# UIC Web Search Engine

Pavana Doddi
Computer Science
University of Illinois at Chicago
pdoddi2@uic.edu

## ABSTRACT

A web search engine or Internet search engine is a software system that is designed to carry out web search (Internet search), in a systematic way for particular information specified in a textual web search query. The search results are generally presented in a line of results, often referred to as search engine results pages (SERPs). Search results can range from zero to arbitrarily large values, hence it's important to display relevant results in first page of the search. Identification of keywords and removal of stop-words also plays an essential role in reducing the noise of the search results.

The objective of this project is to build a web search engine for the UIC domain. The search engine consists of three main components: 1. Web Crawler, 2. Vector Space Model and
3. Page Rank Algorithm. Web search engines get their information by web crawling from site to site. Web crawler is implemented in parallel by using the Queue module to access the resources in a synchronized way. Vector Space Model is used to represent text documents (and any objects, in general) as vectors of identifiers, such as index terms. PageRank works by counting the number and quality of links to a page to determine a rough estimate of importance of the website.

Page Rank Algorithm built for the UIC domain extracts information from all the URL's of the UIC. The collected data includes texts, phrases from the valid HTML tags, URL's pointed by and to the current URL. The related data is crawled and indexed and stored in the filesystem. When users perform a search query by specifying the search term, the related query is transferred to the search engine index and optimization algorithms are performed to retrieve SERPs.

## 1 COMPONENTS

### 1.1 Web Crawler

A Web crawler starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the pages and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. They are one of the main components of web search engines, systems that gathers set of web pages, indexes them thereby allowing users to query against the index and obtain search results.

The web crawling is implemented using multi-threading (in-order to improve the performance and reduce time). The crawling starts from the UIC CS domain: https://www.cs.uic.edu/ BFS strategy is used to implement the crawler: is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key, and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. All pages are placed in a queue and then Every page is dequeued (removes the top), downloaded and parsed using the *HTMLParser* library, the links are extracted and checked if they belong to UIC domain and then added to FIFO queue if it's of appropriate format. In-order to reduce the noise of crawling and time complexity a backlist of all bad formats is used which contains the following extensions: *".avi", ".doc", ".docx", ".gif", ".jpg", ".jpeg", ".mp4", ".png", ".pdf", ".gz",".rar", ".tar", ".tgz", ".exe", ".zip", ".js"*. These are neglected by the crawler. Also as a part of preprocessing, after extraction of URL: http is replaced by https and query strings, path parameters and in links are removed from the URL. As a part of crawling, two dictionaries, URL and out links of URL is concurrently saved to the disk. And HTML data is also saved on to the disk.

### 1.2 Preprocessing

Preprocessing of crawled pages is essential to remove the noise of search results and also improve the performance of search engine. Preprocessing involves 3 steps: 1. Removing unnecessary html tags, 2. Removal of stop words and 3. Stemming.

Plain text is extracted from saved html data and *<script> <style>* are ignored. This is done by using *beautiful soup (more in software specification).* A stop word is a commonly used word *(such as "the", "a", "an", "in")* that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. The page is then tokenized, the tokens are stemmed using the *PorterStemmer:* is a process for removing the commoner morphological and inflexional endings from words in English.

### 1.3 Vector Space Model

Vector space model or term vector model is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as, for example, index terms. It is used in information filtering, information retrieval, indexing and relevancy rankings. Documents and queries are represented as vectors. Each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero.
All words in the document are stored along with the frequency (no of times it appears in the document). Document frequency is also stored in inverted index. Then TF-IDF score of each word in the document is calculated and stored in another vector model. TF-IDF score reflects how important a word is to a document in a collection or corpus. The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

### 1.4 Similarity Measure

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. Cosine similarity is calculated using,

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

Inverted index constructed above, is used to find the limited set of documents by computing cosine similarity of each indexed document as query words are processed one by one. Total score is computed for each retrieved document and stored in a dictionary *(key- URL, value- score)*. Dictionary is sorted based on cosine similarity score and returned in descending order of their relevance.

### 1.5 Page Rank Algorithm (Query Dependent)

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites. A PageRank results from a mathematical algorithm based on the web graph, created by

all World Wide Web pages as nodes and hyperlinks as edges. The rank value indicates an importance of a particular page. A hyperlink to a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it ("incoming links").
Page rank algorithm is run iteratively based on the in-links of a page until it converges. S(A) is initialized to 1/|V| where V is set of all web pages. Until ranks do not converge, for each A belongs to V

$$S(A) = \left[(1-\epsilon)\sum_{B\to A}\frac{S(B)}{out(B)}\right] + \frac{\epsilon}{n}$$

where ε s.t $0<\varepsilon<1$. S(A) models the probability that this random surfer will be on page A at any given time. By using Page Rank Algorithm, the SERP's can be effectively ranked by importance and thereby increasing the efficiency of search algorithm. Order of the page rank is query independent.

To make it more efficient a model based on a **query-dependent PageRank** score of a page which as the name suggests is also a function of query. When given a multiple-term query Q= {q1, q2, ..}, the surfer selects a q according to some probability distribution P(q), and uses that term to guide its behavior for a large number of steps. It then selects another term according to the distribution to determine its behavior, and so on. The resulting distribution over visited web pages is QD-PageRank

### 1.6 User Interface

User Interface is designed in a simple way as with any other search engine. A text box is placed where the user can enter the search query and a search button is placed. As user enters the search query and clicks on the button search results are displayed on screen. Relevant results appear on the top of the page and when user clicks it, the user is redirected towards the result page. Search results are limited to a maximum of 50 and displayed across 5 pages. If there are no results for a particular query then a result not found message appears.

## 2 SOFTWARE SPECIFICATION

The search engine is designed on Python3 using Object Oriented Programming techniques to make it readable and efficient.

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It is used in ***webcrawler.py*** module to parse HTML tags and remove style and script tags. The following commands can be used to import beautiful soup,

```
from bs4 import BeautifulSoup
soup = BeautifulSoup (html_doc, 'html.parser')
```

Natural Language Toolkit(NLTK) library is used in ***preprocessing.py*** module to implement the porter stemmer algorithm. The following command can be used to implement it,

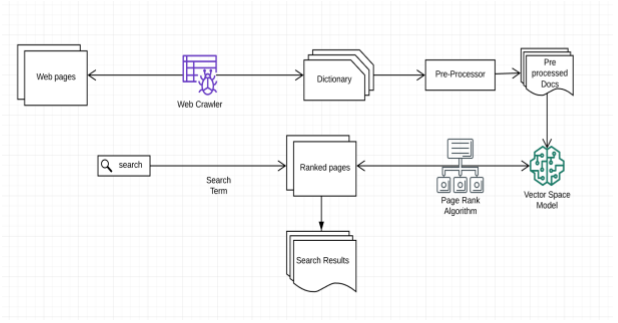
```
from nltk.stem.porter import *
```

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language. PyQt5 library is used for implementing user interface.

The other modules include ***pagerank.py***: implements the page rank algorithm and ***main.py***: launches the search engine and integrates it with other modules.

## 3. ARCHITECTURE DIAGRAM

The following represents the architecture diagram of search engine.



Web crawler crawls the web pages and generates dictionary consisting of url and out-links of the URL. It's the passed to pre-processor. From pre-processed data, vector space model models the data as vectors for the page rank algorithm to parse the data and rank the pages according to relevance. When user enters the search term, relevant pages are retrieved and displayed.

## 4 EVALUATIONS

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances. Precision must be ideally high for a search engine. Evaluation results for 4 search terms (***query independent page rank vs query dependent page rank***) are as given below,

- Search term: "computer science", the precision for QD-PR is 0.6 and PR is 0.9
- Search term: "artificial intelligence", the precision for QD-PR is 0.7 and PR is 0.9
- Search term: "campus care", the precision for QD-PR is 0.6 and PR is 0.7
- Search term: "UI health", the precision for QD-PR is 0.5 and PR is 0.8

## 5 FUTURE WORK

Query dependent PR can be tweaked further based on multiple query terms and probability function can be reviewed to make the precision of the model better. Crawling can be enhanced by trying to omit pages which return 404 errors. Time to crawl pages have been already enhanced by using multi-threading techniques, but as a future scope it can be enhanced further.

## 6 REFERENCES

1. https://www.nltk.org/
2. https://www.crummy.com/software/BeautifulSoup/bs4/doc/
3. https://www.youtube.com/watch?v=nRW90GASSXE&list=PL6gx4Cwl9DGA8Vys-f48mAH9OKSUyav0q
4. https://pypi.org/project/PyQt5/
5. https://www.geeksforgeeks.org/page-rank-algorithm-implementation/