Web Search Engine

CS-582 INFORMATION RETRIEVAL

SAI BHARATH KUMAR BANDI
University of Illinois at Chicago
sbandi3@uic.edu

ABSTRACT

In today's expeditious world, the technology is growing rapidly day by day. In the current emerging technologies search engine is one of the trending topics. There are many web search engines outside, out of which Google search engine stand out to be the best in the market. Google search engine has used different powerful algorithms to improve the performance in retrieving most relevant websites or webpages for a searched query. One of the old algorithms used by google is PageRank.

This is a report for the final project of course CS-582 – Information Retrieval at the university of Illinois at Chicago. The objective of this project is to implement a web search engine that displays most relevant websites when given a query. Different metric calculations and algorithms are implemented and compared based on performance. This project consists of web crawling, data preprocessing, calculation of similarity metrics and display of most relevant pages.

CONCEPTS OF IR

• Vector Space model • IR system • Word graph • PageRank • Cosine Similarity

KEYWORDS

Links, In-bound measure, Out-bound measure, TFIDF, Crawling, PageRank

INTRODUCTION

This search engine is built on the domain www.uic.edu, which retrieves pages that belong to uic.edu domain. The program is written in Python3 in modular format for easy understanding and it was written in two files Spidering.py and Pagerank.py using Spyder IDE. In the first of the program a web crawler is built that fetches the links from different websites. In this project, the web crawler is used to fetch website URLs from https://www.cs.uic.edu/. A limit of 3050 websites is set for the web crawler because of billions of websites present in the world wide web. Now the html documents for the corresponding retrieved link is extracted, preprocessed, indexed and similarity measures has been calculated. The crawled files have been stored in the path '/spideredFiles' zipped folder attached to the homework submission. The components used in this project are

CRAWLER

The web crawler is implemented in spidering.py file. At first a subject URL which is the main URL is considered and is pushed into a queue. The main URL is https://www.cs.uic.edu/. A list of URLs is maintained for checking repeated URLs. If we have a URL which is repeated in the queue, then the crawler is end up running in a infinite loop fetching the same website again and again. Now each document of the website in the queue is retrieved in the fashion of Breadth First Search algorithm. The HTML document of the website is retrieved using BeautifulSoup function of python. Now the HTML document is read and the whole text present in the HTML document is stored in a file named as '/document <number>' in '/spideredFiles' folder. The text of the document is now considered, and the tags belong to <a> is considered and the URLs are retrieved. While retrieving the URLs the net location of the URL is considered and appended with http and requested using http port. If an error occurred while retrieving the URL, and exception is written to handle the error. Now the URL is parsed and checked whether is it is in the domain. Moreover, if the URL contain the formats that are not required such as '.css', '.doc', '.exe', '.gif', '.jpg', '.jpeg', '.mp3', '.avi', '.mpg', '.mpeg', '.pdf', '.png',

'.ram', '.rar', '.tiff', '.wav', '.zip' are filtered. Now if the URL is not retrieved previously, it is pushed into the queue. This process is continued until 3050 URLs from the uic.edu domain is retrieved. The links are dumped into a pickle and stored in 'links' file.

DATA PREPROCESSING

The website links present in the 'links' file are loaded into Pagerank.py file. With the help of document file name present in '/spideredFiles' folder, each is file looped and the text from each document is retrieved. The text obtained from the document is read and the text present in the tags such as script, noscript, style, meta, title, header are filtered and the remaining text is subjected to preprocessing – lower case, stemming each word, removing of stop words, removing punctuations, tokenization, removing words of one or two characters, removing the SGML tags. Now the obtained text is tokenized and stored. This part of the program is present in 'preprocessing' function of Pagerank.py file

COSINE SIMILARITY METRIC

To find the cosine similarity between pages and queries, we need two important components TF (term frequency), IDF (Inverted document frequency). At first, we consider the term frequency of the word. While preprocessing the text from each document retrieved, the words present in each document is tokenized and stored in a dictionary. Now this dictionary is iterated, and the words present in all the documents are stored separately and the frequency of the word occurred in each document and also the frequency of the files in which the word occurred are stored. So, the term frequency will be the number of times the word occurred in a document. Secondly, the IDF of the word is calculated using the document frequency of the word which is calculated while preprocessing the document. The IDF value of word:

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Now TF-IDF of the word is calculated by multiplying the term frequency (TF) and the Inverted Document Frequency (IDF).

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

The TF-IDF values of each word is used to calculate the TF-IDF value of file individually by aggregating the squares of TF-IDF values of words present in that file and calculating the square root. Now a query is taken from the user as an input and it has been preprocessed. The steps followed in the preprocessing section of the documents are followed exactly for the query. The term frequency of the query is calculated and the inverted document frequency of each word in query is considered from the inverted document frequency of the word calculated from the documents. Now the cosine similarity of the pages according the query given is calculated using:

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The cosine similarity scores of the pages according to the given query are sorted and the top scored links are displayed as the most relevant links of the given query according to cosine similarity.

PAGERANK METRIC AND THE INTELLIGENT COMPONENT

The PageRank for the websites retrieved is calculated in different ways. Here, the website links present in 'links' file are extracted and the in-bound links and out-bound links are calculated for every link by again extracting the HTML document of each link. If the link present in a document, then that link is stored as an out-bound link and the document's link is added as in-bound link. Moreover, the in-bound count and out-bound count is calculated for every link and stored. The PageRank score for all the nodes (links) are initialized as PR(link) = (1/number of links) for each link. The PageRank for link u is calculated using:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)},$$

Where B_u is the web graph. The PageRank is calculated for every link for about 25 iterations to make the values become stable. A damping factor = 0.85is used, so the PageRank for the URL P_i will be:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

The PageRank for each URL is calculated and stored. Before calculating the final rank for each page compared to the given query, we must calculate other components from the data we have. The extracted links are considered, and the text is extracted again. The text present in the title of the document also plays an important role in calculating the relevant documents. So, the text present in the title tags of the document content are extracted and subjected to preprocessing as done for the documents and query before. With obtained token information of the titles of the links, we calculate the term frequency and inverted term frequency of the titles. TF-IDF for the tokens present in the title are calculated. Likewise, it is also calculated for words present in the URLs. After that, the lowest, highest and the average values of term frequency of each word of query that present in the document and title are calculated. Also, the lowest, highest, and average value of the TF-IDF values are also calculated for the documents. Now with all the values present, we calculate the final rank of the page by giving weightage to all values and aggregating all the score. The different TF-IDF values are given considerable weightage and the frequency of each word of query present in the document are given 0.01 weightage. The files which covered most of the query words are given high weightage. Moreover, the PageRank, in-bound and out-bound scores of each link plays an important role. The obtained results are sorted and the top links with highest score are considered as the most relevant links for the given query. This part of the program is present in Pagerank.py file in sections.

MAIN CHALLENGES

The main challenges faced while building the entire code are:

1. While reading the anchor tags present in each URL, some of the anchor tags are reference URLs where there is no net location or http portal. For example, the URLs like 'Click Here', 'here' are stuck while filtering the URLs and filtering the excluded formats such as '.mp3', '.zip'
2. Retrieving the text part of the html document which is present only in the body tag excluding the text present in tags such as script, noscript, header etc.
3. Since the number of retrieving URLs are 3050, the IDE takes plenty of time for retrieval and preprocessing the document. So, I have used google colab pro for extracting the files and preprocessing the document.
4. Finding the required python libraries for parsing and extracting the URLs and document text.
5. Assigning weightages to all the components for calculating the PageRank

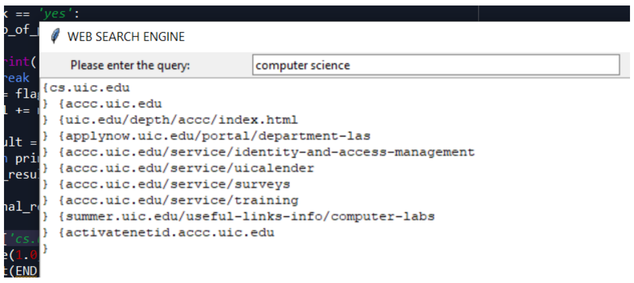
WEIGHTING SCHEMES and ALTERNATIVES:

As we are considering two different approaches of weighting schemes in this project which are cosine similarity and PageRank score. The calculation of both measures are explained in detail in the sections 'Cosine Similarity' and 'PageRank'. The two measures are compared based the recall and precision values of the relevant documents for the given query. The relevance of the URL to the given query is labeled manually. By comparing, the results of PageRank are better than the Cosine similarity results. The recall and precision values are mentioned in the Results part of this report. The other method for calculating the similarity by creating n_grams and phrases occurrences using the tokens obtained from the documents. Here we will calculate the mean reciprocal score of documents and calculate the similarity. This gives us some more better results compare to cosine similarity. The alternative methods for PageRank and cosine similarity are TrustRank, MOZ and Majestic, which gives us even better results among vast amount of web links present in the world wide web.

EVALUATION, RESULTS AND ANALYSIS

Five different queries are considered evaluation and the relevant web links retrieved based on the cosine similarity and PageRank are calculated and compared. The best relevant links retrieved from page rank are

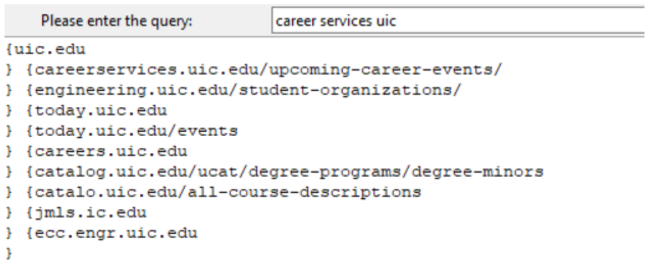
1. **Computer Science accc**
   The retrieved links according to PageRank are:



```
WEB SEARCH ENGINE
Please enter the query: computer science
{cs.uic.edu
} {accc.uic.edu
} {uic.edu/depth/accc/index.html
} {applynow.uic.edu/portal/department-las
} {accc.uic.edu/service/identity-and-access-management
} {accc.uic.edu/service/uicalender
} {accc.uic.edu/service/surveys
} {accc.uic.edu/service/training
} {summer.uic.edu/useful-links-info/computer-labs
} {activatenetid.accc.uic.edu
}
```

The retrieved results from the program for the query Computer Science is listed in the above figure. If we can label the retrieved data manually then the recall of the results will be 60%, which is better result

2. **Carrer services uic**
   The retrieved links according to PageRank are:

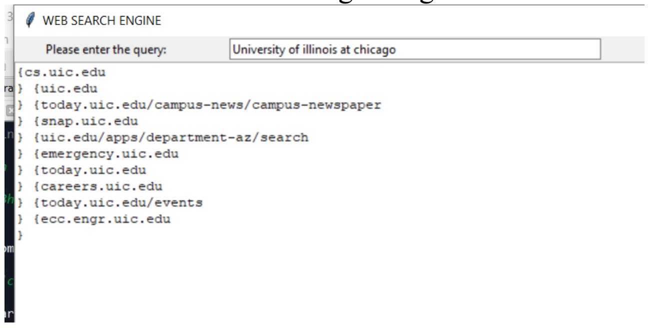


```
Please enter the query: career services uic
{uic.edu
} {careerservices.uic.edu/upcoming-career-events/
} {engineering.uic.edu/student-organizations/
} {today.uic.edu
} {today.uic.edu/events
} {careers.uic.edu
} {catalog.uic.edu/ucat/degree-programs/degree-minors
} {catalo.uic.edu/all-course-descriptions
} {jmls.ic.edu
} {ecc.engr.uic.edu
}
```

The retrieved results from the program for the query Carrer services uic is listed in the above figure. If we can label the retrieved data manually then the recall of the results will be 50%.

3. **University of illinois at Chicago**

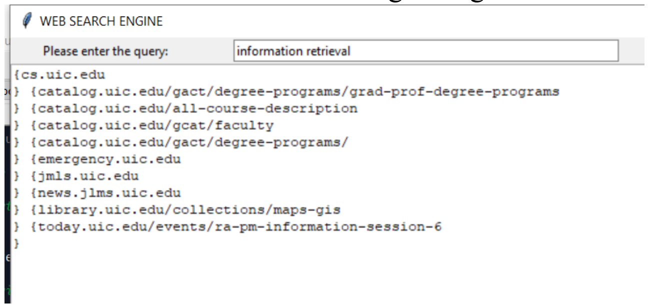
The retrieved links according to PageRank are:



```
WEB SEARCH ENGINE
Please enter the query: University of illinois at chicago
{cs.uic.edu
} {uic.edu
} {today.uic.edu/campus-news/campus-newspaper
} {snap.uic.edu
} {uic.edu/apps/department-az/search
} {emergency.uic.edu
} {today.uic.edu
} {careers.uic.edu
} {today.uic.edu/events
} {ecc.engr.uic.edu
}
```

The retrieved results from the program for the query is listed in the above figure. If we can label the retrieved data manually then the recall of the results will be 70%.

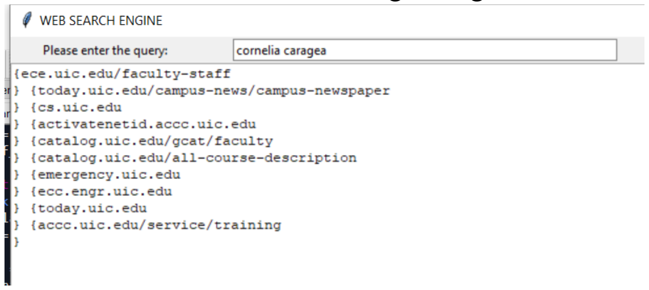
4. **information retrieval**
   The retrieved links according to PageRank are:



```
WEB SEARCH ENGINE
Please enter the query: information retrieval
{cs.uic.edu
} {catalog.uic.edu/gact/degree-programs/grad-prof-degree-programs
} {catalog.uic.edu/all-course-description
} {catalog.uic.edu/gcat/faculty
} {catalog.uic.edu/gact/degree-programs/
} {emergency.uic.edu
} {jmls.uic.edu
} {news.jlms.uic.edu
} {library.uic.edu/collections/maps-gis
} {today.uic.edu/events/ra-pm-information-session-6
}
```

The retrieved results from the program for the query information retrieval is listed in the above figure. If we can label the retrieved data manually then the recall of the results will be 50%.

5. **cornelia caragea**
   The retrieved links according to PageRank are:



```
WEB SEARCH ENGINE
Please enter the query: cornelia caragea
{ece.uic.edu/faculty-staff
} {today.uic.edu/campus-news/campus-newspaper
} {cs.uic.edu
} {activatenetid.accc.uic.edu
} {catalog.uic.edu/gcat/faculty
} {catalog.uic.edu/all-course-description
} {emergency.uic.edu
} {ecc.engr.uic.edu
} {today.uic.edu
} {accc.uic.edu/service/training
}
```

The retrieved results from the program for the query information retrieval is listed in the above figure. If we can label the retrieved data manually then the recall of the results will be 60%.

As per the results we can say that PageRank scores has retrieved the relevant results better with an average recall of 60%. Some of the results retrieved links might not be as expected but the algorithm did get a good recall and if we include further work the algorithm, it might retrieve even better results for the given query.

DISSCUSION ON RELATED WORK:

There are many algorithms for search engine in the market and a high-level research is currently performed in top multi-billion companies such as google, apple. Researched different algorithms among the programs available in the market and the basic algorithm that gives better results is PageRank. So, researched different websites and papers for implementing the algorithm and programmed according.

FUTURE WORK:

There are improvements for PageRank algorithm for better results such as MOZ, Majestic and more. Currently Alexa rating measure stands out in the market. For the project implemented in this report we can further improve the algorithm by calculating the n_grams, phrases comparison and the mean reciprocal ranks for the pages and retrieve better results for the given query.

HOW TO RUN THE CODE:

Please use a python IDE for running the code. First run the file Spidering.py for crawling, the crawled files will be stored in '/spideredFiles' folder and then run PageRank.py file for the ouput.

ACKNOWLEDGMENTS

I would like to thank Professor Cornelia Caragea for this opportunity and allowing me to research on different algorithms present in the market and implementing the best algorithm in the emerging field of Information Retrieval and for her guidance on this.

REFERENCES

1. Christopher D. Manning, Hinrich Schütze, and Prabhakar Raghavan, "Introduction to Information Retrieval" [Online]. Available: https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf.

2. W. Xing and A. Ghorbani, "Weighted PageRank algorithm," Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004., Fredericton, NB, Canada, 2004, pp. 305-314, doi: 10.1109/DNSR.2004.1344743.

3. "Beautiful Soup Documentation" [Online] Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

4. Monica Bianchini, Marco Gori, Franco Scarselli "Inside PageRank" [online] Available: https://dl.acm.org/doi/abs/10.1145/1052934.1052938

5. Juan Ramos "Using TF-IDF to Determine Word Relevance in Document Queries" [online]