**CHAPTER 1**

**INTRODUCTION**

A **reverse dictionary** is a [dictionary](http://en.wikipedia.org/wiki/Dictionary) organized in a non-standard order that provides the user with information that would be difficult to obtain from a traditionally alphabetized dictionary. As opposed to the regular dictionary, it maps words to their definitions. Reverse dictionary performs the converse mapping. i.e, given a phrase describing the desired concept it provides words whose definitions match the entered definition phrase.

For example, supposed a forward dictionary informs the user that the meaning of the word “castigate” is “severe scolding”, on the other hand the reverse dictionary offers the user an opportunity to get the answer “castigation” along with other relative words when “verbal punishment” is given as input.

In these reverse dictionaries all words that have the same [suffix](http://en.wikipedia.org/wiki/Suffix) appear in order in the dictionary. Such a reverse dictionary would be useful for linguists and poets who might be looking for words ending with a particular suffix, or by an anthropologist or forensics specialist examining a damaged text (e.g. a stone inscription, or a burned document) that had only the final portion of a particular word preserved. By way of contrast, in a standard dictionary words are organized such that words with the same prefix appear in order, since the sorting order is starting with the first letter of the entry word and subsequent letters proceeding toward the end of that word. It can provide significant improvements in performance scale without satisfying the quality of the result.

**CHAPTER 2**

**2. SYSTEM STUDY**

**2.1 Existing System**

A meaning phrase is given as input so it will give an output which is appropriate words for the given phrase. It will use forward dictionary to find the word.

**2.2 Proposed System**

The proposed system has greater accuracy than the existing system. This system provides greater accuracy as number of words is reduced. We use html to design the User Interface through which he would give the input. After getting the input, Java Servlets is used to link the HTML to JavaScript. Within the JavaScript, mapping takes place after which the output is obtained and sent back to the UI.

Advantages:

* Better interaction between user and system.
* Greater accuracy in terms of words obtained.
* More efficient tool for people in Literature field.

**2.3 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal Is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* OPERATIONAL FEASIBILITY
* TECHNICAL FEASIBILITY
* ECONOMICAL FEASIBILITY

**OPERATIONAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**ECONOMICAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it.

**CHAPTER 3**

**SYSTEM SPECIFICATION**

**3.1 Hardware Specification**

System : Pentium IV 2.4 GHz.

Hard Disk : 100 MB.

Monitor : 15 VGA Colour.

Ram : 512 MB.

**3.2 Software Specification**

Operating system : Windows XP.

Framework : Visual Studio 4.0.

Coding Language : jsp, ajax, html, java servlets

**CHAPTER 4**

**LANGUAGE DESCRIPTION**

**4.1 Overview of JSP\SERVLETS:**

JavaServer Pages (JSP) is a technology that helps [software developers](http://en.wikipedia.org/wiki/Software_developer) create [dynamically generated web pages](http://en.wikipedia.org/wiki/Dynamic_web_page) based on [HTML](http://en.wikipedia.org/wiki/HTML),[XML](http://en.wikipedia.org/wiki/XML), or other document types .To deploy and run JavaServer Pages, a compatible web server with a [servlet](http://en.wikipedia.org/wiki/Servlet) container, such as [ApacheTomcat](http://en.wikipedia.org/wiki/Apache_Tomcat) or [Jetty](http://en.wikipedia.org/wiki/Jetty_(web_server)), is required JSP can be used independently or as the view component of a server-side [model–view–controller](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) design, normally with [JavaBeans](http://en.wikipedia.org/wiki/JavaBeans) as the model and Java servlets (or a framework such as [Apache)](http://en.wikipedia.org/wiki/Apache_Struts)  as the controller. JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, use Java byte code rather than a native software format. Like any other Java program, they must be executed within a [Java virtual machine](http://en.wikipedia.org/wiki/Java_virtual_machine) (JVM).

The servlet is a [Java programming language](http://en.wikipedia.org/wiki/Java_programming_language) [class](http://en.wikipedia.org/wiki/Class_(computer_programming)) used to extend the capabilities of a [server](http://en.wikipedia.org/wiki/Server_(computing)). Although servlets can respond to any types of requests, they are commonly used to extend the applications hosted by [web servers](http://en.wikipedia.org/wiki/Web_server), so they can be thought of as [Java applets](http://en.wikipedia.org/wiki/Java_applet) that run on [servers](http://en.wikipedia.org/wiki/Server_(computing)) instead of in [web browsers](http://en.wikipedia.org/wiki/Web_browser).

**4.2 CLIENT-SIDE DEVELOPMENT**

The client side consists of HTML, CSS, JAVASRIPT and AJAX. HTML is used to design the front end or the user interface. CSS is “Cascading Style Sheets” that is used to change the font size, font colour, etc. of the information in the HTML webpage. With Ajax, Web applications can send data to, and retrieve data from, a [server](http://en.wikipedia.org/wiki/Web_server) asynchronously (in the background) without interfering with the display and behavior of the existing page.

**4.3 SERVER-SIDE DEVELOPMENT**

The server side consists of the main logic of the software in addition to Java servlets. Java Servlets are the java applets or java applications that run on the server. They are basically the java counterpart to the other dynamic web content technologies such as PHP and ASP.NET. Servlets can be generated automatically from [Java Server Pages](http://en.wikipedia.org/wiki/Java_Server_Pages) (JSP) by the [JavaServer Pages compiler](http://en.wikipedia.org/wiki/JavaServer_Pages_compiler). The difference between servlets and JSP is that servlets typically embed HTML inside Java code, while JSPs embed Java code in HTML.

**4.4 OVERVIEW OF AJAX**

Ajax (an acronym for Asynchronous [JavaScript](http://en.wikipedia.org/wiki/JavaScript) and [XML](http://en.wikipedia.org/wiki/XML))  is a group of interrelated [Web development](http://en.wikipedia.org/wiki/Web_development) techniques used on the [client-side](http://en.wikipedia.org/wiki/Client-side) to create [asynchronous](http://en.wikipedia.org/wiki/Asynchronous_I/O) [Web applications](http://en.wikipedia.org/wiki/Web_application). With Ajax, Web applications can send data to, and retrieve data from, a [server](http://en.wikipedia.org/wiki/Web_server) asynchronously (in the background) without interfering with the display and behavior of the existing page. Data can be retrieved using the [XMLHttpRequest](http://en.wikipedia.org/wiki/XMLHttpRequest) [object](http://en.wikipedia.org/wiki/Object_(computer_science)). Ajax is not a single technology, but a group of technologies. [HTML](http://en.wikipedia.org/wiki/Hypertext_Markup_Language) and [CSS](http://en.wikipedia.org/wiki/Cascading_Style_Sheets) can be used in combination to mark up and style information. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.

**4.5 OVERVIEW OF NLP PARSING**

Natural language processing (NLP) is a field of [computer science](http://en.wikipedia.org/wiki/Computer_science), [artificial intelligence](http://en.wikipedia.org/wiki/Artificial_intelligence), and [linguistics](http://en.wikipedia.org/wiki/Linguistics) concerned with the interactions between [computers](http://en.wikipedia.org/wiki/Computers) and [human (natural) languages](http://en.wikipedia.org/wiki/Natural_language). As such, NLP is related to the area of [human–computer interaction](http://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction). Many challenges in NLP involve [natural language understanding](http://en.wikipedia.org/wiki/Natural_language_understanding), that is, enabling computers to derive meaning from human or natural language input, and others involve natural language generation. The NLP used by us is called “Standford NLP parser”.

**4.6 OVERVIEW OF WORDNET FRAMEWORK**

**WordNet is** a forward dictionary lexical database for the English language. It groups English [words](http://en.wikipedia.org/wiki/Word) into sets of [synonyms](http://en.wikipedia.org/wiki/Synonyms) called [synsets](http://en.wikipedia.org/wiki/Synsets), provides short, general definitions, and records the various [semantic](http://en.wikipedia.org/wiki/Semantic) relations between these synonym sets. The purpose is twofold: to produce a combination of [dictionary](http://en.wikipedia.org/wiki/Dictionary) and [thesaurus](http://en.wikipedia.org/wiki/Thesaurus) that is more intuitively usable, and to support automatic [text analysis](http://en.wikipedia.org/wiki/Natural_language_processing) and  applications. The [database](http://en.wikipedia.org/wiki/Database) and [software](http://en.wikipedia.org/wiki/Software) tools have been released under a [BSD style license](http://en.wikipedia.org/wiki/BSD_License) and can be downloaded and used freely. The database can also be browsed online.

**4.7 OVERVIEW OF JAVASCRIPT**

JavaScript (JS) is a [dynamic](http://en.wikipedia.org/wiki/Dynamic_programming_language) computer [programming language](http://en.wikipedia.org/wiki/Programming_language). It is most commonly used as part of [web browsers](http://en.wikipedia.org/wiki/Web_browser), whose implementations allow [client-side scripts](http://en.wikipedia.org/wiki/Client-side_scripting) to [interact with the user](http://en.wikipedia.org/wiki/User_interface), control the browser, communicate [asynchronously](http://en.wikipedia.org/wiki/Ajax_(programming)), and alter the [document content](http://en.wikipedia.org/wiki/Document_Object_Model) that is displayed. It is also being used in server-side programming, game development and the creation of desktop and mobile applications. JavaScript is a [prototype-based](http://en.wikipedia.org/wiki/Prototype-based_programming) [scripting language](http://en.wikipedia.org/wiki/Scripting_language) with [dynamic](http://en.wikipedia.org/wiki/Dynamic_language) typing and has [first-class functions](http://en.wikipedia.org/wiki/First-class_functions). The key design principles that are within JavaScript are taken from the [Self](http://en.wikipedia.org/wiki/Self_(programming_language))  and  [Scheme](http://en.wikipedia.org/wiki/Scheme_(programming_language)" \o "Scheme (programming language)) programming languages. It is also a [multi - paradigm](http://en.wikipedia.org/wiki/Multi-paradigm) language, supporting [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](http://en.wikipedia.org/wiki/Imperative_programming), and [functional](http://en.wikipedia.org/wiki/Functional_programming) programming styles.

**4.8 METHODOLOGY**

**Agile software development** is a group of [software development methods](http://en.wikipedia.org/wiki/Software_development_methodologies) based on [iterative and incremental development](http://en.wikipedia.org/wiki/Iterative_and_incremental_development), where requirements and solutions evolve through collaboration between self-organizing, [cross-functional teams](http://en.wikipedia.org/wiki/Cross-functional_team). It promotes adaptive planning, evolutionary development and delivery, a [time-boxed](http://en.wikipedia.org/wiki/Timeboxing) iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen tight iterations throughout the development cycle.

**CHAPTER 5**

**SYSTEM DESIGN AND LAYOUT**

**5.1 DESCRIPTION OF THE SYSTEM**

Start Function:  FindReverse

Step 1   : Get Input phrase From User  
 Step 1.1: Call Search Dictionary method   
 Step 1.2: Perform Stop words removal  
 Step 2: Load list of stop words from Database  
 Step 2.1:Load it in a Array list  
 Step 2.2:compare each word from phrase with stop word Array List  
 Step 2.3:Remove Stop words  
 Step 2.4: Store keywords in a array

Step 3 :Pass the word to WordNET API   
 Step 4   :Find similar words for Verbs   
 Step 4.1:Pass the words to Servlet  
 Step 4.2:Send Response to HTML Via Ajax  
 Step 4.3 Display Words  
 Stop

**5.2 MODULE DESCRIPTION**

**5.2.1 Extraction of Keywords**

HTML is used as a front end. A array list is used to store the input sentence as individual words. Each word is cross-checked with the stop-word list. Eliminate the words which are stop-words.

**5.2.2 Mapping with the forward dictionary**

The input after removal of stop words is stored in a array list. The array list is sent to the NLP parser to be mapped with the forward dictionary

**5.2.3 Generation of the word list after analysing the remaining words**

Using NLP parser send the phrase to WordNet dictionary. Display the top 15 results received form the Dictionary on the webpage.

**5.3 ARCHITECTURE DIAGRAM**

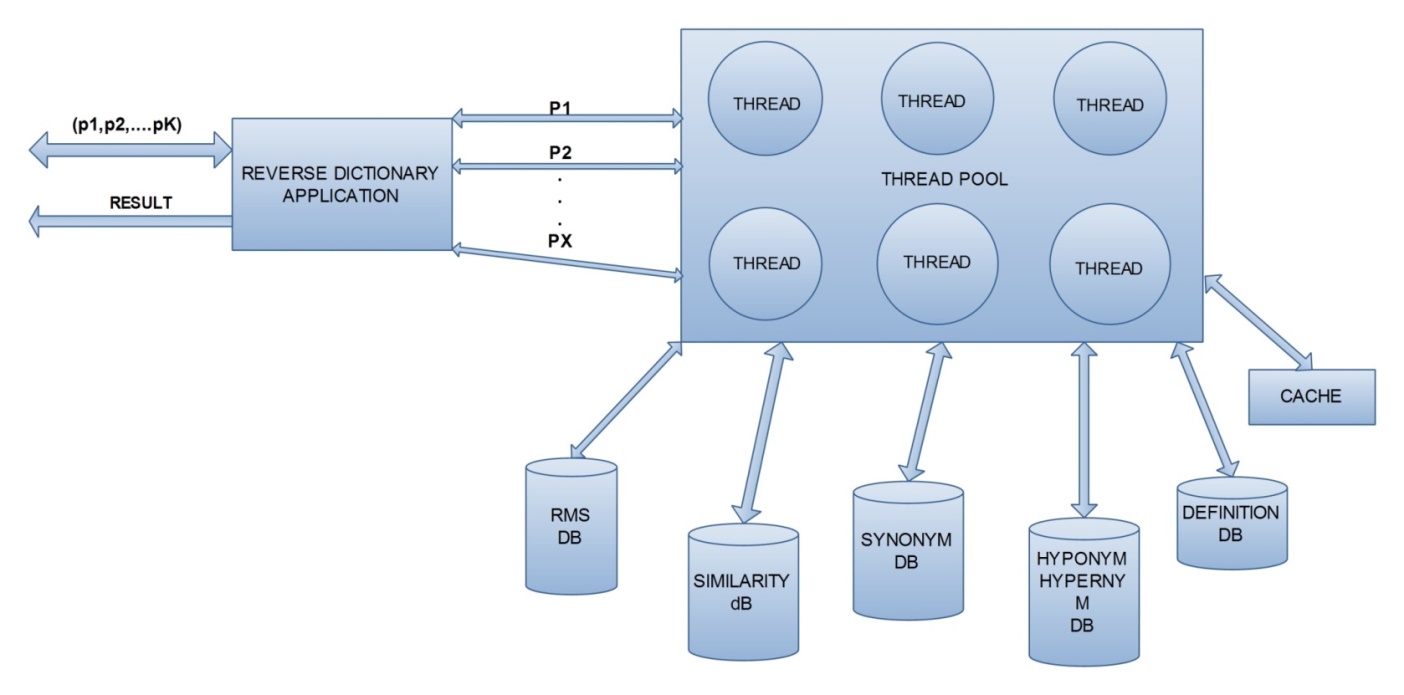


Fig 1:Architecture diagram

**5.4 CLASS DIAGRAM**

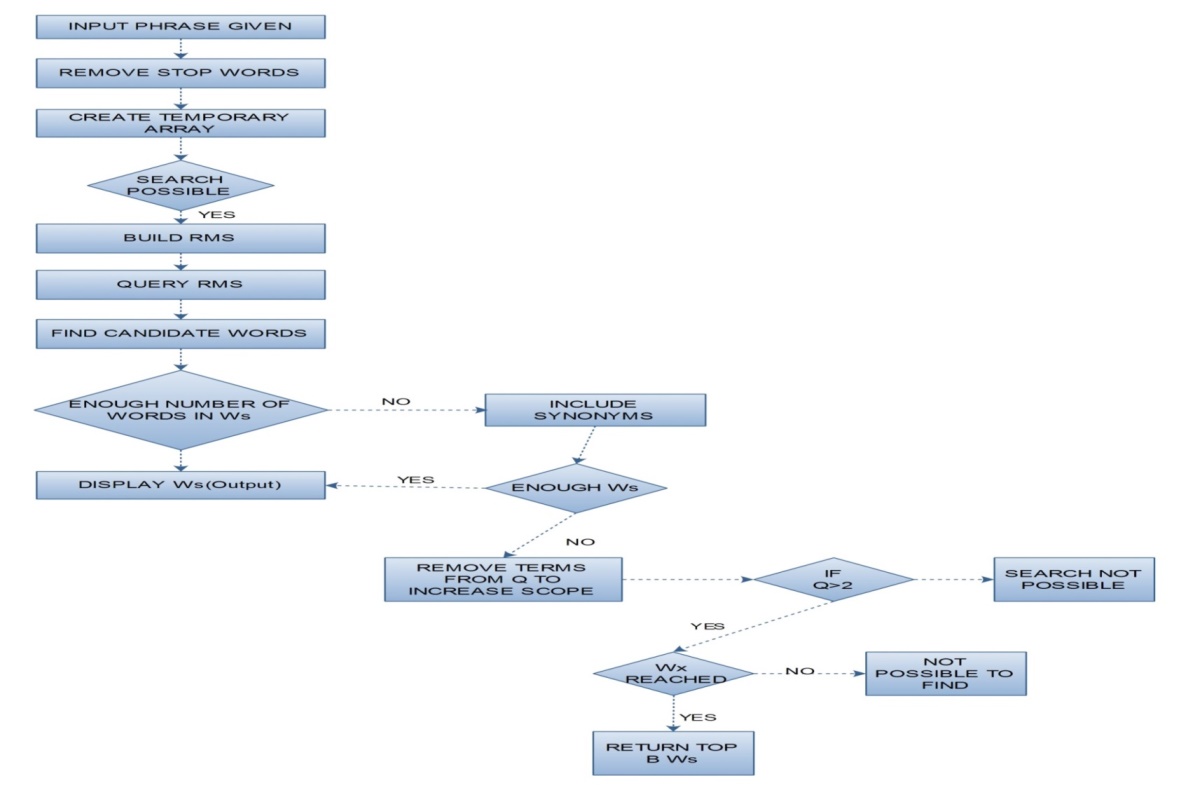


Fig 2: Data flow diagram

**CHAPTER 6**

**TESTING AND IMPLEMENTATION**

**6.1 BLACKBOX TESTING**

|  |  |  |
| --- | --- | --- |
| INPUT | DESIRED OUTPUT | OUTPUT OBTAINED |
| To walk fast | Run | |  | | --- | | run | | march | | festinate | | trot | | pace | | scoot | |
| neither hot nor cold | Lukewarm | |  | | --- | | tepid | | lukewarm | | cool | | indifferent | | neutral | |
| reprimand someone severely | Scold | |  | | --- | | rebuke | | scold | | give | | chastise | | book | |

Table 6.1: Blackbox testing table

**6.2 UNIT TESTING**

In our project, we have two modules:

1)Removing the stopwords

2)Final result from wordnet dictionary

Removing the stopwords:

In this module, we eliminate the stopwords like “is,an,the,of,etc” and then send The remaining words to the wordnet dictionary through the nlp parser. The removal of stopwords can be shown in the console.

|  |  |  |
| --- | --- | --- |
| INPUT | DESIRED RESULT | RESULT OBTAINED |
| To walk fast | Walk, fast | Walk, fast |
| A set of instructions | Set, instructions | Set, instructions |
| To give someone a verbal punishment | Verbal, punishment | Verbal, punishment |

Table 6.2: Stop word removal table

**6.3 VALIDATION TESTING**

**Single character:**

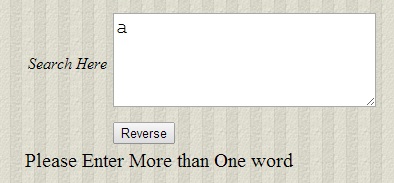


Fig: 6.1: Single character input

**Special character:**

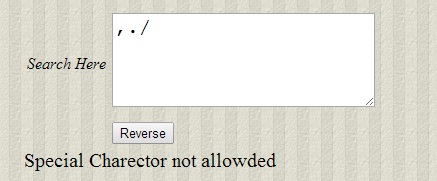


Fig: 6.2 : Special character input

**Empty Text:**

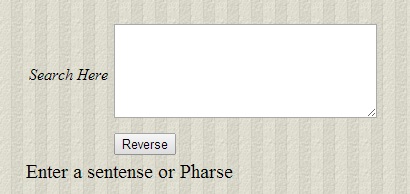


Fig 6.3 : Empty text input

**Valid Input:**

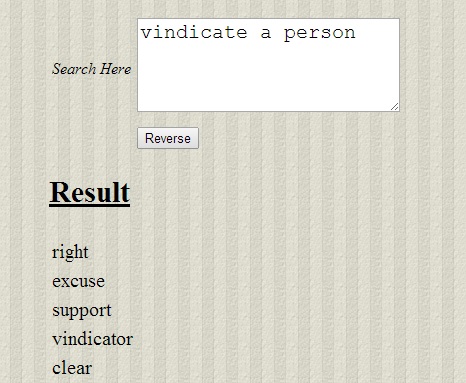


Fig 6.4 : Valid input

**CHAPTER 7**

**CONCLUSION**

Initially when we gave a sentence, we weren’t able to get the required output .Thus We had to eliminate the stop words and then using the NLP parser we mapped it to the WordNet dictionary. The WordNet dictionary checks the given input to its database and then sent the results back. These results are then pasted in the webpage. Building up an actual dictionary takes lot of years and even after that we aren’t sure if we got all the words in the dictionary. Thus we had to map it to a WordNet dictionary database. Using nlp parsing, When we sent the candidate words to it, it gave us a lot of tags. Thus we took only the required tags like verb, adjective, adverb, etc and removed the other tags like noun,pronoun,etc.After the elimination, using simple java logic, the software was created.

**CHAPTER 8**

**FUTURE ENHANCEMENT**

One main enhancement can be the use of speech. That is, the software should be able to recognize the speech of the user and take the input by listening to the voice of the person. Our project is built for computers and laptops alone. A future enhancement can be to make this project available in smartphones, Tablets, iPads, etc. The project can be done to support multiple languages as well. If there are any spelling mistakes in the input, the software must be able to first find the mistake and then suggest possible corrections to the user before sending the input to the software. When the user clicks on a particular word, the software can pronounce the word so that it can be helpful for the user to know the exact pronunciation of the word. The output can be made specific to a particular desire of the user.It can be upgraded to other languages a well.

**CHAPTER 9**

**REFERENCES:**

1. Ryan Shaw,Anindya Datta,Debra Vander\Meer,Kaushik Dutta “Building a Scalable Database-Driven Reverse Dictionary” , 2013
2. R.Baeza-Yates nd B.Ribero-Neto,Modern Information Retrieval ACM Press,2011
3. G.Miller,C.Fellbaum, R.Tengi,P.Wakefield and H.Langone “Wordnet Lexical Database” http://wordnet.princeton.edu/wordnet/download/,2009.
4. T. Dao and T. Simpson, “Measuring Similarity between Sentences, ”http://opensvn.csie.org/WordNetDotNet/trunk/Projects/Thanh/Paper/ wordNetDotNet \_ Semantic\_Similarity.pdf, 2009.
5. Dictionary.com, LLC, “Reverse Dictionary,”http://dictionary.reference.com/

reverse, 2009.

[6] J.Ponte and W.Croft, “ A language modelling approach to information retrieval”

Proc . 21st Ann.Int’l ACM SIGIR Conf. Reserch and development in Information

Rerieval,pp. 275-281,2008

[7] J.Lafferty and C.Zhai “Document Language Models, Query Models, and Risk

Minimization for Information Retrieval “ Proc 24th Ann.int’l ACM SIGIR

Conf.Research and development in information retrieval pg. 111-119 ,2007

**APPENDIX-1**

**Dict.java:**

**package** com.database;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**import** org.apache.commons.lang3.StringUtils;

**import** org.jsoup.Jsoup;

**import** org.jsoup.nodes.Document;

**import** org.jsoup.nodes.Element;

**import** org.jsoup.select.Elements;

**import** com.reverse.QuickStart;

**public** **class** Dict {

**public** **static** **void** main(String[] args) {

**try**

{

String datatoSearch = "Happening quickly";

Dict g = **new** Dict();

g.getReverseWords(datatoSearch);

//g.getSoup();

}**catch**(Exception e){

System.out.println(e);

}

}

**public** List<String> getReverseWords(String pharse) {

List<String> myReverseWords = **new** ArrayList<String>();

String url1 = "";

//http://www.onelook.com/?w=\*&loc=revfp2&clue=happy

String test = "https://www.google.co.in/";

//http://www.onelook.com/?w=\*&loc=revfp2&clue=walking+fast

//String url = "http://www.ebay.in/";

String url = "http://www.onelook.com/?w=\*&loc=revfp2&clue=";

String answerHTMl="";

String datatoSearch = "Happening quickly";

QuickStart quick = **new** QuickStart();

**try** {

String furl = url+pharse;

System.out.println("Pirnt URL ==" + furl);

answerHTMl=quick.getcallRestService(furl);

System.out.println(url+pharse);

// Document doc = Jsoup.connect(url+pharse).timeout(0).get();

Document doc = Jsoup.parse(answerHTMl);

Elements links = doc.select("a[href]");

**for** (Element link : links) {

url1 += link.ownText() + "\n";

}

**int** indexStart = url1

.indexOf("<< Return to the reverse dictionary page");

System.out.println(indexStart);

String title = StringUtils.substringBetween(url1,

"<< Return to the reverse dictionary page",

"<< Return to the reverse dictionary page");

System.out.println(myReverseWords.size());

String[] tile = title.split("\n");

System.out.println("\*\*\*\*\*\*\*\*\*");

String tableCreation="<table>";

**for** (**int** i = 0; i < tile.length; i++) {

**if**(tile[i]!= " "){

myReverseWords.add(tile[i]);

tableCreation+="<tr><td>"+tile[i]+"</td></tr>";

}

tableCreation+="</table>";

}

System.out.println(myReverseWords.size());

myReverseWords.remove("nouns");

myReverseWords.remove("Next page >>");

myReverseWords.remove("More info on wildcards");

myReverseWords.remove("Home");

myReverseWords.remove("Reverse Dictionary");

myReverseWords.remove("Customize");

myReverseWords.remove("Browse Dictionaries");

myReverseWords.remove("Privacy");

myReverseWords.remove("Blog");

myReverseWords.remove("Help");

myReverseWords.remove("Blog");

myReverseWords.remove("Link to us");

myReverseWords.remove("Word of the Day");

myReverseWords.remove("adjectives");

myReverseWords.remove("verbs");

myReverseWords.remove("adverbs");

myReverseWords.remove("adjectives");

myReverseWords.remove(" ");

System.out.println(myReverseWords.size());

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println(tableCreation);

} **catch** (Exception e) {

e.printStackTrace();

}

**return** myReverseWords;

}

**public** **void** getSoup(){

Document doc;

**try** {

String url2 = http://www.onelook.com/?w=\*&loc=revfp2&clue=happy?";

String url = java.net.URLEncoder.encode(url2, "UTF-8");

// need http protocol

String link1 = "?w=\*&loc=revfp&clue=happy";

String u = "http://www.onelook.com/";

System.out.println(u+link1);

doc = Jsoup.connect("http://www.onelook.com").get();

// get page title

String title = doc.title();

System.out.println("title : " + title);

// get all links

Elements links = doc.select("a[href]");

**for** (Element link : links) {

// get the value from href attribute

System.out.println("\nlink : " + link.attr("href"));

System.out.println("text : " + link.text());

}

} **catch** (IOException e) {

e.printStackTrace();

}

}}

**ConsoleReader.java:**

**package** com.reverse;

**import** java.io.BufferedReader;

**import** java.io.InputStream;

**import** java.io.InputStreamReader;

**import** java.io.IOException;

/\*\*

A class to read strings and numbers from an input stream.

This class is suitable for beginning Java programmers.

It constructs the necessary buffered reader,

handles I/O exceptions, and converts strings to numbers.

\*/

**public** **class** ConsoleReader

{ /\*\*

Constructs a console reader from an input stream

such as System.in

**@param** inStream an input stream

\*/

**public** ConsoleReader(InputStream inStream)

{ reader = **new** BufferedReader

(**new** InputStreamReader(inStream));

}

/\*\*

Reads a line of input and converts it into an integer.

The input line must contain nothing but an integer.

Not even added white space is allowed.

**@return** the integer that the user typed

\*/

**public** **int** readInt()

{ String inputString = readLine();

**int** n = Integer.parseInt(inputString);

**return** n;

}

/\*\*

Reads a line of input and converts it into a floating-

point number. The input line must contain nothing but

a nunber. Not even added white space is allowed.

**@return** the number that the user typed

\*/

**public** **double** readDouble()

{ String inputString = readLine();

**double** x = Double.parseDouble(inputString);

**return** x;

}

/\*\*

Reads a line of input. In the (unlikely) event

of an IOException, the program terminates.

**@return** the line of input that the user typed, null

at the end of input

\*/

**public** String readLine()

{ String inputLine = "";

**try**

{ inputLine = reader.readLine();

}

**catch**(IOException e)

{ System.out.println(e);

System.exit(1);

}

**return** inputLine;

}

**private** BufferedReader reader;

}

**ParserDemo.java:**

**package** com.reverse;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.StringReader;

**import** java.util.ArrayList;

**import** java.util.Collection;

**import** java.util.Iterator;

**import** java.util.List;

**import** edu.stanford.nlp.ling.CoreLabel;

**import** edu.stanford.nlp.ling.HasWord;

**import** edu.stanford.nlp.ling.Label;

**import** edu.stanford.nlp.ling.TaggedWord;

**import** edu.stanford.nlp.parser.lexparser.LexicalizedParser;

**import** edu.stanford.nlp.process.Tokenizer;

**import** edu.stanford.nlp.trees.GrammaticalStructure;

**import** edu.stanford.nlp.trees.GrammaticalStructureFactory;

**import** edu.stanford.nlp.trees.Tree;

**import** edu.stanford.nlp.trees.TreebankLanguagePack;

**import** edu.stanford.nlp.trees.TypedDependency;

**public** **class** ParserDemo1 {

/\*\* Usage: ParserDemo2 [[grammar] textFile] \*/

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

ParserDemo1 demo = **new** ParserDemo1();

//demo.stripWords();

String sent3 = "This is one last test run!";

// lp.parse(sent3).pennPrint();

demo.stripWords(sent3);

}

**public** List<String> stripWords(String pharse) {

List<String> nounparsed = **new** ArrayList<String>();

String outputString = **null**;

String grammar = "edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz";

String[] options = { "-maxLength", "80", "-retainTmpSubcategories" };

LexicalizedParser lp = LexicalizedParser.loadModel(grammar, options);

TreebankLanguagePack tlp = lp.getOp().langpack();

GrammaticalStructureFactory gsf = tlp.grammaticalStructureFactory();

Iterable<List<? **extends** HasWord>> sentences;

BufferedReader br = **null**;

StringBuilder sb = **new** StringBuilder();

String s = pharse;;

System.out.println("The sentence is == " + s);

String[] sent = **new** String[] { s };

System.out.println("count == " + sent.length);

/\*

\* List<HasWord> sentence = new ArrayList<HasWord>(); for (String word :

\* sent) { System.out.println("word == "+word); sentence.add(new

\* Word(word)); }

\*/

// String sent2 = ("This is a slightly longer and more complex " +

// "sentence requiring tokenization.");

String sent2 = s;

Tokenizer<? **extends** HasWord> toke = tlp.getTokenizerFactory()

.getTokenizer(**new** StringReader(sent2));

List<? **extends** HasWord> sentence2 = toke.tokenize();

List<List<? **extends** HasWord>> tmp = **new** ArrayList<List<? **extends** HasWord>>();

// tmp.add(sentence);

tmp.add(sentence2);

sentences = tmp;

// }

**for** (List<? **extends** HasWord> sentence1 : sentences) {

Tree parse = lp.parse(sentence1);

parse.pennPrint();// copy this to notepad....

String toFile = parse.pennString();

System.out.println();

GrammaticalStructure gs = gsf.newGrammaticalStructure(parse);

Collection<TypedDependency> tdl = gs.allTypedDependencies();

Iterator<TypedDependency> iter = tdl.iterator();

**while** (iter.hasNext()) {

TypedDependency td = iter.next();

// if(td.reln().toString().equals("conj")){

System.out.println(td);

}

// System.out.println("tdl == "+tdl);

System.out.println();

System.out.println("The words of the sentence:");

**for** (Label lab : parse.yield()) {

**if** (lab **instanceof** CoreLabel) {

// System.out.println(((CoreLabel) lab).toString("{map}"));

// System.out.println("core label...");

} **else** {

// System.out.println(lab);

// System.out.println("not core label");

}

}

System.out.println();

// System.out.println("\*\*\*\*\* "+parse.taggedYield());

List<TaggedWord> listTagged = parse.taggedYield();

TaggedWord[] taggedWords = **new** TaggedWord[listTagged.size()];

TaggedWord[] taggedWord = parse.taggedYield().toArray(taggedWords);

**int** markWordIndex = 0;

String reservedWords = "";

**for** (**int** i = 0; i < taggedWord.length; i++) {

TaggedWord tagWord = taggedWord[i];

System.out.println(tagWord.tag() + "==" + tagWord.word());

String tag = tagWord.tag();

**if** (tag.equals("VB") ||tag.equals("RB") || tag.equals("VBD")|| tag.equals("VBG")||

tag.equals("VBN")|| tag.equals("VBP")|| tag.equals("VBZ")|| tag.equals("JJ")|| tag.equals("JJR")|| tag.equals("JJS")) {

markWordIndex = i;

nounparsed.add(tagWord.word());

//break;

}

reservedWords += tagWord.word() + " ";

System.out.println(""+reservedWords);

}

// System.out.println(markWordIndex);

reservedWords = reservedWords.substring(0,

reservedWords.length() - 1);

// System.out.println(reservedWords);

**int** index = reservedWords.lastIndexOf(" ");

System.out.println(index);

String newTemp = "";

**if** (index != -1) {

newTemp = reservedWords.substring(0, index);

}System.out.println("\*");

System.out.println(newTemp);

String nextSen = "";

String remaining = "";

**for** (**int** i = 0; i < taggedWord.length; i++) {

TaggedWord tagWord = taggedWord[i];

**if** (i > markWordIndex) {

nextSen += tagWord.word() + " ";

}

**if** (i > markWordIndex + 1) {

remaining += tagWord.word() + " ";

}

}

outputString = reservedWords;

}

System.err.println("NOUN");

System.err.println(nounparsed);

**return** nounparsed;

}

}

**QuickStart.java:**

**package** com.reverse;

**import** java.io.IOException;

**import** org.apache.http.HttpEntity;

**import** org.apache.http.HttpResponse;

**import** org.apache.http.client.ClientProtocolException;

**import** org.apache.http.client.ResponseHandler;

**import** org.apache.http.client.methods.HttpGet;

**import** org.apache.http.entity.ByteArrayEntity;

**import** org.apache.http.impl.client.CloseableHttpClient;

**import** org.apache.http.impl.client.HttpClients;

**import** org.apache.http.util.EntityUtils;

**public** **class** QuickStart {

**public** **static** **void** callRestService() **throws** Exception {

CloseableHttpClient httpclient = HttpClients.createDefault();

**try** {

HttpGet httpget = **new** HttpGet(

"http://www.onelook.com/?w=\*&loc=revfp2&clue=running+fast");

// HttpPost post = new

// HttpPost("https://localhost:8443/v1/Hello/echoCus");

System.out.println("executing request " + httpget.getURI());

// httpget.addHeader("Accept", "application/json");

// post.setEntity(new ByteArrayEntity(new

// String("<Word><wordId>1000</wordId><wordString>from java client</wordString></Word>").getBytes()));

// Create a custom response handler

ResponseHandler<String> responseHandler = **new** ResponseHandler<String>() {

**public** String handleResponse(**final** HttpResponse response)

**throws** ClientProtocolException, IOException {

**int** status = response.getStatusLine().getStatusCode();

**if** (status >= 200 && status < 300) {

HttpEntity entity = response.getEntity();

**return** entity != **null** ? EntityUtils.toString(entity)

: **null**;

} **else** {

**throw** **new** ClientProtocolException(

"Unexpected response status: " + status);

}

}

};

String responseBody = httpclient.execute(httpget, responseHandler);

System.out.println("----------------------------------------");

System.out.println(responseBody);

System.out.println("----------------------------------------");

} **finally** {

httpclient.close();

}

}

**public** String getcallRestService(String url) **throws** Exception {

String responseBody = "";

CloseableHttpClient httpclient = HttpClients.createDefault();

**try** {

HttpGet httpget = **new** HttpGet(url);

System.out.println("executing request " + httpget.getURI());

ResponseHandler<String> responseHandler = **new** ResponseHandler<String>() {

**public** String handleResponse(**final** HttpResponse response)

**throws** ClientProtocolException, IOException {

**int** status = response.getStatusLine().getStatusCode();

**if** (status >= 200 && status < 300) {

HttpEntity entity = response.getEntity();

**return** entity != **null** ? EntityUtils.toString(entity)

: **null**;

} **else** {

**throw** **new** ClientProtocolException(

"Unexpected response status: " + status);

}

}

};

responseBody = httpclient.execute(httpget, responseHandler);

System.out.println("----------------------------------------");

// System.out.println(responseBody);

System.out.println("----------------------------------------");

} **finally** {

httpclient.close();

}

**return** responseBody;

}

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

**try** {

callRestService();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**ReverseDict.java:**

**package** com.reverse;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** com.database.Dict;

/\*\*

\* Servlet implementation class ReverseDict

\*/

@WebServlet("/ReverseDict")

**public** **class** ReverseDict **extends** HttpServlet {

**private** **static** **final** **long** serialVersionUID = 1L;

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** ReverseDict() {

**super**();

// **TODO** Auto-generated constructor stub

}

/\*\*

\* **@see** HttpServlet#doGet(HttpServletRequest request, HttpServletResponse

\* response)

\*/

**protected** **void** doGet(HttpServletRequest request,

HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

}

/\*\*

\* **@see** HttpServlet#doPost(HttpServletRequest request, HttpServletResponse

\* response)

\*/

**protected** **void** doPost(HttpServletRequest request,

HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

String pharse = request.getParameter("pharse");

Dict db = **new** Dict();

List<String> reverseWords = **new** ArrayList<>();

List<String> userwords = **new** ArrayList<>();

String splitUser[] = pharse.split(" ");

**for** (**int** i = 0; i < splitUser.length; i++) {

userwords.add(splitUser[i]);

}

String keyPharseWithSign="";

**for**(**int** j=0;j<userwords.size();j++){

keyPharseWithSign+=userwords.get(j)+"+";

}

System.out.println(keyPharseWithSign);

reverseWords = db.getReverseWords(keyPharseWithSign);

System.out.println(pharse);

// remove stopwords !

String[] stopwords = { "a", "about", "above", "above", "across",

"after", "afterwards", "again", "against", "all", "almost",

"alone", "along", "already", "also", "although", "always",

"am", "among", "amongst", "amoungst", "amount", "an", "and",

"another", "any", "anyhow", "anyone", "anything", "anyway",

"anywhere", "are", "around", "as", "at", "back", "be",

"became", "because", "become", "becomes", "becoming", "been",

"before", "beforehand", "behind", "being", "below", "beside",

"besides", "between", "beyond", "bill", "both", "bottom",

"but", "by", "call", "can", "cannot", "cant", "co", "con",

"could", "couldnt", "cry", "de", "describe", "detail", "do",

"done", "down", "due", "during", "each", "eg", "eight",

"either", "eleven", "else", "elsewhere", "empty", "enough",

"etc", "even", "ever", "every", "everyone", "everything",

"everywhere", "except", "few", "fifteen", "fify", "fill",

"find", "fire", "first", "five", "for", "former", "formerly",

"forty", "found", "four", "from", "front", "full", "further",

"get", "give", "go", "had", "has", "hasnt", "santhosh", "have",

"he", "hence", "her", "here", "hereafter", "hereby", "herein",

"hereupon", "hers", "herself", "him", "himself", "his", "how",

"however", "hundred", "ie", "if", "in", "inc", "indeed",

"interest", "into", "is", "it", "its", "itself", "keep",

"last", "latter", "latterly", "least", "less", "ltd", "made",

"many", "may", "me", "meanwhile", "might", "mill", "mine",

"more", "moreover", "most", "mostly", "move", "much", "must",

"my", "myself", "name", "namely", "neither", "never",

"nevertheless", "next", "nine", "no", "nobody", "none",

"noone", "nor", "not", "nothing", "now", "nowhere", "of",

"off", "often", "on", "once", "one", "only", "onto", "or",

"other", "others", "otherwise", "our", "ours", "ourselves",

"out", "over", "own", "part", "per", "perhaps", "please",

"put", "rather", "re", "same", "see", "seem", "seemed",

"seeming", "seems", "serious", "several", "she", "should",

"show", "side", "since", "sincere", "six", "sixty", "so",

"some", "somehow", "someone", "something", "sometime",

"sometimes", "somewhere", "still", "such", "system", "take",

"ten", "than", "that", "the", "their", "them", "themselves",

"then", "thence", "there", "thereafter", "thereby",

"therefore", "therein", "thereupon", "these", "they", "thickv",

"thin", "third", "this", "those", "though", "three", "through",

"throughout", "thru", "thus", "to", "together", "too", "top",

"toward", "towards", "twelve", "twenty", "two", "un", "under",

"until", "up", "upon", "us", "very", "via", "was", "we",

"well", "were", "what", "whatever", "when", "whence",

"whenever", "where", "whereafter", "whereas", "whereby",

"wherein", "whereupon", "wherever", "whether", "which",

"while", "whither", "who", "whoever", "whole", "whom", "whose",

"why", "will", "with", "within", "without", "would", "yet",

"you", "your", "yours", "yourself", "yourselves", "1", "2",

"3", "4", "5", "6", "7", "8", "9", "10", "1.", "2.", "3.",

"4.", "5.", "6.", "11", "7.", "8.", "9.", "12", "13", "14",

"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L",

"M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X",

"Y", "Z", "terms", "CONDITIONS", "conditions", "values",

"interested.", "care", "sure", ".", "!", "@", "#", "$", "%",

"^", "&", "\*", "(", ")", "{", "}", "[", "]", ":", ";", ",",

"<", ".", ">", "/", "?", "\_", "-", "+", "=", "a", "b", "c",

"d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o",

"p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z",

"contact", "grounds", "buyers", "tried", "said,", "plan",

"value", "principle.", "forces", "sent:", "is,", "was", "like",

"discussion", "tmus", "diffrent.", "layout", "area.", "thanks",

"thankyou", "hello", "bye", "rise", "fell", "fall", "psqft.",

"http://", "km", "miles", "brought" };

ArrayList<String> holdWords = **new** ArrayList<String>();// word in db

ArrayList<String> wordsDb = **new** ArrayList<String>();// word in db

String splitWord[] = pharse.split(" ");

List<String> output = **new** ArrayList<String>();

List<String> inputWordNet = **new** ArrayList<String>();

**for** (**int** i = 0; i < splitWord.length; i++) {

holdWords.add(splitWord[i]);

}

**for** (**int** k = 0; k < stopwords.length; k++) {

wordsDb.add(stopwords[k]);

}

**for** (**int** j = 0; j < holdWords.size(); j++) {

**if** (wordsDb.contains(holdWords.get(j))) {

System.out.println("Equals >>" + holdWords.get(j));

} **else** {

output.add(holdWords.get(j));

System.out.println("Not Equals >>" + holdWords.get(j));

}

}

// print keywords ...

String keyword = "";

**for** (**int** g = 0; g < output.size(); g++) {

keyword += output.get(g) + " ";

}

System.out.println("\*\*\*\*\*\*\*\*\*\*KEYWORDS\*\*\*\*\*\*\*\*\*\*\*");

System.out.println(keyword);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//send to NLP parser to find noun and verb!

ParserDemo1 demo = **new** ParserDemo1();

List<String> nouns = **new** ArrayList<>();

nouns=demo.stripWords(keyword);

System.out.println("\*\*\*\*\*\*\*\*\*NOUN\*\*\*\*\*\*\*\*\*");

System.out.println(nouns);

wordNet wordReverse = **new** wordNet();

//send keywords to WordNet !

String outputResponce="";

**for**(**int** t=0;t<nouns.size();t++){

outputResponce+= wordReverse.getReverse(nouns.get(t).toString());

}

System.out.println("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$");

System.out.println(outputResponce);

System.out.println("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$");

// get only first few words from arrayList

List<String> subList = reverseWords.subList(0, 10);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

String tableCreation="<table>";

System.out.println("Size :"+subList.size());

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

tableCreation+="<tr><td>"+subList.get(i)+"</td></tr>";

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println(subList.toString());

List<String> removedSpace = **new** ArrayList<>();

Iterator iter = subList.iterator();

**while** (iter.hasNext()) {

String object = (String) iter.next();

**if**(!object.equals(" ")){

// System.out.println("ss = "+object);

}

**if**(object!=" "){

// System.out.println("ss = "+object);

}

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

response.getWriter().write(tableCreation);

}

}

**SensesLookup.java:**

**package** com.reverse;

**import** rita.wordnet.RiWordnet;

**public** **class** SensesLookup {

**public** **static** **void** main(String[] args) {

ConsoleReader reader = **new** ConsoleReader(System.in);

// Would pass in a PApplet normally, but we don't need to here

RiWordnet wordnet = **new** RiWordnet(**null**);

**while** (**true**) {

// Grab a word from the user

System.out.println("\n\nPlease enter a word (q to quit): ");

System.out.print("% ");

String input = reader.readLine();

// Check to see if we should quit

**if** (input.equals("q")) {

**break**;

}

// First we should make sure it's a valid wordnet word

**if** (wordnet.exists(input)) {

// Get parts of speech as an array

String[] pos = wordnet.getPos(input);

System.out.println("\nPossible parts of speech: ");

**for** (**int** i = 0; i < pos.length; i++) {

System.out.println(i + ": " + pos[i]);

}

// Ask user to select a part of speech

System.out.println("\nPlease select a part of speech (enter index #) ");

System.out.print("% ");

**int** selection = reader.readInt();

// Get all of the IDs associated with possible "senses" (synsets) of that word

// we will use these IDs as a means for getting data about that sense

**int**[] ids = wordnet.getSenseIds(input, pos[selection]);

// Display information about the synsets that word belongs to

System.out.println("\nHere are all the possible senses of " + input + " with pos: " + pos[selection]);

System.out.println("==========================================");

**for** (**int** i = 0; i < ids.length; i++) {

// Sense ID #

System.out.println("Sense: " + ids[i]);

String description = wordnet.getDescription(ids[i]);

// Sense Description (definition)

System.out.println("Description: " + description);

// All words that belong to this synset

String[] words = wordnet.getSynset(ids[i]);

**if** (words != **null**) {

System.out.print("Synset: ");

**for** (**int** j = 0; j < words.length; j++) System.out.print(words[j] + " ");

}

System.out.println("\n-------------------------");

}

} **else** {

System.out.println("That word is not in wordnet.");

}

}

System.out.println("Quitting. . .");

}

}

**WordNet.java:**

**package** com.reverse;

**import** rita.wordnet.RiWordnet;

**public** **class** wordNet {

RiWordnet wordnet = **new** RiWordnet(**null**);

/\*\*

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

// Would pass in a PApplet normally, but we don't need to here

wordNet n = **new** wordNet();

n.getReverse("lukewarm");

}

**public** String getReverse(String word){

String reverse="";

//word = "Moderate ";

String[] poss = wordnet.getPos(word);

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

System.out.println("\n\nSynonyms for " + word + " (pos: " + poss[j] + ")");

String[] synonyms = wordnet.getAllSynonyms(word,poss[j],10);

**for** (**int** i = 0; i < synonyms.length; i++) {

System.out.println(synonyms[i]);

reverse+=synonyms[i]+",";

}

}

System.out.println(reverse);

**return** reverse;

}}

**WordNetDemo.java:**

package com.reverse;import rita.wordnet.RiWordnet;public class WordNetDemo { public static void main(String[] args) { // Would pass in a PApplet normally, but we don't need to here RiWordnet wordnet = new RiWordnet(null); // Demo finding parts of speech String word = "run"; System.out.println("\nFinding parts of speech for " + word + "."); String[] partsofspeech = wordnet.getPos(word); for (int i = 0; i < partsofspeech.length; i++) { System.out.println(partsofspeech[i]); } word = "eat"; String pos = wordnet.getBestPos(word); System.out.println("\n\nDefinitions for " + word + ":"); // Get an array of glosses for a word String[] glosses = wordnet.getAllGlosses(word, pos); // Display all definitions for (int i = 0; i < glosses.length; i++) { System.out.println(glosses[i]); } // Demo finding a list of related words (synonyms) word = "Moderate "; String[] poss = wordnet.getPos(word); for (int j = 0; j < poss.length; j++) { System.out.println("\n\nSynonyms for " + word + " (pos: " + poss[j] + ")"); String[] synonyms = wordnet.getAllSynonyms(word,poss[j],10); for (int i = 0; i < synonyms.length; i++) { System.out.println(synonyms[i]); } } // Demo finding a list of related words // X is Hypernym of Y if every Y is of type X // Hyponym is the inverse word = "chair"; pos = wordnet.getBestPos(word); System.out.println("\n\nHyponyms for " + word + ":"); String[] hyponyms = wordnet.getAllHyponyms(word, pos); for (int i = 0; i < hyponyms.length; i++) { System.out.println(hyponyms[i]); } System.out.println("\n\nHypernyms for " + word + ":"); String[] hypernyms = wordnet.getAllHypernyms(word, pos); for (int i = 0; i < hypernyms.length; i++) { System.out.println(hypernyms[i]); } String start = "dog"; String end = "giraffe"; pos = wordnet.getBestPos(start); // Wordnet can find relationships between words System.out.println("\n\nRelationship between: " + start + " and " + end); float dist = wordnet.getDistance(start,end,pos); String[] parents = wordnet.getCommonParents(start, end, pos); System.out.println(start + " and " + end + " are related by a distance of: " + dist); // These words have common parents (hyponyms in this case) System.out.println("Common parents: "); if (parents != null) { for (int i = 0; i < parents.length; i++) { System.out.println(parents[i]); }} }}

**JAVASCRIPT FUNCTION:**

**Reverse.js:**

**function** validate()

{

**var** s=document.getElementById("pharse").value;

**if**(s==**null** || s=="")

{

document.getElementById("info").innerHTML="Enter a sentense or Pharse ";

}

**else** **if**(/^[a-zA-Z0-9- ]\*$/.test(s) == **false**)

{

document.getElementById("info").innerHTML="Special Charector not allowded";

}

**else** **if**(s.length==1)

{

document.getElementById("info").innerHTML="Please Enter More than One word";

}

**else**

{

document.getElementById("info1").style.display='block';

reverseMe(s);

}

}

**function** reverseMe(s)

{

xmlHttp = GetXmlHttpObject();

**var** url = "/ReverseDictionaryPro/ReverseDict?pharse=" + s;

xmlHttp.onreadystatechange = getReplay;

xmlHttp.open("POST", url, **true**);

xmlHttp.send(**null**);

}

**function** getReplay()

{

**if** (xmlHttp.readyState == 4 || xmlHttp.readyState == "complete") {

**var** showdata = xmlHttp.responseText;

//alert(showdata);

//if(showdata=="failed")

//{

//document.getElementById("info").innerHTML="no data";

//}

//else

//{

document.getElementById("info1").style.display='none';

**var** result="<h2><b><u>Result</u></b></h2>";

document.getElementById("info").innerHTML=result+showdata;

//}

}}

**function** GetXmlHttpObject() {

**var** xmlHttp = **null**;

**try** {

xmlHttp = **new** XMLHttpRequest();

} **catch** (e) {

**try** {

xmlHttp = **new** ActiveXObject("Msxml2.XMLHTTP");

} **catch** (e) {

xmlHttp = **new** ActiveXObject("Microsoft.XMLHTTP");

}

}

**return** xmlHttp;

}

**HTML PAGES:-**

**About.html:**

<!DOCTYPE html>

<!-- Website template by freewebsitetemplates.com -->

<head>

<title>About - Retro Diner Website Template</title>

<meta charset="utf-8">

<link href="css/style.css" rel="stylesheet" type="text/css">

</head>

<body>

<div id="header">

<div>

<a href="index.html"><img class="logo" src="images/logo.jpg" width="513" height="84" alt="" title=""></a>

<a href="index.html"></a>

<ul class="navigation">

<li>

<a href="index.html">Home</a>

</li>

<li>

<a class="active" href="about.html">About</a>

</li>

<li>

<a href="contact.html">Contact</a>

</li>

</ul>

</div>

</div>

<div id="body">

<div id="content">

<div>

<div>

<h1>About</h1>

<ul>

<li>

<h2><a href="about.html">Wordnet and NLP Process</a></h2>

<p>

The Java API for WordNet Searching (JAWS) is an API that provides Java applications with the ability to retrieve data from the WordNet database. It is a simple and fast API that is compatible with both the 2.1 and 3.0 versions of the WordNet database files.

</p>

</li>

<li>

<h2><a href="about.html">NLP</a></h2>

<p>

Natural language processing is the technology for dealing with our most ubiquitous product: human language, as it appears in emails, web pages, tweets, product descriptions, newspaper stories, social media, and scientific articles, in thousands of languages and varieties.

</p>

</li>

</ul>

</div>

</div>

</div>

</div>

<div id="footer">

<div>

<ul>

<li class="first">

<h2>Srm University</h2>

<h3></h3>

</li>

<li>

<a href="index.html"></a>

<ul class="navigation">

<li>

<a href="index.html">Home</a>

</li>

<li>

<a href="about.html">About It</a>

</li>

<li>

<a href="contact.html">Contact Us</a>

</li>

</ul>

<span>&copy; 2014 Reversedict.com</span>

</li>

</ul>

</div>

</div>

</body>

</html>

**blog.html:**

<!DOCTYPE html>

<!-- Website template by freewebsitetemplates.com -->

<head>

<title>Blog - Retro Diner</title>

<meta charset="utf-8">

<link href="css/style.css" rel="stylesheet" type="text/css">

</head>

<body>

<div id="header">

<div>

<a href="index.html"><img class="logo" src="images/logo.png" width="513" height="84" alt="" title=""></a>

<a href="index.html"><img src="images/waitress.png" width="332" height="205" alt="" title=""></a>

<ul class="navigation">

<li>

<a href="index.html">Home</a>

</li>

<li>

<a href="about.html">About IT</a>

</li>

<li>

<a href="contact.html">Contact</a>

</li>

</ul>

</div>

</div>

<div id="body">

<div id="section">

<div class="section">

<div>

<div> <a href=""><img src="images/breakfast3.jpg" width="169" height="163" alt=""></a>

<h2><a href="blog.html">All YOU CAN EAT</a></h2>

<h3>Posted by <a href="#">Sheila</a> in</h3>

<a href="blog.html"><span>Promos,</span></a> <a href="blog.html"><span>Weekends,</span></a> <a href="blog.html"><span>Breakfast</span></a>

<p>

This website template has been designed by Free Website Templates for you, for free. You can replace all this text with your own text. You can remove any link to our website from this website template, you're free to use this website template without linking back to us.

</p>

<a href="blog.html" class="price">Jan <span>31st</span></a> </div>

</div>

</div>

<div class="section">

<div>

<div> <a href=""><img src="images/shakes2.jpg" width="169" height="163" alt=""></a>

<h2><a href="blog.html">love day</a></h2>

<h3>Posted by Sheila in</h3>

<a href="blog.html"><span>Promos,</span></a> <a href="blog.html"><span>Weekends,</span></a> <a href="blog.html"><span>Breakfast</span></a>

<p>

This website template has been designed by Free Website Templates for you, for free. You can replace all this text with your own text. You can remove any link to our website from this website template, you're free to use this website template without linking back to us.

</p>

<a href="blog.html" class="price">Feb <span>1st</span></a> </div>

</div>

</div>

</div>

<div class="sidebar">

<h1>Recent Posts</h1>

<ul>

<li class="first">

<a href="blog.html">All You Can Eat Breakfast Everyday</a>

</li>

<li>

<a href="blog.html">Love Day Buy 2 Large Shakes Get 1 Free</a>

</li>

<li>

<a href="blog.html">Super Duper Burger New Year With Free Extra Topping</a>

</li>

<li>

<a href="blog.html">Free Membership Card From Star Roller Skates</a>

</li>

<li>

<a href="blog.html">Free Retro Diner Tumbler For Every Burger Special Order</a>

</li>

</ul>

<h1>Categories</h1>

<ul>

<li class="first">

<a href="blog.html">Promos</a>

</li>

<li>

<a href="blog.html">Weekends</a>

</li>

<li>

<a href="blog.html">Breakfast</a>

</li>

<li>

<a href="blog.html">Events</a>

</li>

<li>

<a href="blog.html">Valentines</a>

</li>

<li>

<a href="blog.html">Christmas</a>

</li>

<li class="last">

<a href="blog.html">2023</a>

</li>

</ul>

</div>

</div>

<div id="footer">

<div>

<ul>

<li class="first">

<h2>Delivery Hotline</h2>

<h3>Call 0-123-456-789</h3>

<ul>

<li>

<a href="http://www.freewebsitetemplates.com/go/facebook" class="facebook"></a>

</li>

<li>

<a href="http://www.freewebsitetemplates.com/go/twitter" class="twitter"></a>

</li>

<li>

<a href="http://www.freewebsitetemplates.com/go/googleplus" class="googleplus"></a>

</li>

</ul>

</li>

<li>

<a href="index.html"><img class="logo" src="images/logo-footer.png" alt=""></a>

<ul class="navigation">

<li>

<a href="index.html">Home</a>

</li>

<li>

<a href="about.html">About Us</a>

</li>

<li>

<a href="menu.html">Menu</a>

</li>

<li>

<a href="contact.html">Contact Us</a>

</li>

</ul>

<span>&copy; 2023 RetroDiner.com. All Rights Reserved</span>

</li>

<li class="last">

<h2>Follow Us By Email</h2>

<form action="index.html">

<input type="text" name="subscribe" value="Enter Your Email Here...">

<input type="submit" value="">

</form>

</li>

</ul>

</div>

</div>

</body>

</html>

**Index.html:**

<!DOCTYPE html>

<!-- Website template by freewebsitetemplates.com -->

<head>

<title>Reverse Dictionary</title>

<meta charset="utf-8">

<link href="css/style.css" rel="stylesheet" type="text/css">

</head>

<script type="text/javascript" src="script/reverse.js"></script>

<body>

<div id="header">

<div>

<a href="index.html"><img class="logo" src="images/logo.jpg" width="513" height="84" alt="" title=""></a>

<a href="index.html"></a>

<ul class="navigation">

<li>

<a class="active" href="index.html">Home</a>

</li>

<li>

<a href="about.html">About</a>

</li>

<li>

<a href="contact.html">Contact</a>

</li>

</ul>

</div>

</div>

<div id="body">

<div class="featured"> <a href="burger.html"></a>

<h2>Welcome to Reverse Dictionary</h2>

<p>

To build a software where input the user gives, is an explanation of the word and output the software gives is the relative word list.

</p>

<table><tr><td>

<i>Search Here</i></td><td>

<textarea rows="4" cols="20" id="pharse" style="font-size: 15pt"></textarea></td><td> <div id="info1" style="display: none">

<form>

<h2>Please Wait</h2>

</form>

</td></tr><tr><td></td><td><input type="submit" id="but" value="Reverse" onclick="validate()"></td><td></td></table>

</div> </td><td></td><td><div id="info" style="font-size: 15pt"></div></td></tr></table>

</div>

</div>

<div id="footer">

<div>

<ul>

<li class="first">

<h2>Srm University</h2>

<h3></h3>

</li>

<li>

<a href="index.html"></a>

<ul class="navigation">

<li>

<a href="index.html">Home</a>

</li>

<li>

<a href="about.html">About It</a>

</li>

<li>

<a href="contact.html">Contact Us</a>

</li>

</ul>

<span>&copy; 2014 Reversedict.com</span>

</li>

</ul>

</div>

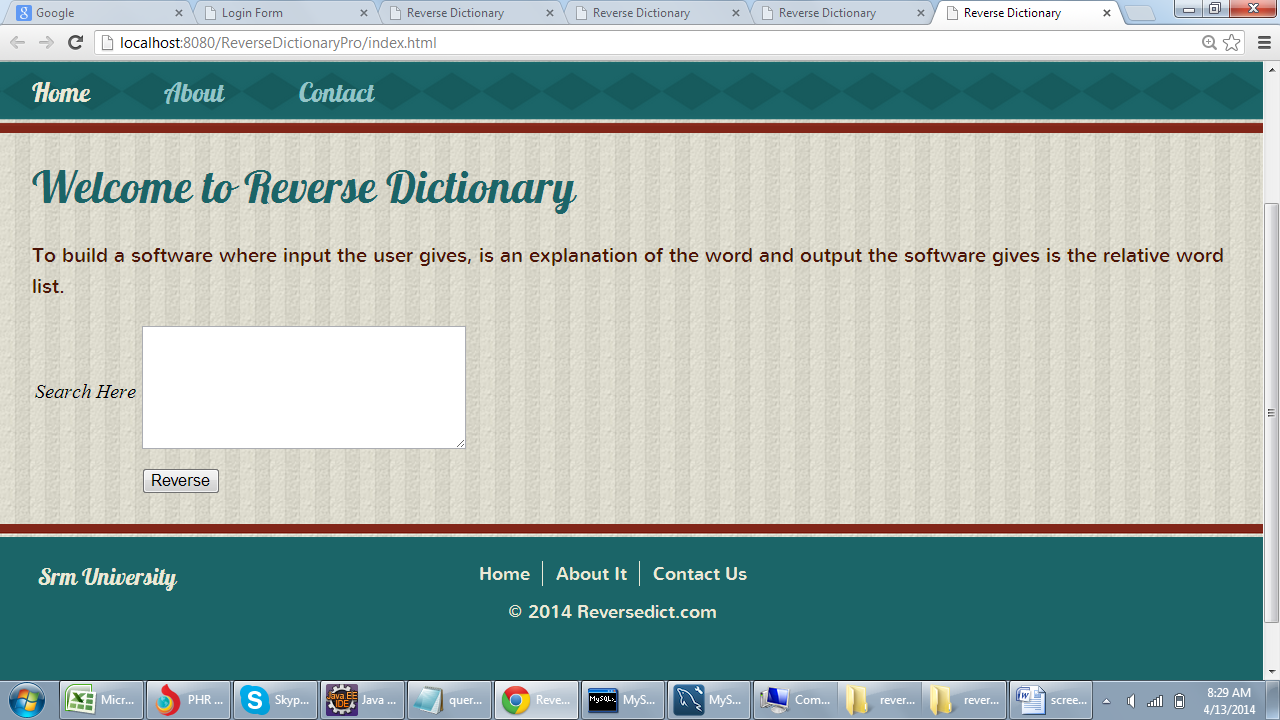
</div>

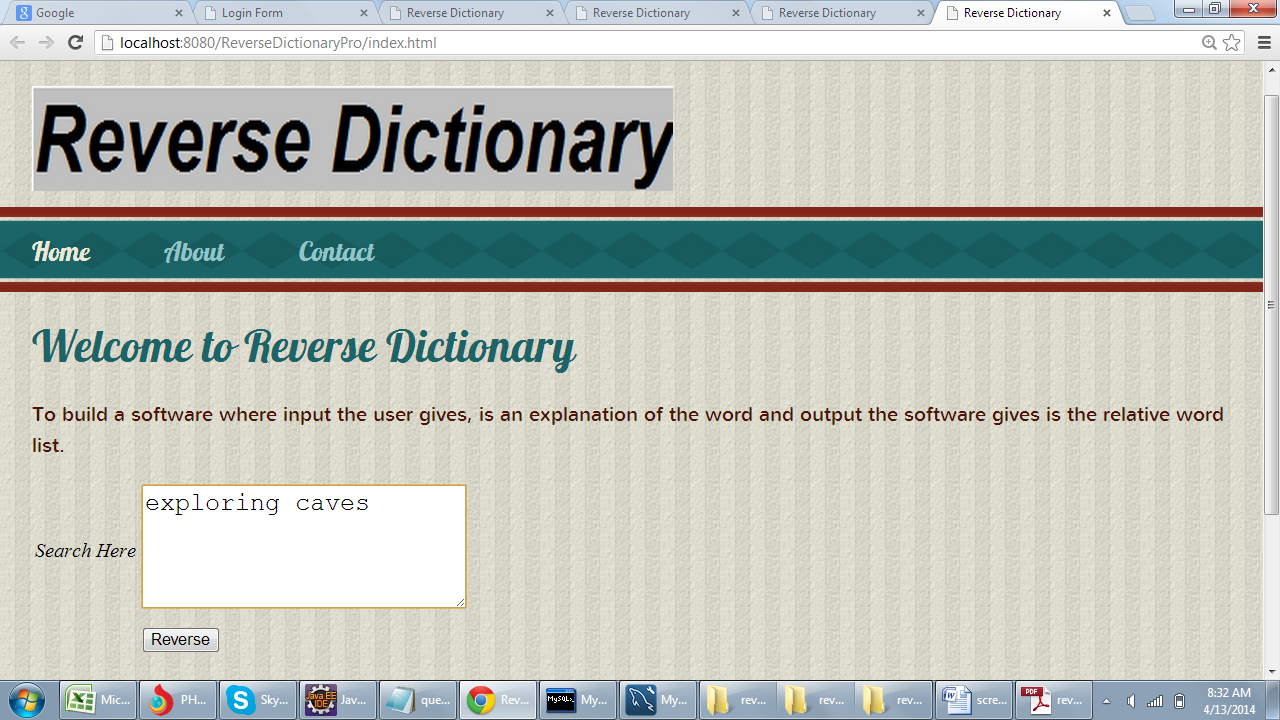
</body>

</html>

**APPENDIX-2**

**1)Index Page**



**2)User Input**

**3)Hold Page**



**4)output**