SOFTWARE REQUIREMENTS SPECIFICATION

for

DEVELOPMENT OF SEARCH ENGINE

Version 1.0

Prepared by Neeraj Kumar, Khushboo Kumari, Shweta Kumari

Tutorialspoint

December 28, 2016

Contents

Revision History

Name	Date	Reason For Changes	Version
Initial Version	Dec 16,2016	Initial	1.0

1 Introduction

1.1 About

This SRS is about a light version of Search Engine which works for a single as well as group of sites. A web search engine is a software system that is designed to search for information on the World Wide Web. The search results are generally presented in a line of results often referred to as search engine results pages (SERPs). The information may be a mix of web pages, images, and other types of files.

1.2 Purpose

- To help you find the web page with the info you need. It has been designed to help people find information stored on a single as well as group so sites.
- This document describes the requirements specification (SRS) for the software infrastructure (or product) that enables the users to search the keywords and get a list of relevant links. This will be like the basic version of Google Search Engine.
- A search engine, that search web links for specified keywords and returns a list of web links and some code snippets where keywords are found.

1.3 Document Conventions

This document follows MLA Format. Bold-faced text has been used to emphasize section and sub-section headings. Highlighting is to point out words in the glossary and italicized text is used to label and recognize diagrams.

1.4 Intended Audience and Reading Suggestions

- This document is to be read by the development team, the project managers, marketing staff, testers and documentation writers.
- One should have basic idea of how search engine works. There are many materials available on the Internet. You can have a look on them before proceeding above. In particular have a look on different stages of search engine development like crawling, indexing, presentation.

1.5 Project Scope/Functionality

- A spider (also called a ”crawler” or a ”bot”) that goes to every page or representative pages on every Website that wants to be search-able and reads it.
- The crawler collects all the links and stores in a database.
- A program that creates a huge index (sometimes called a ”catalog”) from the pages that have been read.
- A program that receives the search request, compares it to the entries in the index, and returns results to the user.

1.6 References

- http://www.tutorialspoint.com/http/http˙requests
- http://computer.howstuffworks.com/internet/basics/search-engine1.htm
- http://www.brickmarketing.com/define-search-engine-index.htm
- http://stackoverflow.com/questions/1501690/parsing-out-data-using-beautifulsoup-in-python
- http://stackoverflow.com/questions/11709079/parsing-html-using-python

2 Overall Description

2.1 Product Components

- **Web crawling, Indexing and Ranking:-** This is the part of the search engine which combs through the pages on the Internet and gathers the information for the search engine. IT then extracts the keywords from the text and uses them during query by users
- **Database:-** Whenever a users enters a query ,the search engine's database is what actually searched. All of the information that a web crawler retrieves is stored in a database. Every time a search engine is used, it is this database that is searched not the live Internet.
- **Search Interface:** This component is an interface between user and the database. It helps the user to search through the database.

2.2 Operating Environment

- **Operating System:-** Ubuntu 16.04
- **Programming environment:-** Python 2.7
- **Data Base:-** Sqlite3
- **Python Packages Used:-** List of packages for python 2.7:
  - **whois**:- for finding domain of a siteSearch Interfaces.
  - **urlparse**:-for parsing url and finding different components.
  - **datetime**:-for finding age of a web page.
  - **urllib2**:-for working on url.
  - **sqlite3**:-for storing and retrieving data from database.
  - **time**:- for finding load time of a web page.
  - **re**:- for providing regular expression matching operations
  - **robotparse**r:-for reading robot.txt file of each site.
  - **bs4** :-for keyword extraction and html file downloading.
  - **os**:- for providing a portable way of using operating system dependent functionality.

- **nltk**:-for Natural Language Processing and extracting keywords
- **collections**;- For counting and sorting efficiently

3 System Features

3.1 Features:

- Scalable i.e. It can be extended up to large scale.
- It can be used to search from multiple websites at a single time
- Fast searching and indexing, simple
- Follows robot exclusion protocol
- Portable(current version will support linux only)
- Page rank can be customized using configuration table
- Implemented in python so it is memory efficient, dynamic and general-purpose programming language
- During Keyword extraction, list of stop words like “is” ,”am”, “are” etc has been used which discard these words from keyword list
- Keywords like “son”, “sons” has been taken same as these have similar importance during searching
- Database has been implemented in sqlite which has lot of advantages
  - Zero-Configuration
    - No ”setup” procedure
    - Easily portable
    - No server process that needs to be started, stopped, or configured.
    - no need for an administrator to create a new database instance or assign access permissions to users
    - No configuration files.
    - Nothing needs to be done to tell the system that SQ-Lite is running.
    - No actions are required to recover after a system crash or power failure.
    - There is nothing to troubleshoot.
  - Server-less
  - Single Database File

- Stable Cross-Platform Database File
- Compact

4 Database Schema

Database name: SearchEngineDB.db

4.1 Description of Attributes of the Database:-

- **SITE_ID**:- It represents unique site.
- **URL_ID** :- It represents unique URL.
- **SITE_URL** :- It represents url of site.
- **SITE_AGE**:- It represents how much old is site
- **LINKS_QUALITY**:- It represents how much percent link in a particular site is good
- **TEXT_QUALITY**:- It represents how much a particular site is well written
- **BOUNCE_RATE** :- It represents how much time user is spending on that site.
- **INBOUND_LIMIT** :- It represents how much a particular site is referred by the other sites.
- **SITE_POPULARITY**:- It represents of much popular is site
- **NO_OF_VIEWERS**:- It represents number of viewers of site.
- **LAST_UPDATE** :- It represents when a site is last updated.
- **SECURITY_WEIGHT** :- It is used to represent whether a URL is secured or not.
- **LOAD_TIME**:- It represents how much time a URL is taking to load
- **FREQUENCY_WEIGHT** :- It represent whether a word is present in body or not ,
- **TITLE_WEIGHT** :- It represent whether a word is present in title or not
- **METADESCRIPTION_WEIGHT** :- It represent whether a word is present in meta word or not
- **URLNAME_WEIGHT** :- It represents whether a word is present is in URL name or not

- **HEADING_WEIGHT**:- It represents whether a word is present in heading or not

4.2 SITE_INFO

CREATE TABLE **SITE_INFO**(

```
SITE_ID      integer ,
SITE_URL      text,
SITE_AGE      real,
TEXT_QUALITY      real,
LINKS_QUALITY      real,
BOUNCE_RATE      real,
INBOUND_LIMIT      integer,
SITE_POPULARITY      integer,
NO_OF_VIEWERS      integer,
SITE_IP_LOCATION      text,
);
```

4.3 URL_INFO

CREATE TABLE **URL_INFO**(

```
SITE_ID      integer ,
URL_ID      text,
URL_LINK      text,
LAST_MODIFIED      text,
SECURITY_STATUS      real ,
LOAD_TIME      real ,
FINAL_URL_WEIGHT      real ,
);
```

4.4 KEYWORD_INFO

CREATE TABLE **KEYWORD_INFO**(

```
KEYWORDS      text,

   URL_ID      integer,
SITE_ID      integer

   FREQ_IN_BODY      integer,
FREQ_IN_TITLE      integer,
FREQ_IN_META       integer,
URLNAME_WEIGHT       integer,
HEADING_WEIGHT       integer
FINAL_FREQ_WEIGHT       integer
);
```

4.5 WEIGHT_CONFIG

CREATE TABLE **WEIGHT_INFO**(

```
WEIGHT_NAME       text
WEIGHT_VALUE       real
);
```

Note:-The above table weight_info is a configuration table in which the weight_name and their respective weight_value is already predefined.

4.6 WEIGHT_CONFIG TABLE CONTENTS:-

Factor_Name	Factor_Value	Max_Weight
Domain_age_factor	1	3
Keyword_body_factor	1	4
Meta_desc_factor	0.5	–
Mta_keyword_factor	0.5	–
Title	1	–
H1	0.5	–
H2	0.25	–
http/https(security)	1	–
Url-name	1	–
Url-site	2	–
Site_quality_factor	–	–
link_quality_factor	–	–
Img_tag_factor	0.01	1
Anchor_tag_factor	0.01	1
Site_popularity	1	–
No._of_viewers	1	–
Inbound_limit	1	–
Bounce_rate	1	–

4.6.1 Explanation for different weight factors:-

- **Domain_age_factor:-** After finding the age of domain in years, the age is multiplied by the **factor_value** . If the result is more than or equal to **maxi-**

mum_weight ,the result is replaced by **maximum_weight** otherwise it remains same.Explanations for the different weight factors:-

- **Keyword_body_factor:-** The keywords in a html page is divided by the total no of keywords in a given html page and multiplied by 100 to find the percentage .Then again the result is multiplied by Keyword_body_factor. If the resultant is greater or equal to **maximum_weight** then it is replaced by **maximum_weight** else it is kept same.
- **Meta_desc_factor:-** If the searched keyword is found in meta description of the page then a **factor value** 0.5 is used for ranking calculation else 0 is used.
- **Meta_key_factor:-** If the searched keyword is found in meta keywords of the page then a**factor value** 0.5 is used for ranking calculation else 0 is used.
- **Img_tag factor:-** This factor will be used as penalty if the **alt** attribute is not found in the img tag. For this , the percentage of img tag in which **alt**attribute is not found is calculated. This percentage is multiplied with the **factor_value** and the result is used for ranking purpose if it is less than maximum_weight else**Maximum_weight** is used.
- **Anchor_tag factor:-** This factor will be used as penalty if the**title** attribute is not found in the img tag. For this , first of all the percentage of anchor tag in which **title** attribute is not found is calculated. Then this percentage is multiplied with the **factor_value** and then the result is used for ranking purpose if fit is less than **maximum_weight** else **Maximum_weight** is used .
- Similar strategy as that of Meta_key_factor is used for Http/Https(Security status), Url_name, Url_site, Site_popularity, no_of_viewers, bounce_rate, Inbound_limit , Link_quality_factor and Site_quality_factor.

4.6.2 How to update WEIGHT_CONFIG TABLE CONTENTS

- Change the corresponding value in *weight_config.py* file and run it
- Also update the tables **KEYWORD_INFO** and **KEYWORD_RANK** table because these tables have used the values from WEIGHT_CONFIG table. So, run again the files *keyword_info.py* and *keyword_rank.py.*

4.7 KEYWORD_RANK

CREATE TABLE **KEYWORD_RANK**(

KEYWORD text
FILE_LOCATION text
URL text

```
TOTAL_WEIGHT      real
);
```

4.8 LOAD_TIME

```
CREATE TABLE WEIGHT_INFO(

LOAD_TIME      text
LOAD_TIME_WEIGHT      real
);
```

4.9 Entries in the LOAD_TIME Table

Load_time1	load_time > 5s	0.0
Load_time2	load_time > 4s	0.1
Load_time3	load_time > 3s	0.2
Load_time4	load_time > 2s	0.3
Load_time5	load_time > 1s	0.4
Load_time6	load_time > 0s	0.5

5 Architecture/System Flow Diagram

5.1 Overall-Architecture

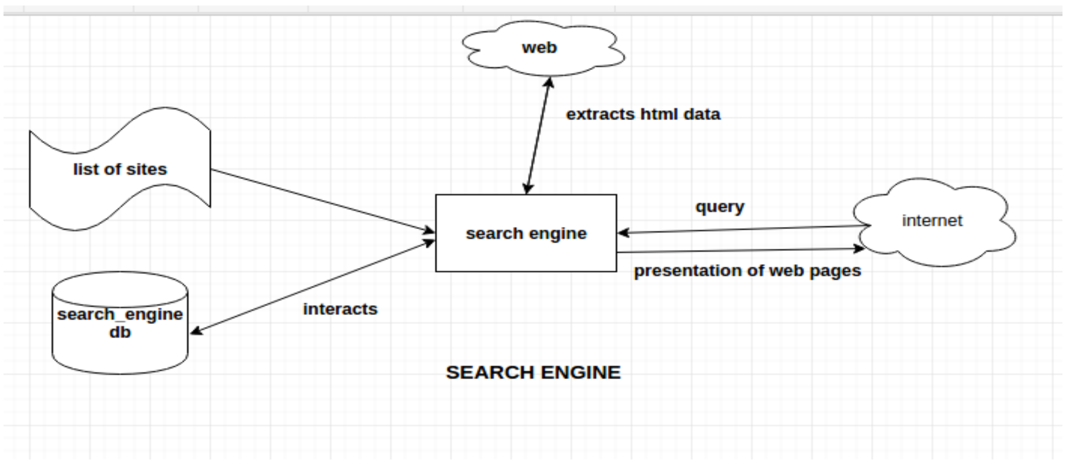

Figure 5.1: Search Engine Architecture

5.2 Crawling

- **Crawling**: Crawlers look at web-pages and follow links on those pages. They go from link to link and bring data about those web-pages. It takes the input as list of site links and crawl all the urls in DB table URL_INFO which contains the information about all the links and also downloads the html data for further requirements
  - **Input:**List of Site urls
  - **Output**: URL_INFO table corresponding to all the urls in that site and html files
  - **Processing**: It starts from site url and extract all links on that site url page and then follow those links to do the same. It crawls till all the urls from that page get crawled
  - **Files used** : url_info.py, site_info.py

- **Python Libraries used** : urllib2, urlparse, re, time, robotparser, sqlite3, bs4, whois, datetime
- **Database Tables used** : URL_INFO,SITE_INFO

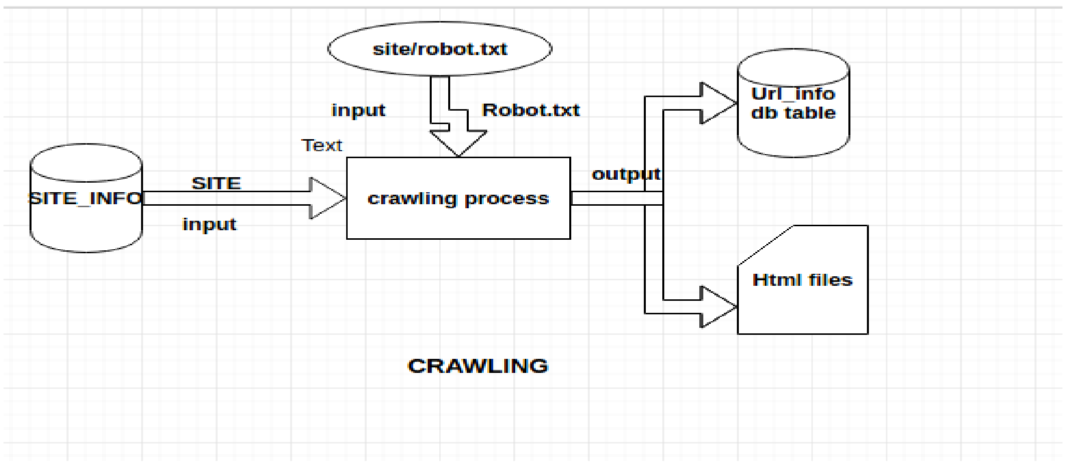

Figure 5.2: Crawling Architecture

- **Updating the tables** : Site urls, url data etc are often modified and new links are also added. So, it is important to update our tables and html data. For this, a new python file has been created which compares the "last_updated" column in URL_INFO table with current last_updated and modify the relevant tables.
  - **Input**: "last_updated", "url_link" column in URL_INFO table
  - **Output**: updated table
  - **Processing**: It compares on the basis of last_modified and update the tables
  - **Files used** : url_info.py, site_info.py, keyword_info.py
  - **Python Libraries used** : urllib2, urlparse, re, time, robotparser, sqlite3, bs4, whois, datetime
  - **Database Tables used** : URL_INFO, SITE_INFO, KEYWORD_INFO

5.3 Indexing

- **Indexing**: It includes various methods for indexing the contents. Html text, title, meta description/keywords has been used for extracting the keywords and make the table KEYWORD_INFO. In this table, occurrence of keywords has been also taken. In this step, html tags were remo
  - **Input**: html data downloaded during crawling

- **Output**: KEYWORD_INFO table
- **Processing**: html data are processed to get text and then tokenized to get keywords
- **Files used** : keyword_info.py
- **Python Libraries used** : os, urllib, bs4, nltk, collections, sqlite3
- **Database Tables used** : KEYWORD_INFO

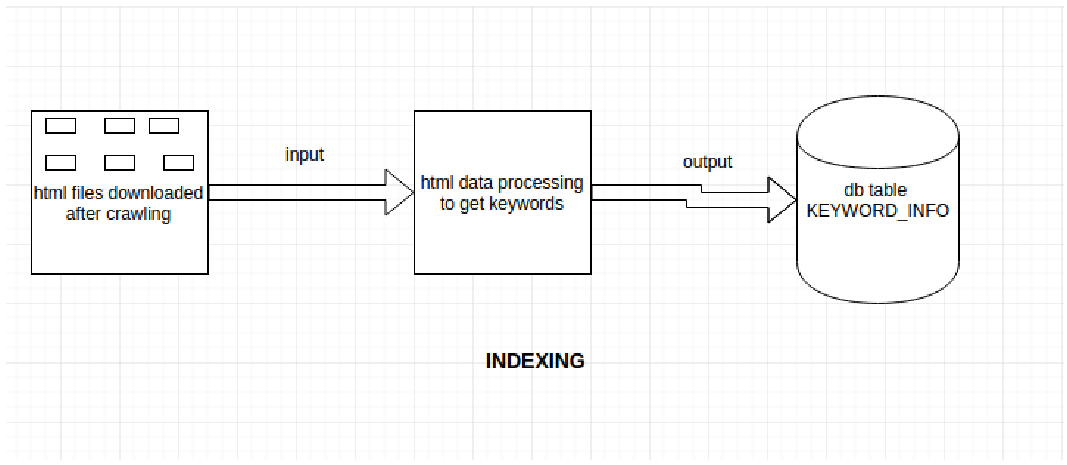
html files downloaded after crawling
input
html data processing to get keywords
output
db table KEYWORD_INFO
INDEXING

Figure 5.3: Indexing Architecture

5.4 Ranking

- **Ranking**: Relevant urls containing query keywords are ranked using different factors. In this step, we take find the urls in which query keywords are present and then based on the weight of url, site and occurrence weight, urls are ranked. Site having good reputation, more number of viewers, good links and other factors are given more weight-age compared to others sites and same goes for urls. We have also made a weight table that contains the weight of different parameters and weight factors which can be modified as per requirements.
  - **Input**: html data downloaded during crawling
  - **Output**: KEYWORD_INFO table
  - **Processing**: html data are processed to get text and then tokenized to get keywords
  - **Files used** :
  - **Python Libraries used** :
  - **Database Tables used** : KEYWORD_RANK

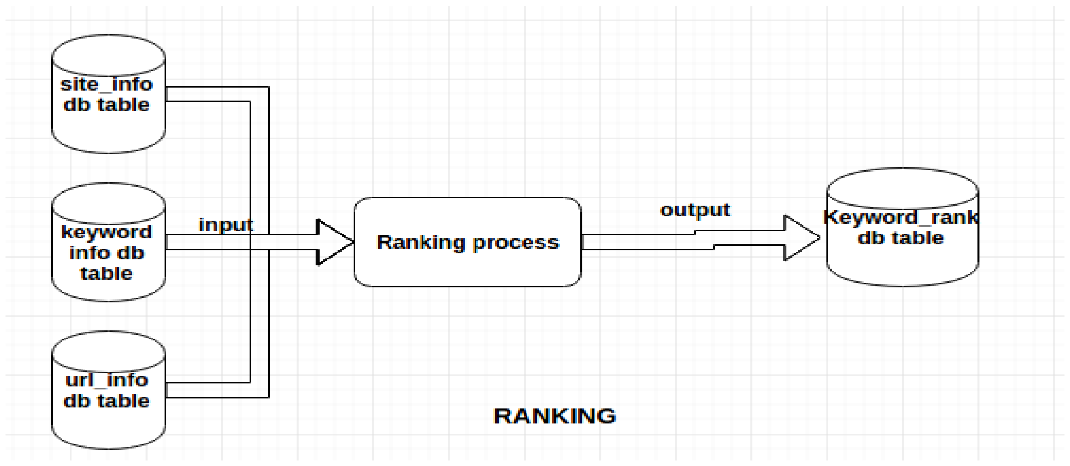

Figure 5.4: Ranking Architecture

5.5 Presentation

- **Presentation**: Displaying the links with title and some code snippet based on the result obtained after ranking. This step will take help from front-end and back-end web developing. There will be a html page containing search form which will take input from the user and based on the back-end processing, it will display the result.

- While presenting, the keywords searched is highlighted(by replacing the keyword with its bold (<b>keyword </b>) form in each of the result ed links and their description.

  - **Input**: html data downloaded during crawling
  - **Output**: KEYWORD_INFO table
  - **Processing**: html data are processed to get text and then tokenized to get keywords
  - **Files used** :
  - **Python Libraries used used** :
  - **Database Tables used** :

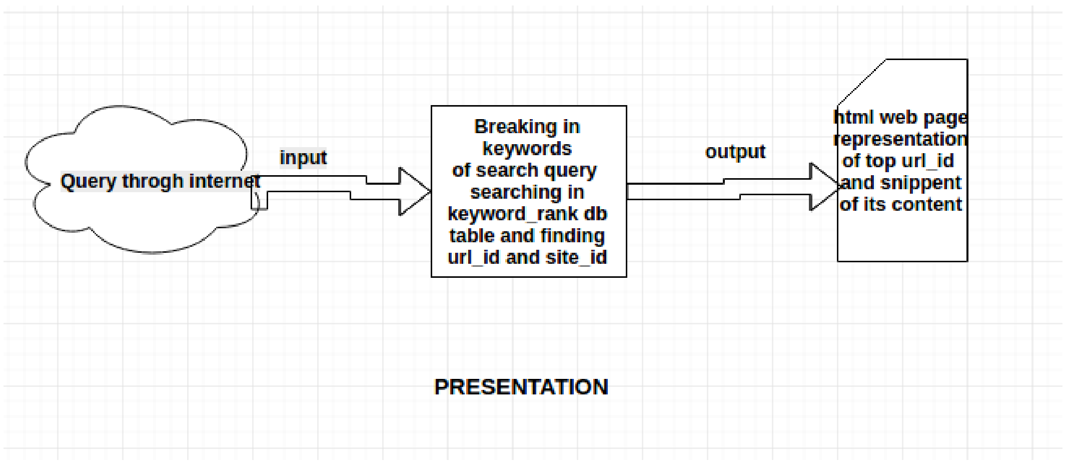
Query throgh internet
input
Breaking in keywords of search query searching in keyword_rank db table and finding url_id and site_id
output
html web page representation of top url_id and snippent of its content
PRESENTATION

Figure 5.5: Presentation Architecture

6 Design

6.1 Search Engine Working

- The search form takes the query from user and stores it in a variable.
- The variable will be tokenized using nltk tokenizer and each obtained keyword will be separately searched in KEYWORD_INFO table.
- Based on the search, a list of url_id, site_id and corresponding weight based on the occurrence will be taken from KEYWORD_INFO table. Also, weight for these url_id and site_id will be taken from URL_INFO and SITE_INFO table.
- These weights for each url will be added and these urls will be ranked on the basis of total weight.
- Urls with some code snippet will be shown on the result page.

6.2 Different python files and their input and output:

- **weight_config.py**
  - **Input:-**values for different weight factors
  - **Output:-**Database table WEIGHT_CONFIG
- **site_info.py**
  - **Input:-**list of sites.
  - **Output:-**Database table SITE_INFO
- **url_info.py**
  - **Input:-**Site_link from the table SITE_INFO..
  - **Output:-**-Database Table URL_INFO
- **Keyword_info.py**
  - **Input:-**:url_link from the table URL_INFO.
  - **Output:-**Database Table KEYWORD_INFO
- **Keyword_rank.py**

  - **Input:-**database tables (KEYWORD_INFO, URL_INFO, SITE_INFO) .
  - **Output:-**Database Table KEYWORD_RANK

- **Update.py:**
  - **Input:-**last_updated parameter from URL_INFO table.
  - **Output:-**update the corresponding columns in different tables

7 Yet to be implemented

- There are many factors while deiding ranking of URL are not implemented. They are:-
  - Broken link
  - No. of viewers of a URL
  - Site Popularity
    - Based on the search query a new table can be created which will store the stat about search query and should be updated after each query.
  - Bounce Rate
- content quality factors
  - check whether alt attribute is present or not in the img tag of web page of each scrolled URL. If it is not present then give some penalty factor.
  - check whether Title attribute is present or not in the img tag of web page of each scrolled URL. If it is not present then give some penalty factor.
  - check whether **rel="no follow**"is present or not in the anchor tag of web page of each scrolled URL. If it is not present then give some penalty factor.