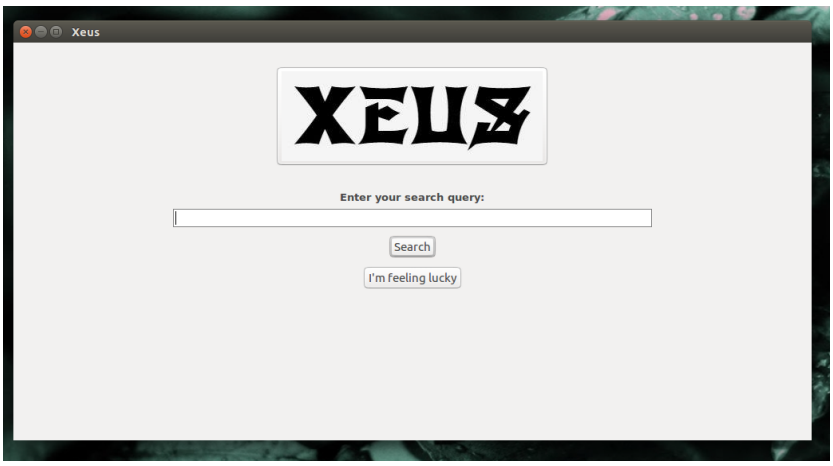
XEUS Search Engine

Animesh Bohara
(160050040)
ani.bohara@gmail.com

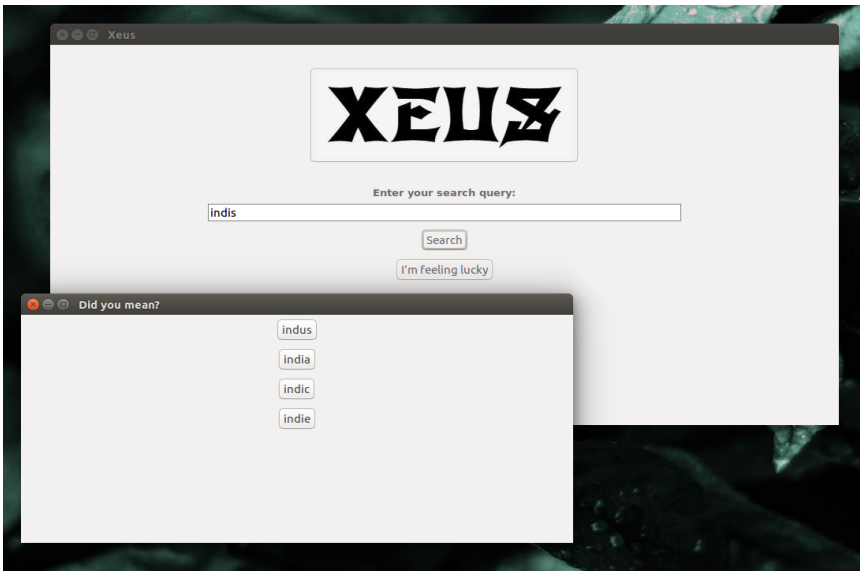
Anmol Singh
(160050107)
anmol107iitb@gmail.com

Gaurav Didwania
(160050020)
gauravdidwania998@gmail.com



Problem Statement:
To develop a search engine which would search for given query within a predefined set of web pages. The same concept can be used in a search engine for a digital library of books.

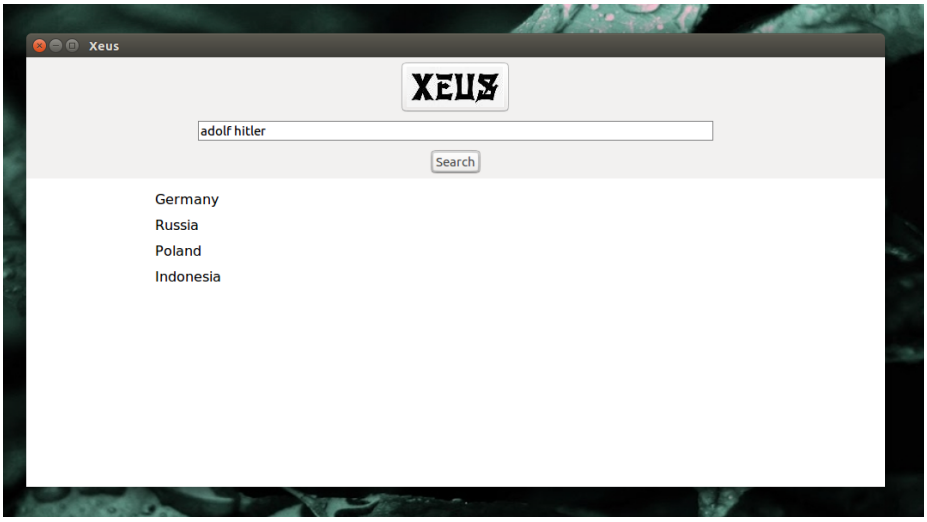
Program Design:
Our program crawls over the entire text of all the webpages and creates a database of words in the form of a global tree (refer the image). The tree contains all the words along with the list all web pages containing the word and the number of occurrence of the word in a web page. This enables lightning quick search speed at the cost of longer compilation time although we have crawled each web page only once and extracted both texts and links in the same crawl.
Next we have used the number of occurrences as a means to sort the results. For a query containing more than one word the common webpages containing most words is placed higher in rank order.
In case of a mistyped query our program involves a “did-you-mean” feature which searches for the closest possible words in our database.



We have developed the algorithm for pagerank but we have not used it in the final output as its relevancy in our context is minimum.
Using racket/gui/base we've presented the top searches in a canvas with URL links to the respective web pages, where this GUI system takes as input a list of hyperlinks. Also "I'm feeling lucky" button directly gives one the web page to the top search output.

Sample Input-Output:

Input	***Output (top search)***
Arc de Triumph is in which country	France
Mumbai is in which country	India
Which country capital is London	United Kingdom
Donald Trump	United States
Narendra Modi	India
Adolf Hitler	Germany



Points of Interest: (commented out HOFs and Abstractions in the code)

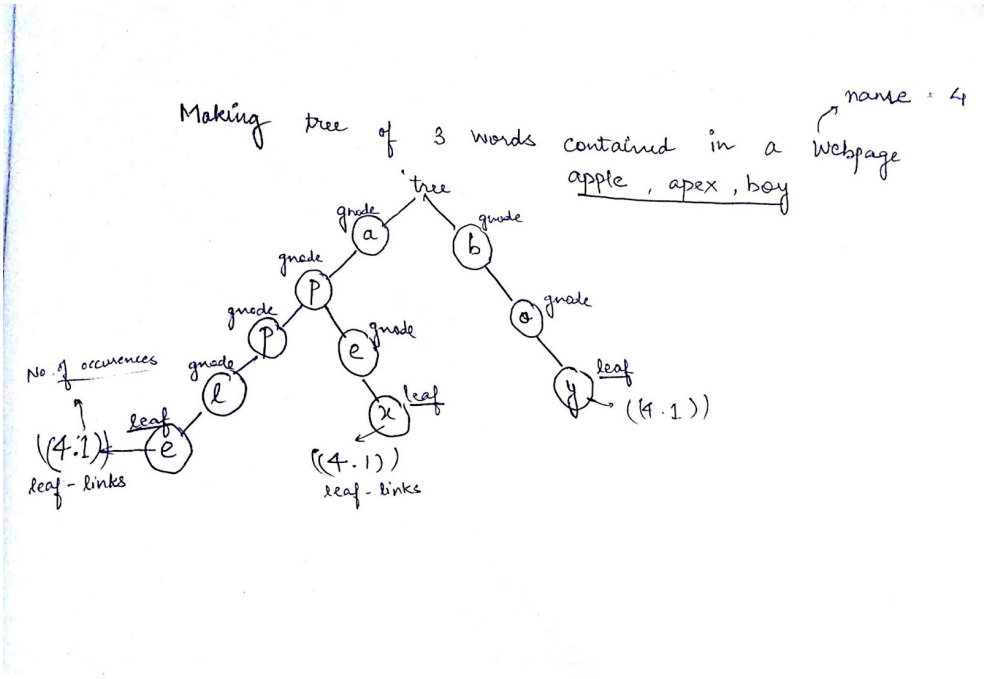
- ❏ HOF:

Higher order function used in creating the function providing did-you-mean(spell checker) and in present function which checks the presence of a given leaf or node at a given tree level.

- ❏ Abstractions:

Used in GUI for defining the general forms to frames, buttons, text-fields, messages, etc. Also using abstractions to avoid hard coding for printing the search results in canvas with their respective links.

- ❑ Clever Coding in Tree



This helps us achieve a LIGHTNING FAST SEARCH.

Packages Used:

racket/gui/base, racket/draw net/url, net/sendurl, sxml, (planet neil/html-parsing:3:0), math/matrix

Concepts covered by project:

- ➢ Crawling and indexing of web
- ➢ Fast search through word storage in trees
- ➢ Rank relevancy (sorting according to priority)
- ➢ Scheme representation of an HTML file
- ➢ Scheme graphical interface
- ➢ Vectors and strings
- ➢ Pagerank algorithm

Limitations and Bugs:

- ➢ Accumulation of buttons in “Did you mean?” dialog box on its continuous calling
- ➢ Limited amount of web pages that could be searched upon within a reasonable amount of compilation time.

Shortcomings:

Pagerank : Our original idea was to implement the pagerank algorithm for ranking order. Our code for database formation also creates a vector each element of which contains list of all the links going out of a webpage (wikipedia contains many links but the important ones are mainly contained in paragraphs and list. Java Scripts and image links are not considered).

- ❑ The problem is that every web page contains many other links than the webpages saved. This caused the need to filter the links based on their presence among the downloaded webpages causing drastic increase in the compilation time.
- ❑ Since our entire database of web pages is limited, the very necessity of pagerank has been compromised. Since the incoming links will be very limited and only from the webpages of a particular website, difference in the pagerank values of webpages will be hardly noticeable and not practically relevant.