Engine Gaadi

Abhishek Bhatta
Aneem Patrabansha
Bibhuti Regmi
Prabesh Aacharya
Pratik Shrestha

July 2, 2014

Abstract

Search engine has been one of the greatest developments of this modern era following the invention of World-Wide Web(WWW) and internet. Searching and fetching the required files under a click of a word, is itself a miracle come real. However, daunting this task may seem, organized approach and development through scratch, we can create a 'search system' that can search required stuffs with the networked system. So, our project 'E library Search Engine' is what it is all about. Searching files through the server of E-library and displaying the most probable results in an user friendly interface is the core idea that fuels our actions.

Acknowlegement

We would like express our sincere gratitude to the Department of Computer Science and Engineering for providing us the semester project. This project provides platform for developing a system with teamwork as a motto, along with acquiring some programming skills. Furthermore, We would also like to thank our supervisor, Mr. Manoj Shakya for providing his time in supervising our project. Furthermore, we would also like to provide our sincere gratitude to Mr. Prabin Gautam,Mr. Aadesh Neupane and Mr. Rohit Amatya. Midst of all the busy schedules they had, help and guidance they provided regarding the project was really moving. Also we would like to thank other friends who indirectly helped us build our project.
Thank You!

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Background:

Help Nepal Network(HeNN) is the largest charitable network of Nepal which is running various campaigns to provide assistance in the fields of health and education in rural Nepal. Among various campaigns, E-library is the most ambitious and innovative project launched by HeNN. It has aim of opening at-least a e-library per district. However, the cost requirement for the resources limits this project. So for the sake of cost management and other technical difficulties, this project has initiated Linux Terminal Server Project (LTSP) such that clients can operate their monitors through a central powerful server, without requiring hard disk for each computer.

1.1.1 Present Scenario and Motivation

Exciting it may seem, it brings database full of files(different formats like text, pdf, mp4 and many others) in gigabytes under a single server. This creates the challenge of finding the required file in no time. This inability to search file can create a huge setback in terms of establishing a LTSP networked system, where searching and locating files through different clients is very important. However, till now there has not been development of such searching system that would meet the aforementioned requirement. So, keeping on mind the need for a search system, we got the idea of 'Search Engine' development for improved and comprehensive search. E-library is one of biggest project in the electronic history of Nepal. However, this limiting factor of not having search system would lag the project a big time. So, there comes the system 'E-library Search Engine' under the play. Thus, the legacy of this project was itself sufficient to inspire and motivate us to do this project.

1.1.2 Objectives

Our major objectives are:

- To make an E-library system more effective and efficeint by making its search engine.
- To make an interactive GUI for the users to search the e-library contents
- To make the search even for the contents inside the files like metadata, tags, etc.
- To make search faster and display the required results in a ranked manner.
- To add filters(like pdf, text, audio,etc) to the search.

Chapter 2

LITERATURE REVIEW

Before proceeding further with our 'E-Library Search Engine', we decided to gain some information about the existing search engines in market. So for that purpose we researched about the core concepts and ideas behind them. Obviously, Google and Windows Search caught our sight, so we did some case study in it so that the structure and design of it may be helpful for our project in future.

2.1 Existing Engines

2.1.1 Google

Google is large-scale web search engine which makes heavy use of the structure present in hypertext.It is designed to crawl and index the Web efficiently and display results in ranked manner. Google uses following three concepts in its anatomy:-

Web Crawling:

It is the process of browsing through world wide web to update the web contents in the google server so that it can be used for further processing.

Indexing:

Google parses out all the links in every web page and stores important information about them in a 'anchor file'. This file contains enough information to determine where each link points from and to and the text of the link. This is called indexing.

Searching:

The URL resolver reads the anchors file and converts relative URLs into absolute URL. It also generates a database of links which are pairs of docIDs. The links database is use to compute PageRanks for all the documents using the 'PageRank Algorithm' especially designed by Google.

2.1.2 Ubuntu Dash

Dash is very constrained to do text and facet searching. It's not for browsing photos, or browsing music . Dash things are search results - a search box at the top, with search results, with headers, and search filters. Application Lenses integrate with your system and applications. There are currently three application Lenses.

Applications

This is basically your application launcher, search for the app you want and launch Files - this is a quick way to find a file via a dash search. People - this is a quick way to find a person via a dash search.

Web Lenses

Web Lenses integrate your system with the web. The idea here is hit the super key, search, and then have it return results from whatever website you care about.

2.1.3 Windows Search

Windows Search collectively refers to the indexed search on Windows Vista and later versions of Windows. Windows Search Builds a full-text index of the files on a computer. The time required for the initial creation of this index depends on the amount and type of data to be indexed, and can take up to several hours, but this is a one-time event. Once a files contents have been added to this index, Windows Search is able to use the index to search results more rapidly than it would take to search through all the files on the computer. Searches are performed not only on file names, but also on the contents of the file (provided a proper handler for the file type is installed) as well as the keywords, comments and all other forms of meta data that Windows Search recognizes. Windows Search supports natural language searches; so the user can search for things like "photo taken last week" or "email sent from Dave". However, this is disabled by default. Windows Search consists of the following components:

Indexer

Crawls the file system on initial setup, and then listens for file system notifications to pick up changed files in order to create and maintain the index of data.

Gatherer

Retrieves the list of URIs that need to be crawled and invokes proper protocol handler to access the store that hosts the URI, and then the proper property-handler (to extract metadata) and IFilter to extract the document text.

Merger

Periodically merge the indices. While indexing, the indices are generally maintained in-memory and then flushed to disk after a merge to reduce disk I/O. The metadata is stored in property store, which is a database maintained by the ESE database engine. The text is tokenized and the tokens are stored in a custom database built using Inverted Indices. Apart from the indices and property store, another persistent data structure is maintained: the Gather Queue. The Gather Queue maintains a prioritized queue of URIs that need Indexing.

Backoff Controller

Monitors the available system resources, and controls the rate at which the indexer runs.

Chapter 3

SYSTEM ANALYSIS AND ARCHITECTURE

3.1 System Requirements

3.1.1 Software Requirements

- **Platform** : Linux
- **Programming language** : Python
- **Interface Design Tool** : PyGTL
- **Modules and Libraries** : NLTK, SubProcess, os, Python-magic, re, Sqlite3, pyPdf, PIL, ID3, WebBrowser, PyGtk
- **Development tools** : Python Shell, Gedit, VI editor

3.1.2 Hardware Requirements:

Minimum requirements are 32MB RAM ,400Mhz CPU, 200MB free disk space

3.2 System Description:

System is a Search Engine which is supposed to search the contents of the local server. A root folder will be provided to the system and it will make the required queries possible with all the possible contents present inside the root folder. The program is accompanied by two phases:

3.2.1 Database Update:

Searching the contents directly by tracing every files was in each directories inside root folder was a time taking process as data scraping , meta-data extraction, tokenization, extensions filtering and other processes are to be run simultaneously. So the basic idea to make search efficient was to create a database which shall do all time taking processes like meta-data extraction, data scraping, file type determining. The database consist of the following table:

File Name	Target	File Type	Meta-data	Content

Once the root directory is given to the program, it will scan through all the possible directories and update the database. Once the database is updated, each change in the server content will be recognized and the database will be re-updated for the changes.

3.2.2 Search Process:

It is provided with the interactive search GUI which does the queries for the client. In the section the user inputs the query in the search bar. The string he inputs is then processed, stop words are removed, and stemming is done. The processed string is then looked up in the database. Even file filters are considered to deliver the required results. Finally the results are displayed in the screen in a ranked order.

Chapter 4

DISCUSSION

4.1 Programming Language Used

4.1.1 Bash Scripting

Bash is a "Unix shell", a command-line interface for interacting with the operating system. It is widely available, being the default shell on many GNU/Linux distributions and on Mac OS X; and ports exist for many other systems. Bash scripting is one of the easiest types of scripting to learn, and is best compared to Windows Batch scripting. Bash is very flexible, and has many advanced features that you won't see in batch scripts.

4.1.2 Python:

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as en extension language for applications that need programming interfaces. Finally, Python is portable across all major hardware and software platforms.

4.2 Python Modules Used

4.2.1 NTLK

The Natural Language Toolkit (NLTK) is a Python package for natural language processing. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

We are using this module to perform text processing like tokenization, stemming and all the stop words removed.

4.2.2 SubProcess

The subprocess module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. This module intends to replace several older modules and functions.
We are using this modile to run various shell commands and manipulate its outputs.

4.2.3 os

The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on be that Windows, Mac or Linux. You can find important information about your location or about the process.
We are using this module to get the directories and crawl through each directory inside the root folder and run various terminal command.

4.2.4 Python-Magic

This module uses ctypes to access the libmagic file type identification library. It makes use of the local magic database and supports both textual and Multipurpose Internet Mail Extensions MIME-type output.
We are using this module to extract the filetypes of the different files present in server.

4.2.5 re

This module provides regular expression matching operations similar to those found in Perl. Both patterns and strings to be searched can be Unicode strings as well as 8-bit strings.
We are using it to execute regular expressions to filter text patterns.

4.2.6 PIL

PIL stands of Python Imaging Library. PIL adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.
This module is used to extract the metadata(i.e. tags, size, pixels) present in the image files.

4.2.7 ID3

This module allows one to read and manipulate so-called ID3 informational tags on MP3 files through an object-oriented Python interface. We are using this module to extract the metadata present in the audio files.

4.2.8 WebBrowser

The webbrowser module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling the open() function from this module will do the right thing.
We are using this module to provide the link to the file that is the result of the required query.

4.2.9 PyGTK

PyGTK lets you to easily create programs with a graphical user interface using the Python programming language. The underlying GTK+ library provides all kind of visual elements and utilities for it and, if needed, you can develop full featured applications for the GNOME Desktop. PyGTK has been used in a number of notable applications like gedit, bittorrent, gnome sudoku, Ubuntu Software Center, etc.
We are using this module to design the basic GUI of our Search-Engine.

4.3 Database Used

4.3.1 Sqlite3

SQLite is a C library that provides a lightweight disk-based database that doesnt require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. The SQLite accesses its storage files directly.
We are using sqlite3 as a database of our system where all the informations like contents, metadata, type and path of the files will be stored and search will be performed of the database and the possible results will be displayed on the interface with the path link of the file.

4.4 Techniques Used

4.4.1 Probing

Probing is the technique of accessing the different folders and subfolders inside the root directory. Probing is implemented to crawl through the every diectory so as to access the every files and their contents.

4.4.2 Text-processing

Text-processing is the technique that scrapes the contents of the document files. While scraping the documents, the stop words are removed, punctuations are discarded and the main words are tokenized. This technique solved the problem of database being populated by bulky and useless words. This also made search process efficient.

4.4.3 Metadata Extraction

By metadata extraction, we mean extracting the exif tags present document files. Exif tags are used largely to encode additional information related to image generation by digital still cameras. EXIF is the abbreviation of 'Exchangeable image file format'.The metadata is then stored in the database that would make search more significant.

4.4.4 Ranking

Ranking refers to displaying the results in a specifeed manner. The rank weight of the searched file is initially 0. If searched keyword is found in the name column of the database, add 1 to the rank weight. Similarly if found in meta-data or content, add 0.1 and 0.01 respectively and this gives the rank-weight of the file. Hence the search results are displayed on the basis of their rank weights.

4.5 Major Problems Encountered and Solutions Implemented

Though the strategies were effectively implemted, but there were some serious problems encountered during the development phase. The problems were anyhow solved using various tactics. Some major problems encontered and their ultimate solution are given below.

Problem 1:

Searching was a time taking process as the Search Engine have to scrape the contents from bulky files which were all ramdom. So there was more task than just search for the engine i.e. to scrape all the possible contents and search for the possible results. Scraping is the time taking process that would retard the efficiency and speed of search process.

Solution 1:

The ultimate solution was to create a database that would store all the metadata and processed contents of all the files present in the root directory and sub-directories. This would though take time at beginning to update the database, but the search process would be a lot faster.

Problem 2:

While creating a database, it stores all the possible informations of the files inside the root folder. But the problem was if the contents inside the root directory are modified, or files are added or removed, the database still carries the old information of the files. This would create no access to the newly added files or unrecognized access to the deleted files. And reupdating the whole database would take more time.

Solution 2:

A bash script was created and inotify was used that would watch and notify the changes that are occuring inside root directory. So any changes when notified, the particular change is recognized and that file is scraped automatically and database is reupdated. So the access was made dynamic.

Chapter 5

CODE DESCRIPTION AND PROGRAM FLOW

5.1 General Algorithm :

5.1.1 Database Management:

```
1: Start
2: Provide the root folder to the system where the contents are stored
3: Probe the current folder.
    3.1: Look for the files
    3.2: If file found, file processing
        3.2.1: Store file name, file path, and file type in the name, target, type column of database
        3.2.2: Extract the meta-data and store it in meta-data column of the database
        3.2.3: If the file is document type, like .odt, .docx, . Html, .pdf , etc. scrape the contents using text processing and store it in content column of database.
    3.3: If all files processed or no files found, look for directories
    3.4: If folders found and unvisited, enter the folder and mark visited
    3.5: Goto 3:
    3.6: If no files and folders are found return back from the current directory, make it current directory and goto 3.4:
    3.7: If current is out of the root, goto 4:
4: Stop
```

5.1.2 Searching:

1: Start
2: Read the query as a string
3: Process the text
4: Search the processed text in the database
5: If text found in meta-data or content or name column , mark the row as found
6: If search is made with a filter, see the filetype of the row marked as found, if file type doesn't match, discard the row and mark it unfound
7: See the occurrence of the query text in each row
8: Display the name of the files with its path as hyper link and sort the results in a ranked way.
9: Stop
text processing: by text processing we mean, scraping the documents, and stemming i.e. stop words, punctuations are removed and useful information are only selected.
rank process: The rank weight of the searched file is initially 0. If searched keyword is found in the name column of the database, add 1 to the rank weight. Similarly if found in meta-data or content, add 0.1 and 0.01 respectively and this gives the rank-weight of the file.

5.2 Flowcharts

Figure 5.1: Setup and Scraping.

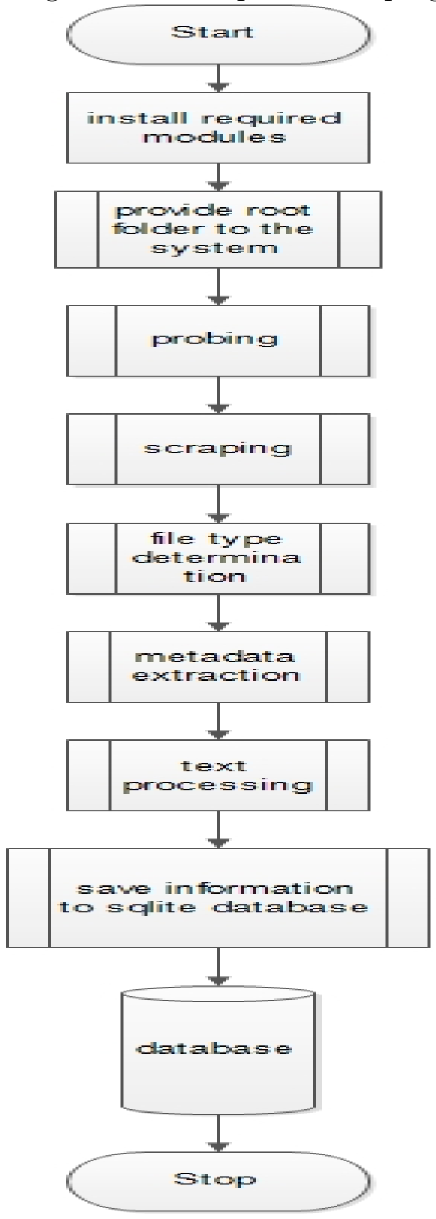


Figure 5.2: Text Processing.

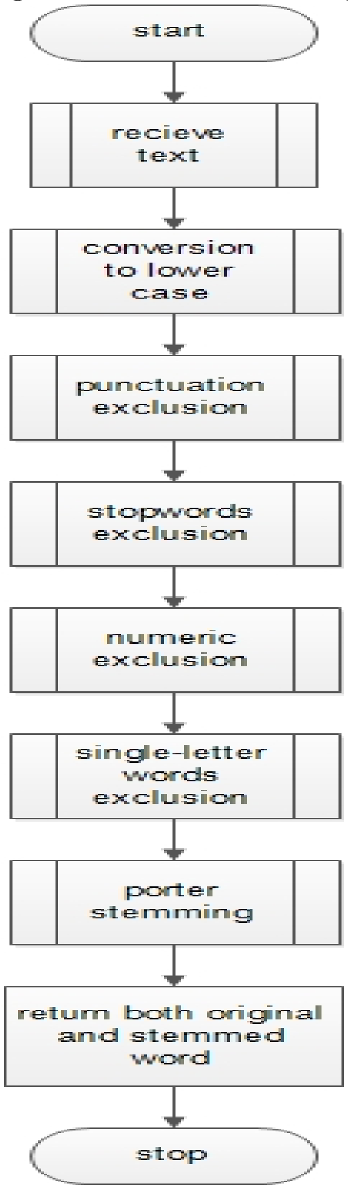


Figure 5.3: Search.

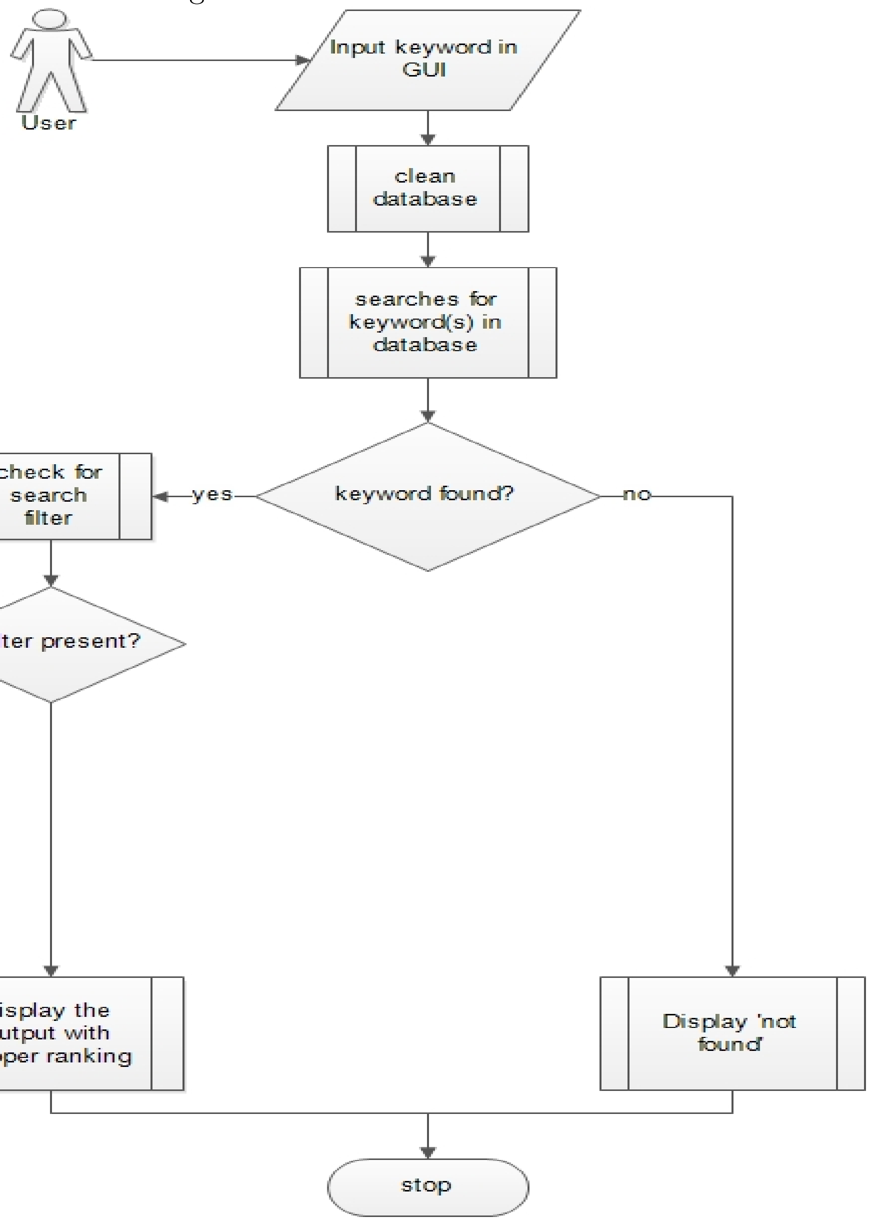
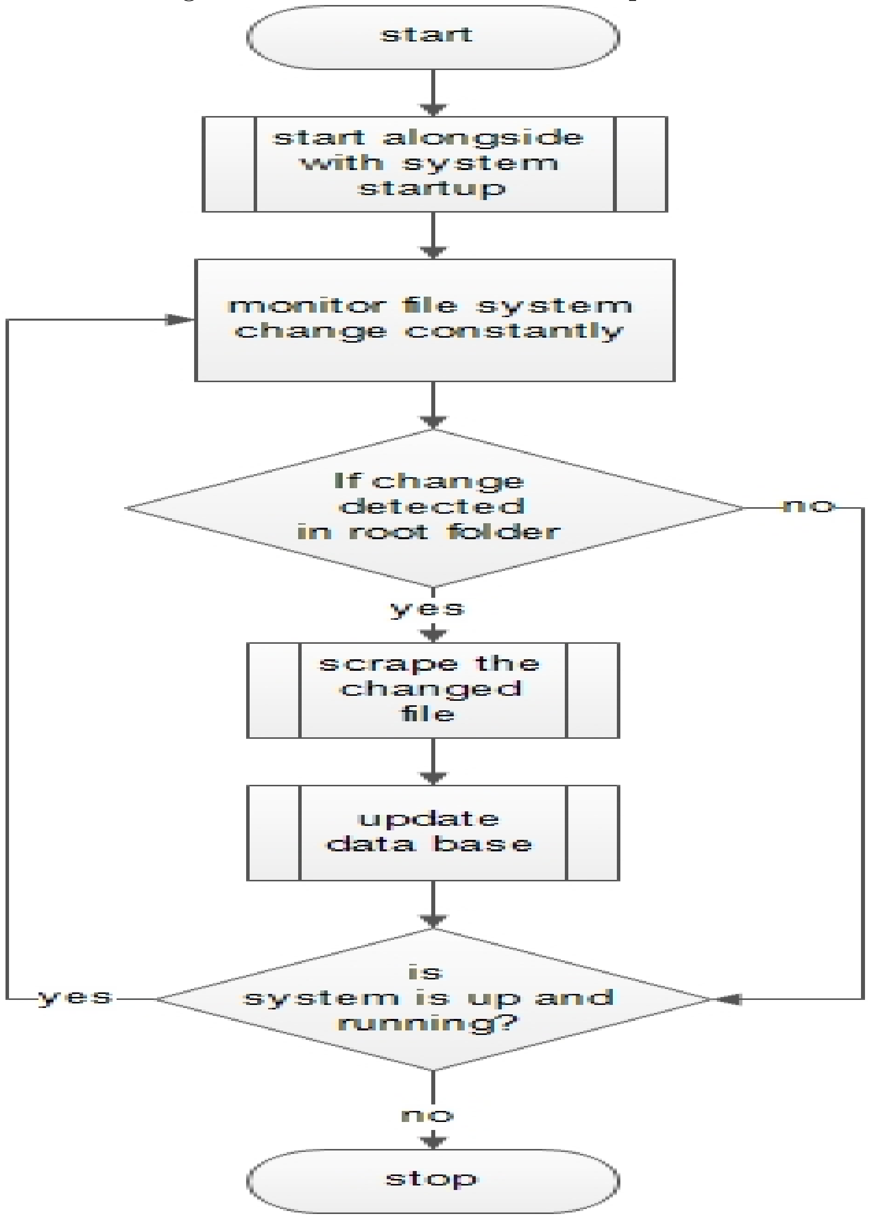


Figure 5.4: Automatic Database Update.


start
start alongside with system startup
monitor file system change constantly
If change detected in root folder
no
yes
scrape the changed file
update data base
is system is up and running?
yes
no
stop

5.3 Code Description

5.3.1 Setup.sh

It is a bash script used to prepare the computer for the search. It contains of various terminal codes that basically installs all required python modules and managed various files in root folders allowing the program to work.

5.3.2 Scrape

```
for path, subdirs, files in os.walk(home):
    for subdir in subdirs:
        file_stack.append(os.path.join(path, subdir))

    for name in files:
        a_file = os.path.join(path, name)
        file_stack.append( os.path.join(path, name) )
```

The Scrape file probes through the whole fiesystem and gathers name and location of all the files within it in an array. It then passes all those information to Scraper for further processing.

5.3.3 Scraper

```
    file_words = [ nltk.PorterStemmer().stem_word(word).lower() and word.lower()
        for word in tokenizer.tokenize(content)
        if word not in nltk.corpus.stopwords.words(english) and word.isdigit()= =False
and len(word) greater than 1 ]
```

The Scraper file acquired a file path, extracts name, extension and type out of it. It also extracts its metadata and content text and filters it through various text processing. Finally it stored the name, location, file type, metadata and content to a database.

5.3.4 Extensions

It is simply a text file with most of available extensions which is referred in case MIME type is not specified.

5.3.5 Search

weight = name occurence x 1 + metadata occurence x .1 + content occurence x .01

This program simply recieves keywords and then quickly scans the database for its presence. In case of multiple occurence it provides them a weight depending on number of occurence on file name, metadata and content. The files are ramked according to this weight

5.3.6 Autoupdate.sh

It is a bash script that uses inotify to monitor our filesystem of changes. If so it acknowledges it to Update.

5.3.7 Update

Update recieves the name of file and the type of action on it such as create, move or modify. It manages the database accordingly. If new files are created or modified, it runs it past Scraper and adds to database.

5.3.8 Clean

It is executed everytime we search for something. It deletes the record of deleted files in the filesystem.

5.3.9 GUI

It is the interface for the search engine. It consists of entrybox and checkbox for filters like text,audio,video which will enable us to filter according to the required text files we need .after entering the keywords in the entry field , the entire database is searched for the keywords. With the help of the ranking algorithm the results are then displayed.

Table

self.table = gtk.Table()

The entire screen is taken as a table and all the buttons are displayed by attaching it to the table. Here we have table of size 20X12.

Buttons

Buttonname = gtk.Button()
Buttonname.connect('clicked',asd)
gtk.CheckButton()

There are mainly two types of buttons used here. Checkbuttons and normal button.checkbuttons is for the reqired filter.

Pane no

table.attach(self.pane_no_label,5,6,10,11)

If the search result exceeds the window sixe then it is dispalyed in the next page. The page number is displayed with the help of label which is attached to the table.

Chapter 6

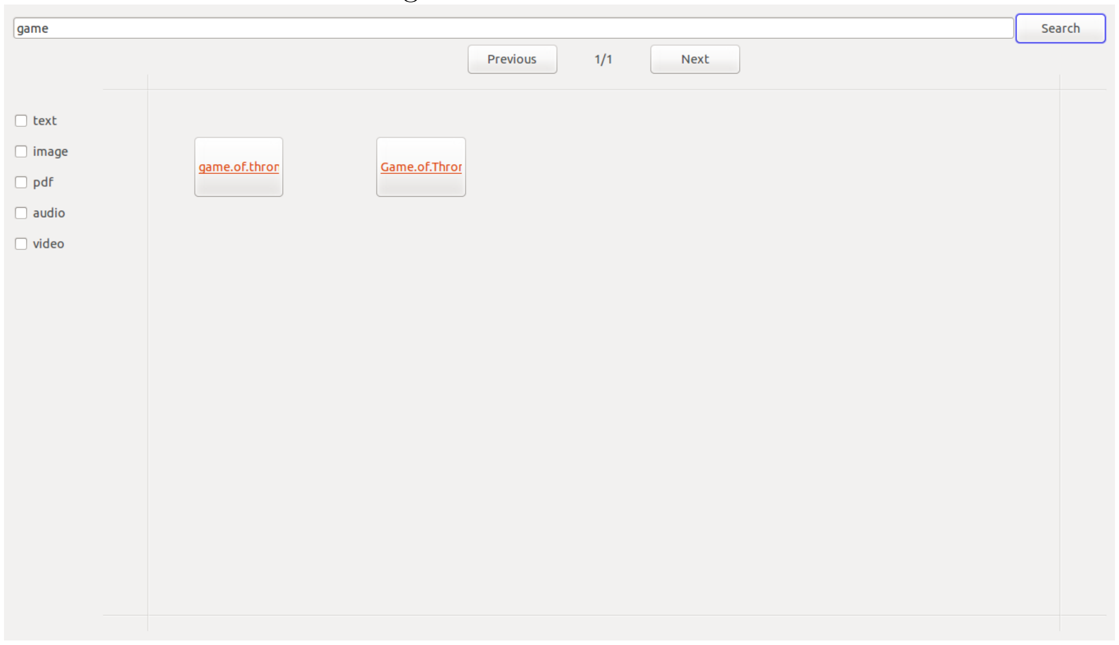
PRODUCT DESCRIPTION AND PERFORMANCE

The ”E-Library Search Engine” was developed as a one stop destination for the E-library who wanted ease to access the contents of it. Thus our program has been made as simple as possible so that it can be understood by even a layman easily.

6.1 Product Overview

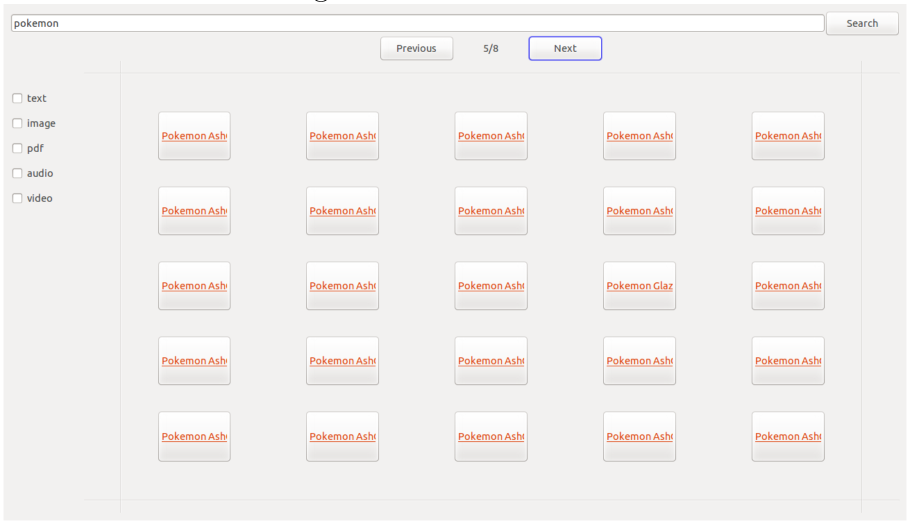
The graphical interface of our software is shown below:

Figure 6.1: Basic Interface.



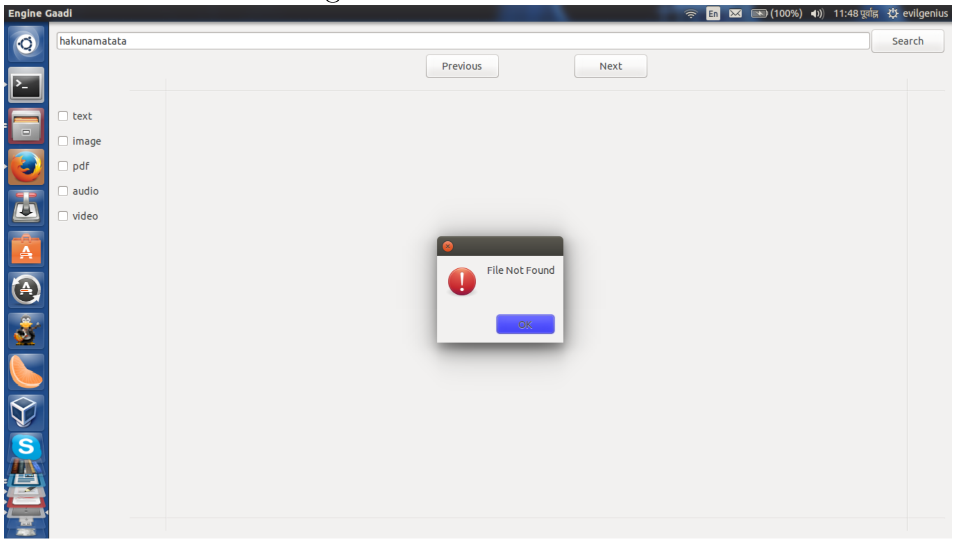
We are to search our requirements in the searchbox. The results are then displayed in the display panel. 25 results are shown at a time. If exceeded, we can navigate through many panes for it by Prev anf Next Button. Clicking on the item opens the file.

Figure 6.2: Basic Search.



If file isn't found a message is displayed for it.

Figure 6.3: File not found.



File Not Found

Our search can also be filtered by type i.e. file of a particular file type can be searched after.

Figure 6.4: Searching for Video.

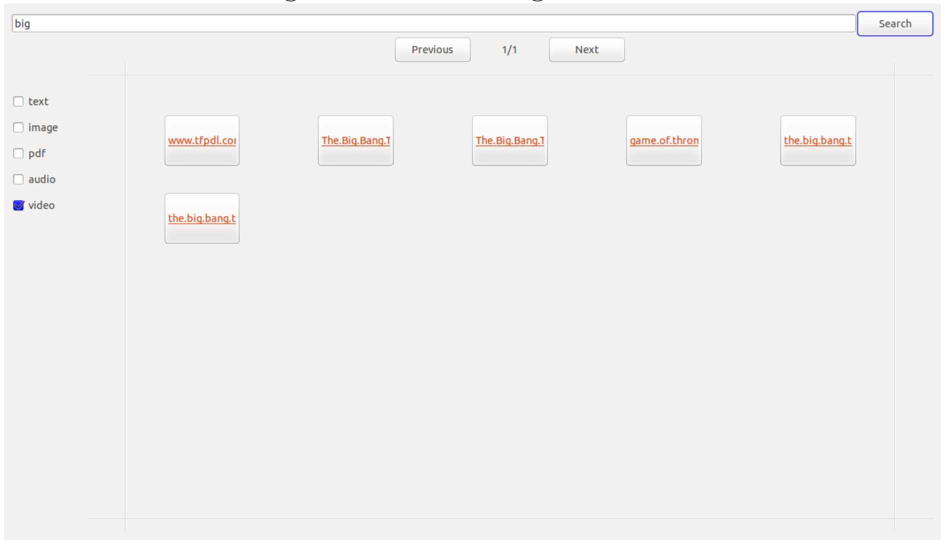
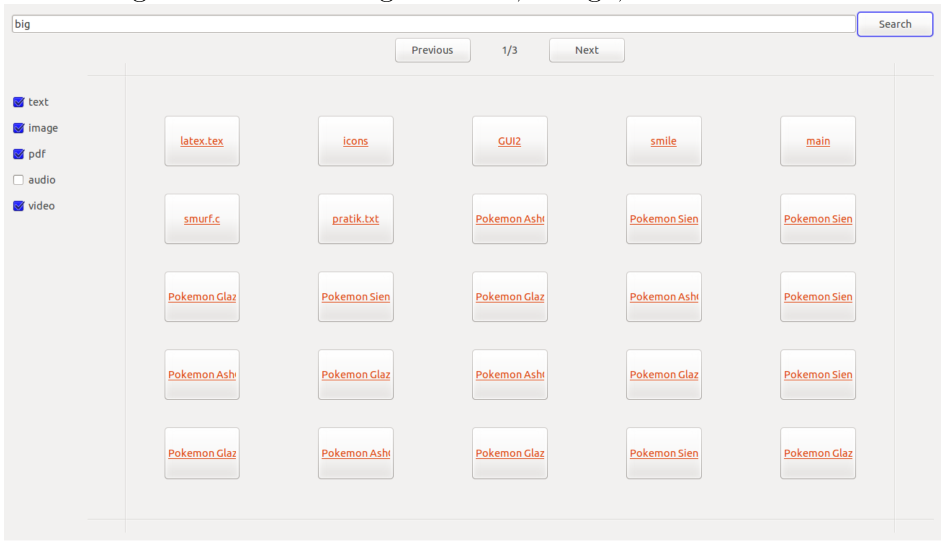


Figure 6.5: Searching for Text, Image, Pdf and Video.



6.2 User Manual:

You can search for the required keywords by typing in the entry box and pressing search button if you want to filter files like music and text only check on the required type you want to search and press search.

6.3 Basic Features:

The basic features of our program are:

- Search through each sub-directories inside the root folder.
- Search even inside the contents of the files present.
- Auto update the database when any changes are done with the files inside the root directory.
- Add the filters for the filetypes for required search results.
- Search results are displayed in a ranked manner.
- An interactive interface for ease of access

6.4 Future Enhancements:

Although our project met up the expectations and requirements of the search system, during the course of this project, we were able to find some points that would make this project more scopeful. Many features would be integrated in that case and this system would be more widespread in use. Our system is able to search files from huge chunks of databases, so making this system packaged with web would be our primary motive for enhancing our system. This meets the philosophy of our search engine anyways. Making it 'Web-Spread', we can optimize its use.
Furthermore, before launching this in web, however, we must be able to expand the types of file extensions, this system can search. Our system now is limited to more popular and useful file extensions, but making it web-based would mean constantly updating and increasing the domain of extensions. Moreover, our system is made for a linux server system. However, in future, in order to make this more useful and ubiquitous, this system should be able to perform under every platforms possible for instance Windows, apple and android as well. Therefore, some of these afforementioned ideas would enhance this system making it more user friendly and interesting.

Chapter 7

CONCLUSION

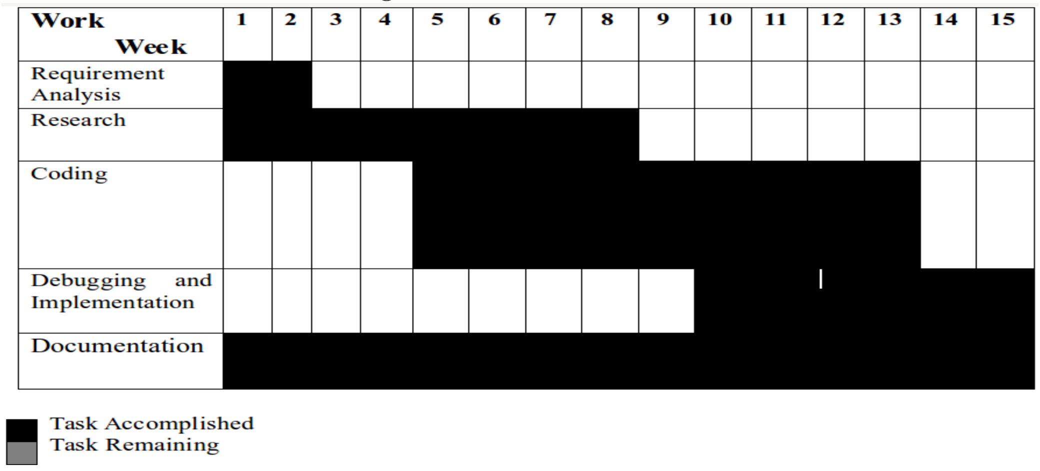
This has been a hell of a project so far. Beginning from a literature review and researches done on search engines to basic engineering of codes, the experience and the knowledge we acquired have been overwhelming. Befoe the start of the project, all we knew was the word 'Google', but then with the help of google itself, we tracked down techniques of filing and indexing so as to keep the organized track of the files. The keywords entered and the subsequent tokenized words would be searched through the database. Then, the development of scientic and customized technique of ranking algorithm for this system, would facilitate the listing of files related to the keyword in one way or the other in order of their importance. Finally, a design and development of user interface would provide platform for the display of the result. Hence, with the distributed labor and brain, we were able to devise this system that shall in future resonant with the needs of search system for the E-library organization.

Chapter 8

APPENDIX

8.1 Gantt Chart

Figure 8.1: Gantt Chart.



Work / Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Requirement Analysis	■	■													
Research	■	■	■	■	■	■	■	■							
Coding					■	■	■	■	■	■	■	■	■		
Debugging and Implementation										■	■	■	■	■	■
Documentation	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

■ Task Accomplished
▨ Task Remaining

8.2 Work Division

Database Management: Abhishek Bhatta, Pratik Shrestha
Content scraping: Abhishek Bhatta, Pratik Shresha
Text processing: Aneem Patrabansha
Search and Sorting Algorithms: Prabesh Aacharya
Interface Design: Bibhuti Regmi
Coding: All

Chapter 9

BIBLIOGRAPHY

http://en.wikipedia.org/wiki/Windows_Search [2014/03/26]
http://infolab.stanford.edu/ backrub/google.html [2014/03/28]
https://www.youtube.com/watch?v=cY7pE7vX6MU [2014/04/15]
http://www.tutorialspoint.com/sqlite/sqlite_syntax.html [2014/05/01]
http://zetcode.com/gui/pygtk/ [2014/05/03]
https://www.youtube.com/watch?v=SzJ9koJBNbs [2014/05/03]
http://www.pygtk.org/pygtk2tutorial/ [2014/05/13]
https://wiki.ubuntu.com/DashAsBinSh [2014/06/13]
http://zetcode.com/db/sqlitepythontutorial/ [2014/06/20]
http://www.nltk.org/book/ch01.html [2014/06/23]