[] { "5","6"});  
    map.put("4", new String[] { "7" });  
    map.put("5", new String[] { "7","8"});  
    map.put("6", new String[] { "8" });*/  
    return (String[]) map.get(pageId);  
}  
  
}
```

6.8 WORDNET DEMO:

```
package examples;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import rita.wordnet.RiWordnet;

public class WordNetDemo {
    static double denom=0;
    public static double method(int i,int j) {
        String start;
        int f=0;
        double dist1=0;
        try{
            Connection con =DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "root");
            Statement statement = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            // count no. of documents

            ResultSet rs=statement.executeQuery("SELECT  words,frequency  FROM list where
id='"+i+"'");
            dist1=0;
            while(rs.next()){
```

```

        start = rs.getString("words");
        f=rs.getInt("frequency");
        double d= method1(j,start,f);
        System.out.println("d="+d);
        dist1=dist1+d;
        System.out.println("dist1="+dist1+"denom="+WordNetDemo.denom)
    ;

    }

    dist1=dist1/WordNetDemo.denom;

    System.out.println("rel between doc "+i+"doc"+j+"="+dist1)
    statement.executeUpdate("insert      into      relevance      values('"+i+"",
    '"+j+"', '"+dist1+"')");
    con.close();
    }catch(Exception e)
    { System.out.println("Exception :"+e);}
    return dist1;
}

public static double method1(int j1,String start, int f1)
{
    RiWordnet wordnet1 = new RiWordnet(null);
    double rel=0;
    float dist=(float)0.000000001;
    //float denom=0;

    try{
Connection    con1    =DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "root");
Statement statement1=con1.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE

```

```

,ResultSet.CONCUR_UPDATABLE);
ResultSet bs = statement1.executeQuery("SELECT words, frequency FROM list where
id='"+j1+"'");
while(bs.next())
{
    String end = bs.getString("words");
    int f2=bs.getInt("frequency");
    String    pos = wordnet1.getBestPos(start);
    System.out.println("\n\nRelationship between: " + start + " and " + end);
    try{
        if (start!=end)dist = wordnet1.getDistance(start,end,pos);
        else {dist=(float)1.0;System.out.println("dist error");}
        if (dist==0.0)dist=(float)0.1;
    }catch(Exception e){dist=(float)0.1;
    System.out.println("dist error in catch");}
    System.out.println(start + " and " + end + " are related by a distance of: " +
dist+"j"+j1);
    rel=rel+(f1*f2/dist);
    WordNetDemo.denom=WordNetDemo.denom+(f1*f2);
    System.out.println("method1 "+rel+"denom="+WordNetDemo.denom);
}
con1.close();
System.out.println("method1 "+rel);
}
catch(Exception e){
    System.out.println("Exception :");
}
}

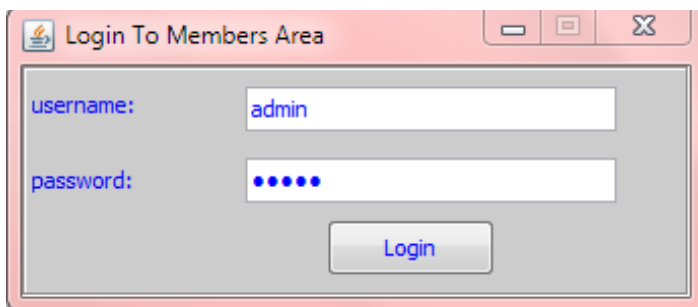
```

```
        return rel; // relevance based on one word in current doc and all words in  
other doc  
    } // end of method1  
} // end of class
```

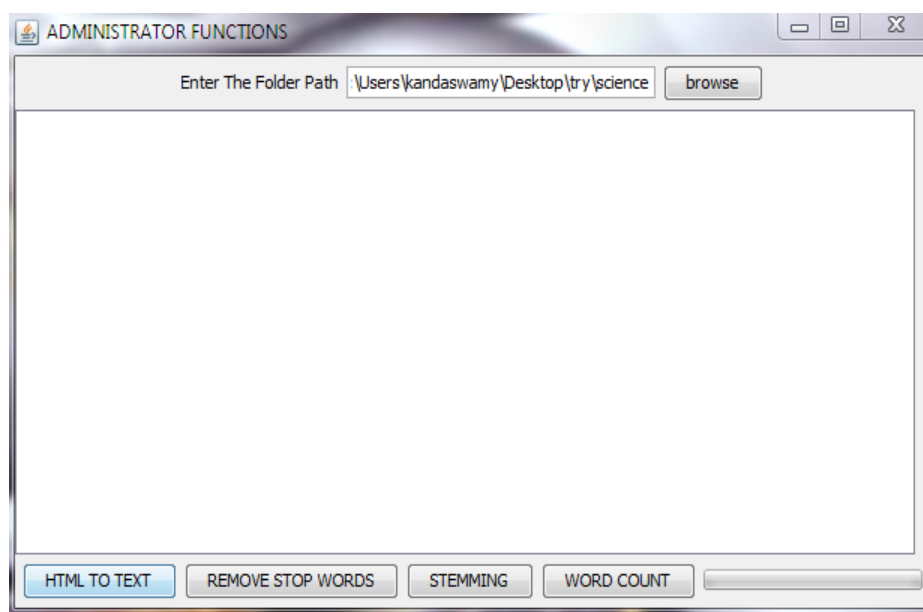
CHAPTER 7

SCREEN SHOTS

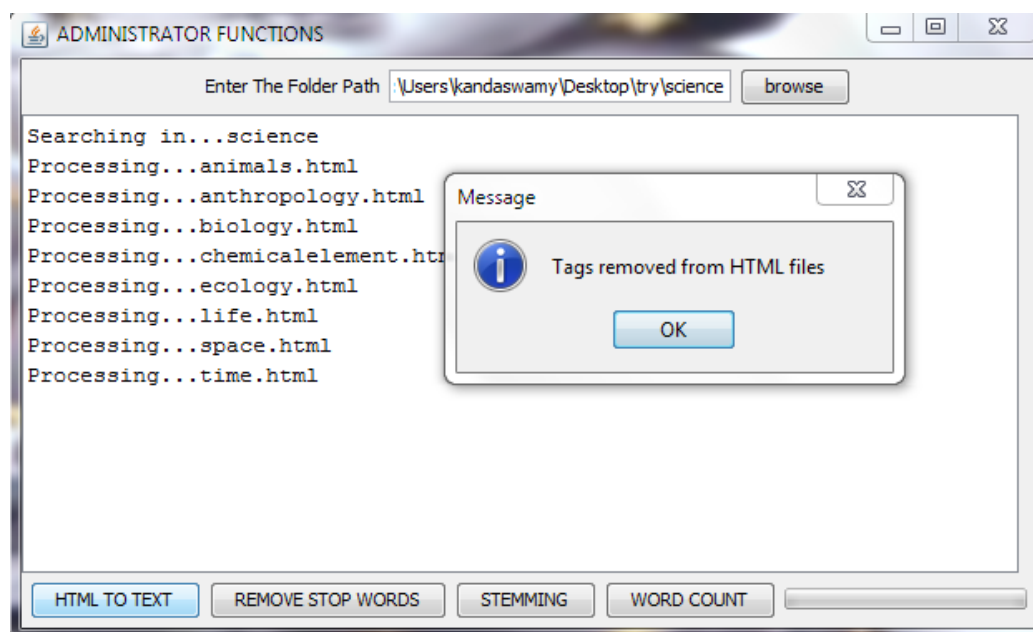
7.1 LOGIN FOR ADMINISTRATOR

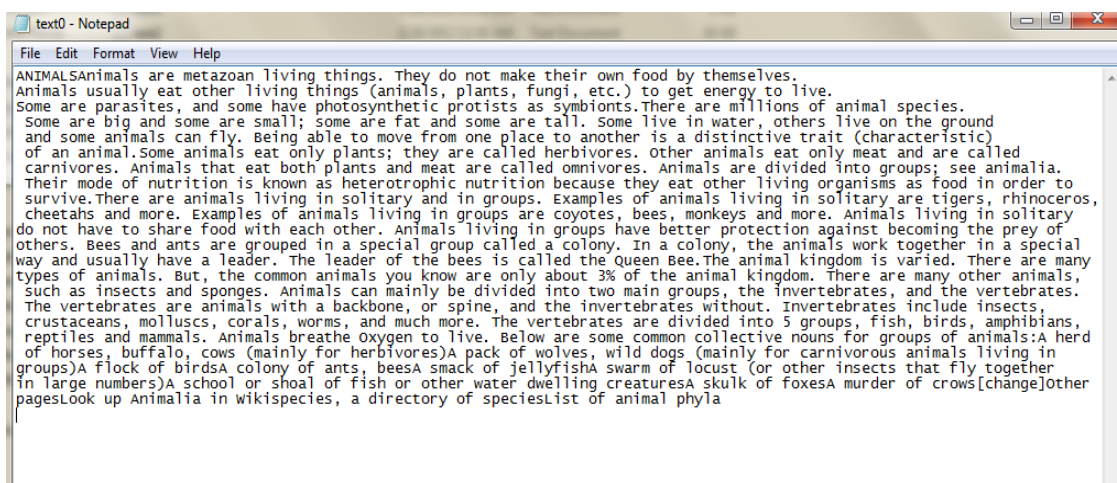


7.2 ADMINISTRATOR FUNCTIONS

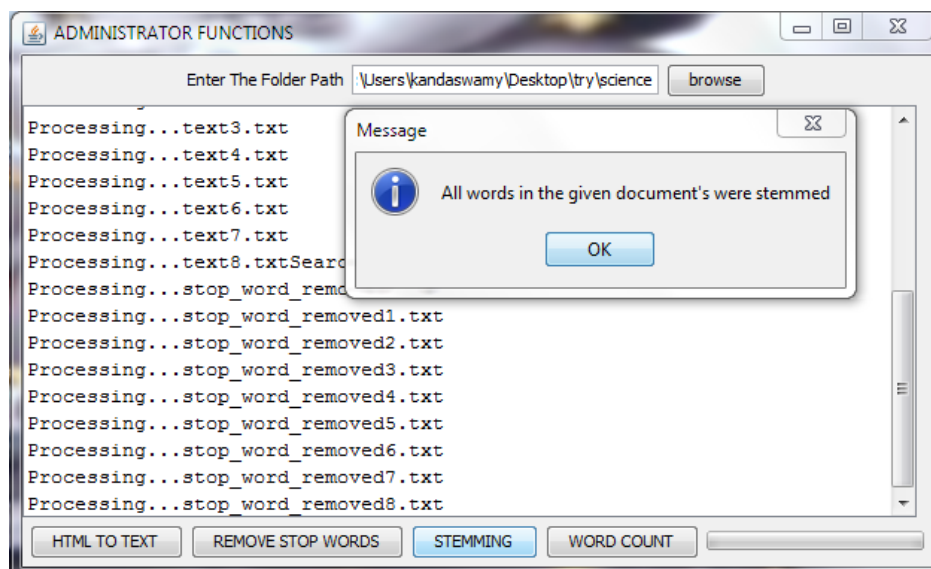


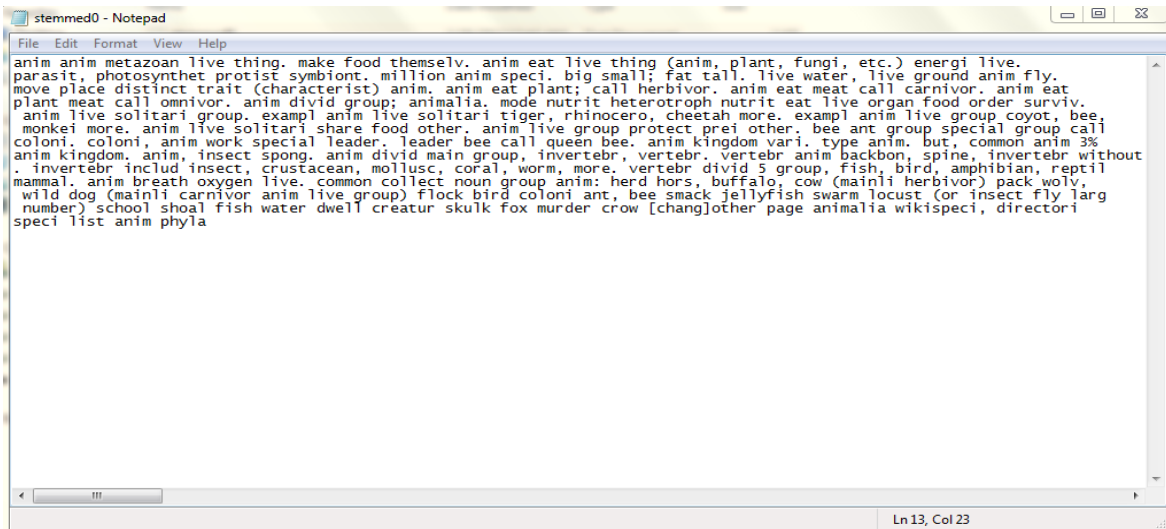
7.3 REMOVING HTML TAGS



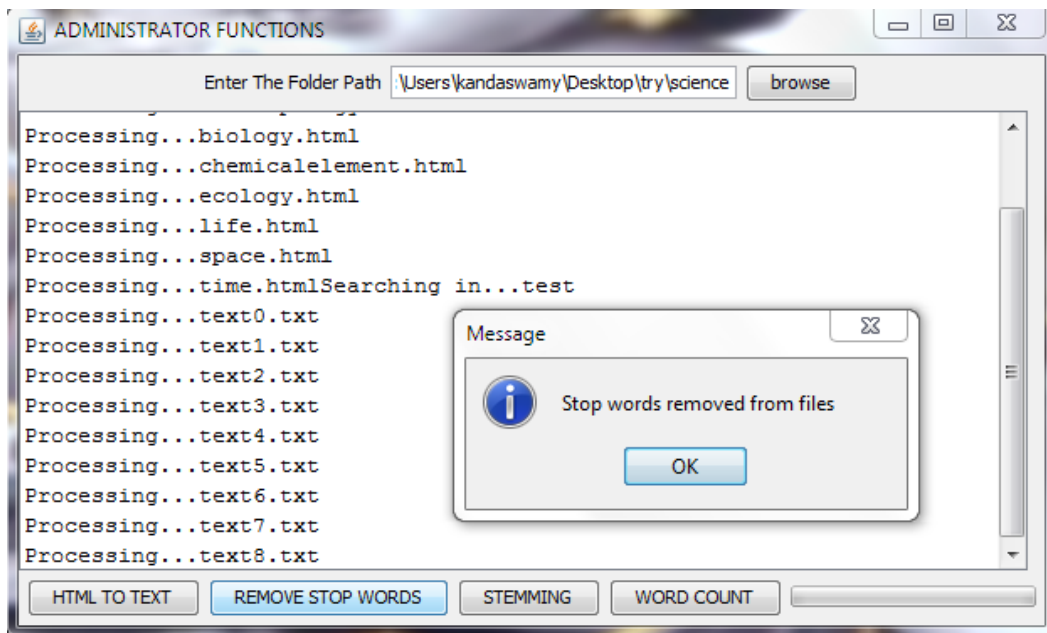


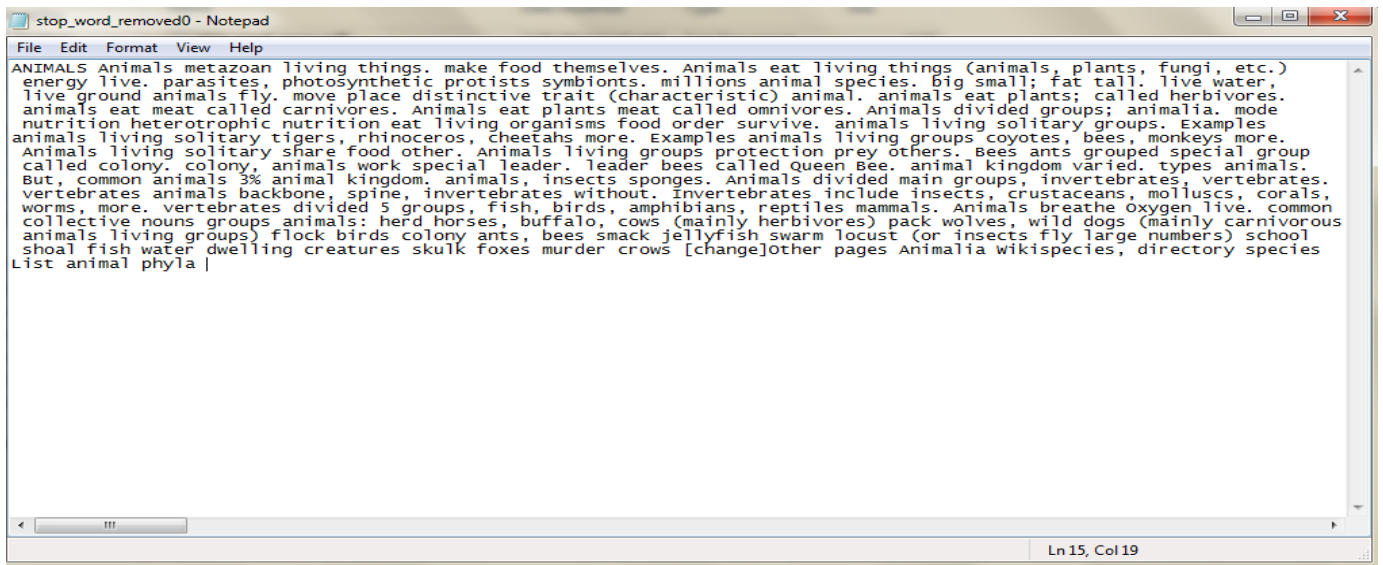
7.4 STEMMED WORDS



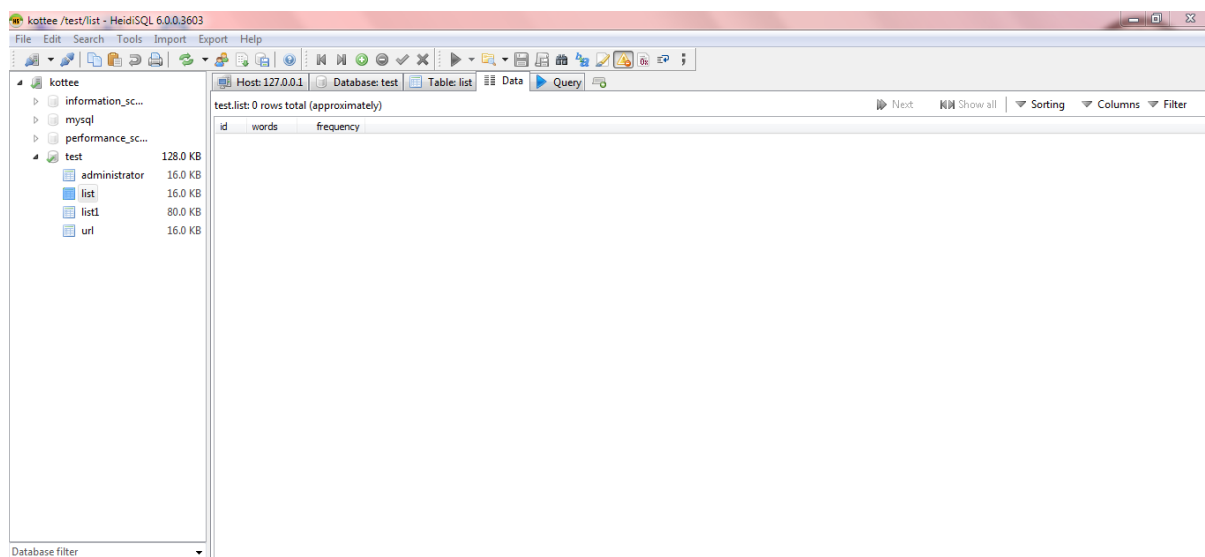


7.5 REMOVING STOPWORDS

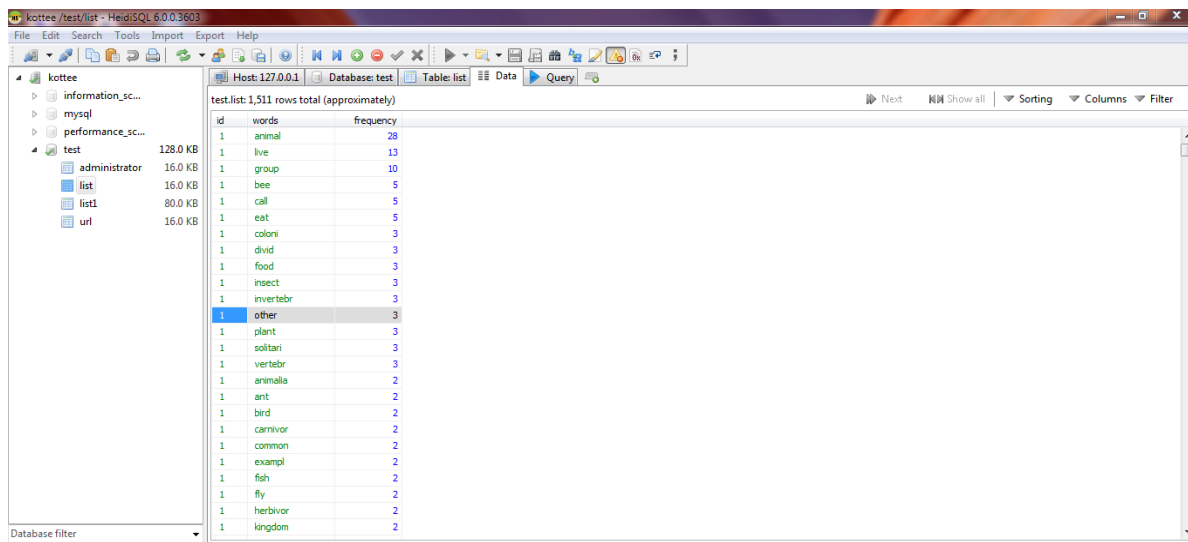




7.6 EMPTY DATABASE FOR STORING WORD COUNT VALUES



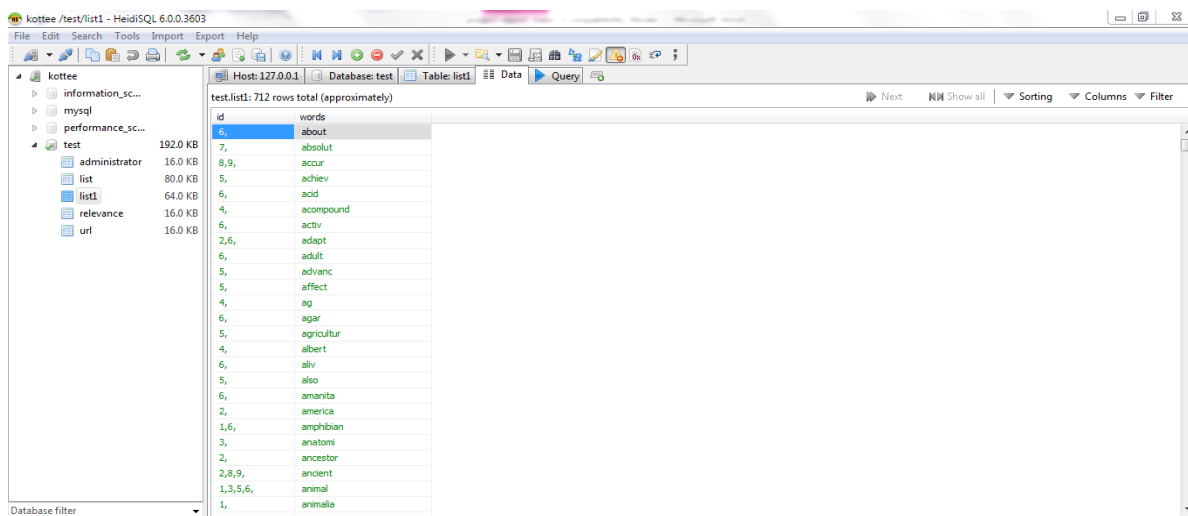
7.7 DATABASE UPDATED WITH WORDS AND FREQUENCY



test.list: 1,511 rows total (approximately)

id	words	frequency
1	animal	28
1	live	13
1	group	10
1	bee	5
1	call	5
1	eat	5
1	colori	3
1	divid	3
1	food	3
1	insect	3
1	invertebr	3
1	other	3
1	plant	3
1	solitari	3
1	vertebr	3
1	animalia	2
1	ant	2
1	bird	2
1	carnivor	2
1	common	2
1	exampl	2
1	fish	2
1	fly	2
1	herbivor	2
1	kingdom	2

7.8 COLLECTING THE WORDS WITH DIFFERENT ID'S



test.list1: 712 rows total (approximately)

id	words
6,	about
7,	absolut
8,9,	accor
5,	achiev
6,	acid
4,	acomound
6,	activ
2,6,	adapt
6,	adult
5,	advanc
5,	affect
4,	ag
6,	agar
5,	agricultur
4,	albert
6,	oliv
5,	also
6,	amanita
2,	america
1,6,	amphibian
3,	anatomi
2,	ancestor
2,8,9,	ancient
1,3,5,6,	animal
1,	animalia

7.9 DATABASE UPDATED WITH URL,RANK, AND RELEVANCE.

Host: 127.0.0.1 Database: test Table: url

test.url: 8 rows total (approximately)

id	url	rank	relevance
1	http://en.wikipedia.org/wiki/Animal	0.39435217687431234	0.26752076886714526
2	http://en.wikipedia.org/wiki/Biology	0.4851993503431655	0.26074436205136076
3	http://en.wikipedia.org/wiki/Ecology	0.4851993503431655	0.2446092400814003
4	http://en.wikipedia.org/wiki/Anthropology	0.4851993503431655	0.4851993503431655
5	http://en.wikipedia.org/wiki/Life	0.48519935034316547	0.29887320806841516
6	http://en.wikipedia.org/wiki/Space	0.2874731492638969	0.2874731492638969
7	http://en.wikipedia.org/wiki/Chemical_element	0.4851993503431655	0.26679205204573764
8	http://en.wikipedia.org/wiki/Time	0.2874731492638969	0.16204744638819224

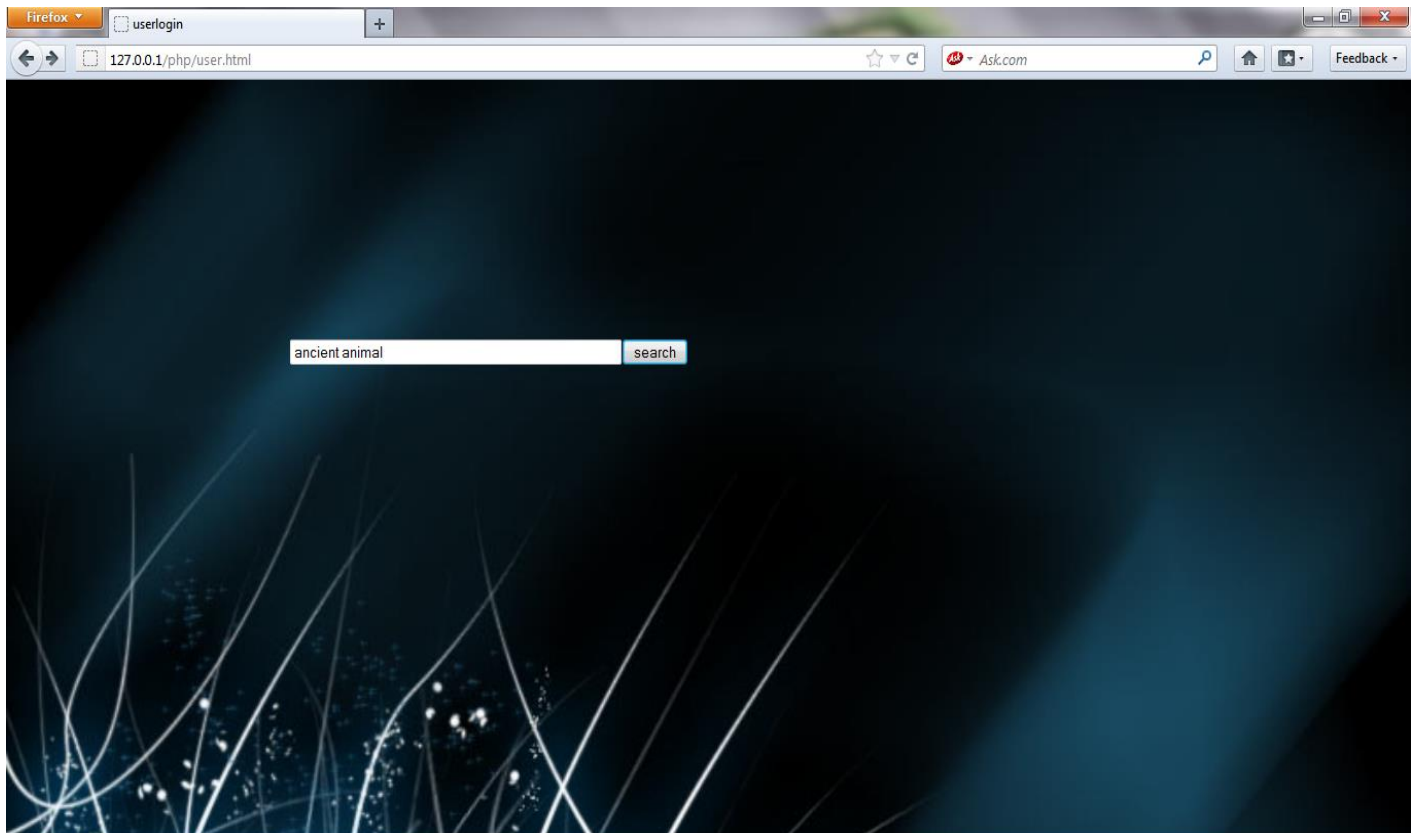
7.10 DATABASE UPDATED WITH RELEVANCE BETWEEN THE EACH DOCUMENT

Host: 127.0.0.1 Database: test Table: relevance

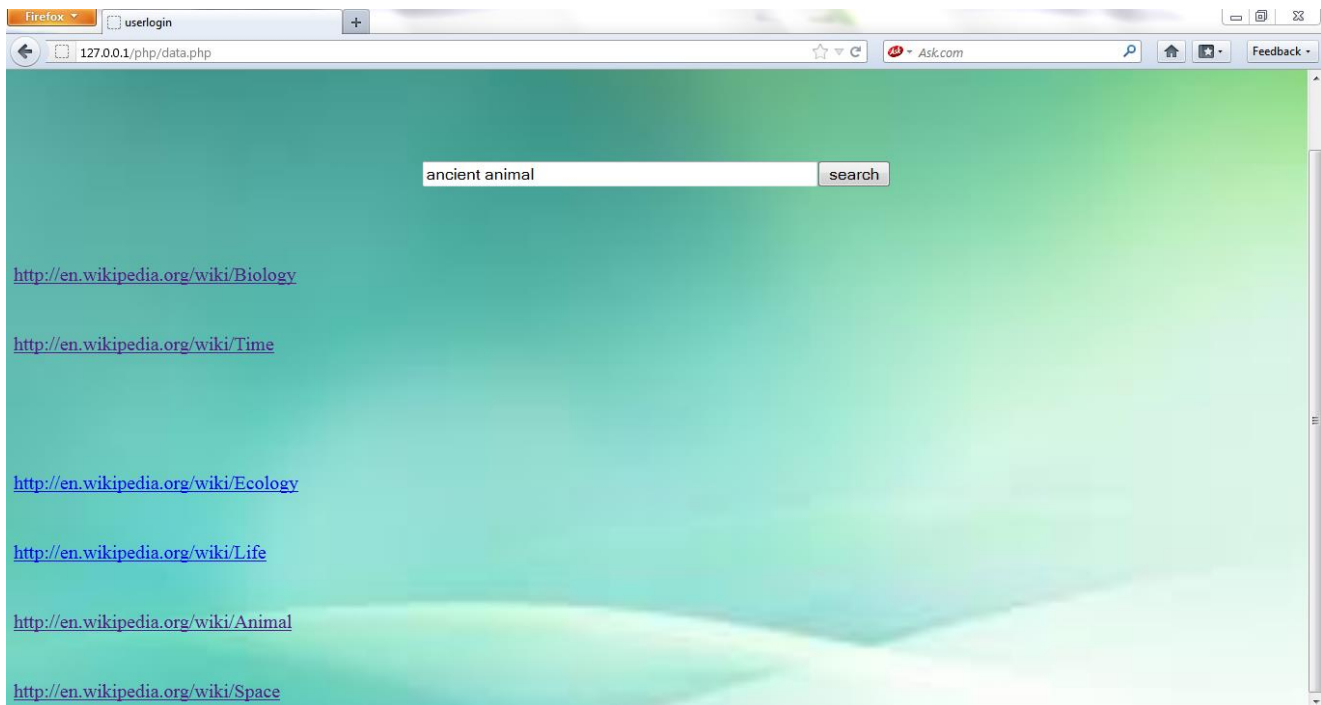
test.relevance: 10 rows total (approximately)

doc id	out id	pagerellevance
1	4	3.4388457967806416
1	5	1.630354369016551
2	5	1.5972383601882714
2	6	1.374941013545911
3	5	0.9300940680045857
3	6	0.9148118619745151
4	7	0.23520268611982376
5	7	0.3472715616266771
5	8	0.5570349089353647
6	8	0.4040019567276424

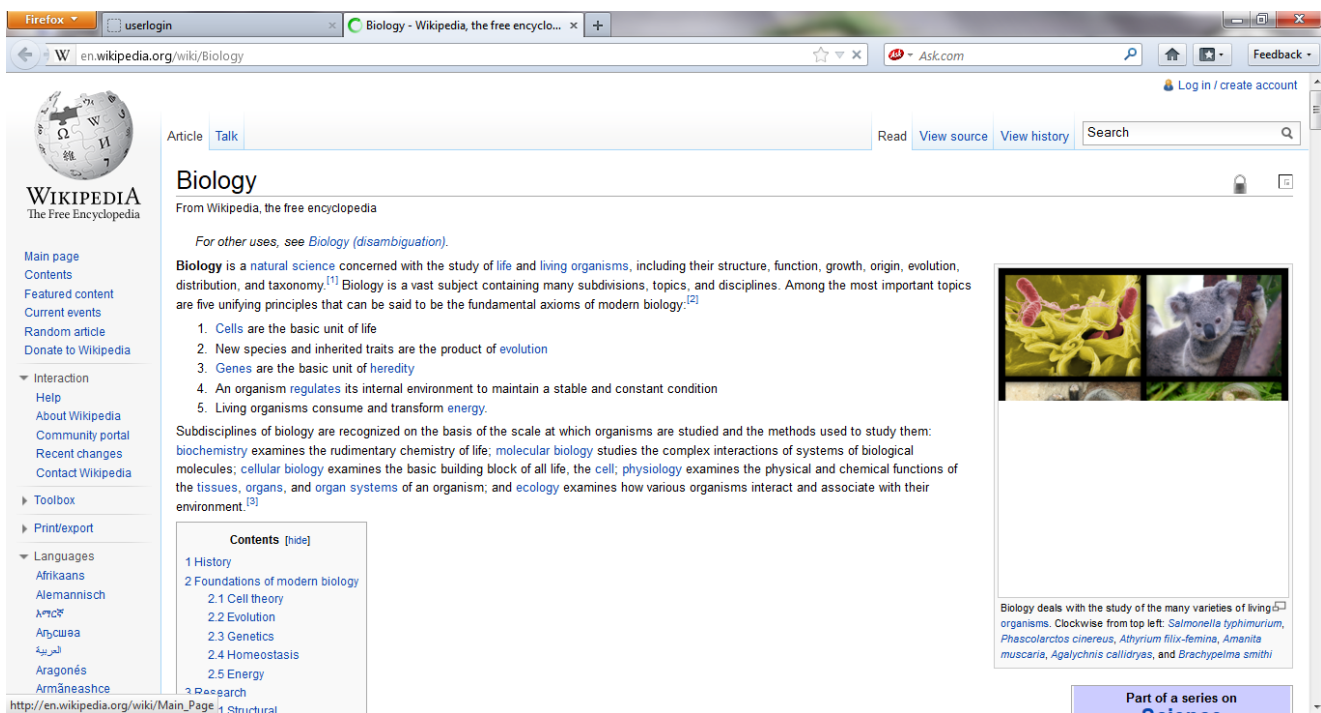
7.11 USER'S SEARCH ENGINE



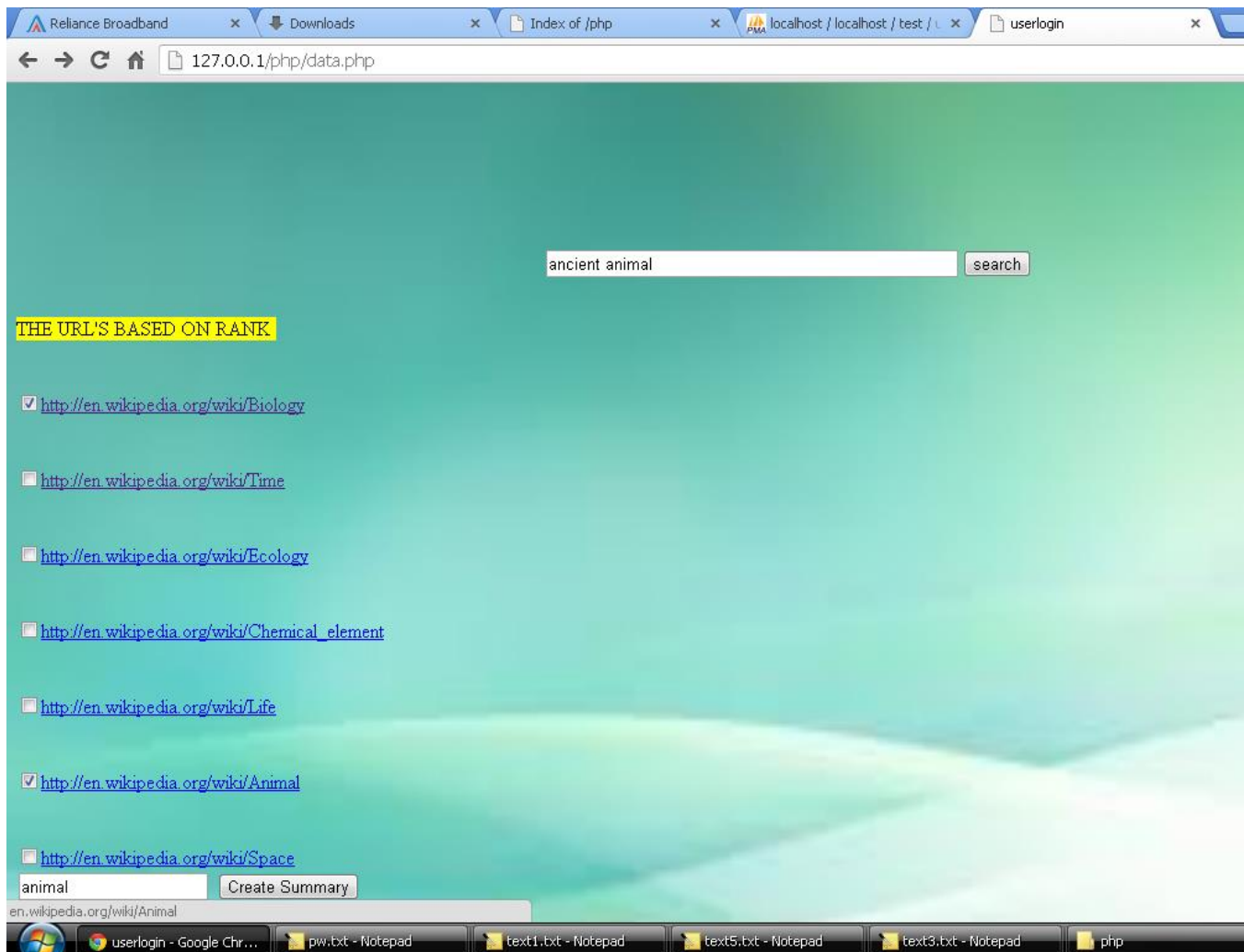
7.12 RANK BASED DOCUMENTS URL'S



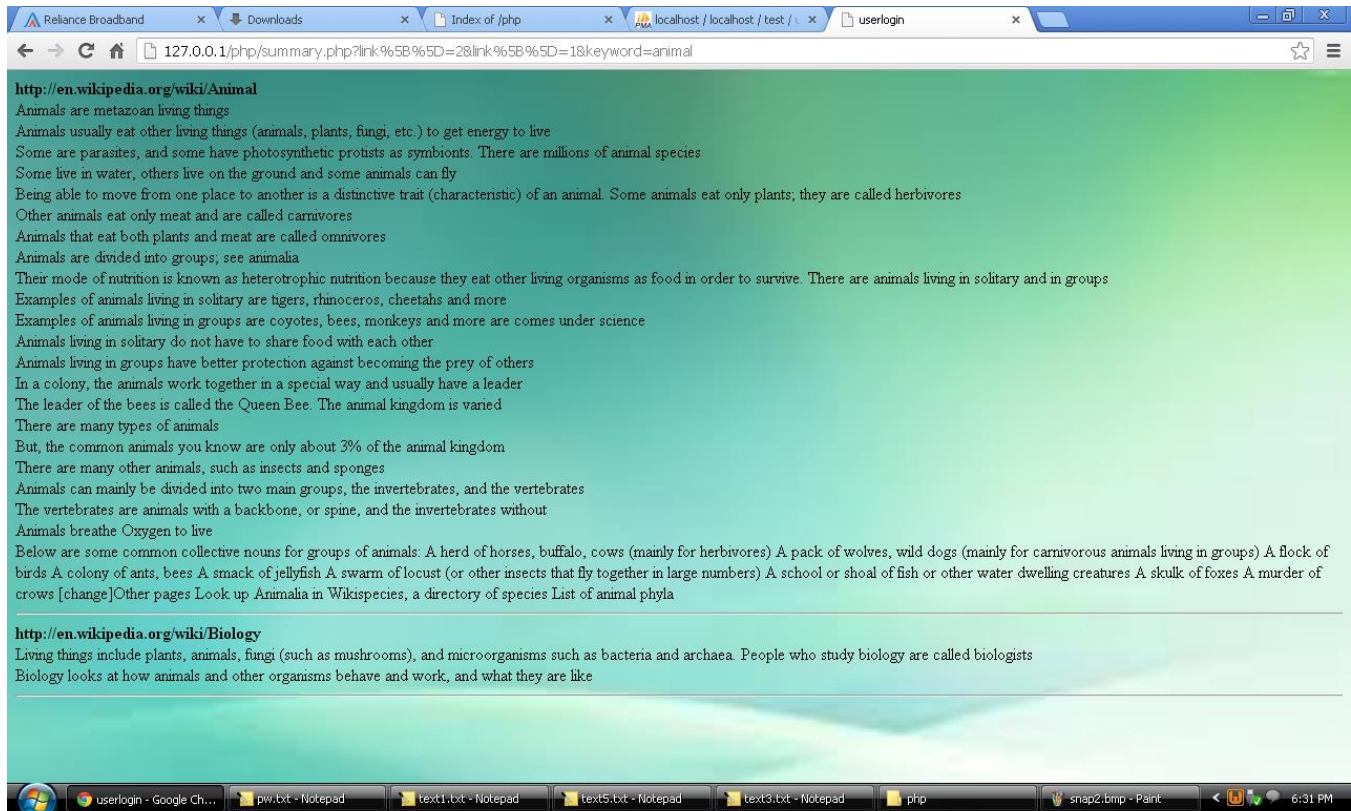
7.13 DOCUMENT IN THE WEB BROWSER



7.14 SELECTING THE URL LINKS FOR REQUESTING SUMMARY



7.15 SUMMARY FOR THE SELECTED LINKS IN THE SEARCH RESULT



CHAPTER 8

CONCLUSION AND FUTURE WORK

The input is the search key word and that is used to compare the key words with those stored in the data base. The result shown is split into two screens where one side of the screen displays the algorithm and the other side displays the user ranking based page ranking .The Summarization can also be made as a tool where the users can use it in every page where it will highlight only the important points based on the title of the article .There is a current facility in google for searching images alone, so similarly a separate search could be used to search mp3 songs.

CHAPTER 9

REFERENCES

- 1) Chia-Chen Yen and Jih-Shih Hsu, “Associated page rank: Improved page rank measured by frequent term sets”, IEEE 2009.
- 2) Zha Peng , Xu Xiu , Zuo Ming , “An Efficient Improved Strategy for the PageRank Algorithm”, IEEE 2011
- 3) Zhou Cailan, Chen Kai, Li Shasha , “Improved PageRank Algorithm Based on Feedback of User Clicks”, IEEE 2011.
- 4) L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: bringing order to the web”, IEEE 1998.
- 5) Wenpu Xing and Ali Ghorbani, “Weighted PageRank Algorithm” IEEE 2004.
- 6) A. Kritikopoulos, M. Sideri and I. Varlamis, “Wordrank: A Method for Ranking Web Pages Based on Content Similarity”, IEEE 2007.
- 7) C.H. Li and K.Q. Lv, “Hyperlink Classification: A new approach to improve pagerank”. IEEE 2007.
- 8) Ao-Jan Su, Y. Charlie Hu, Aleksandar Kuzmanovic, and Cheng-Kok Koh, “How to Improve Your Google Ranking: Myths and Reality”, IEEE 2010.