Search Engine for UIC Domain

ASHWIN DESHPANDE, University of Illinois at Chicago, USA

This report summarizes a search engine project developed as a part of CS 582 Information Retrieval Project. In here I provide detailed information regarding three modules that summarize the search engine. This projects consists of a Crawler responsible for downloading characteristic information of web pages in the UIC domain, the methods used to represent text data, a search engine that outputs a ranked list of query specific web pages and Topic-Specific PageRank Algorithm that obtains re-ranks results according to their page authority.

CCS Concepts: • **Information systems** → **Retrieval models and ranking**; **Probabilistic retrieval models**; **Language models**;

Additional Key Words and Phrases: BERT features, Search Engine, Cosine Similarity, Web Crawling

ACM Reference Format:
Ashwin Deshpande. 2021. Search Engine for UIC Domain. 1, 1 (May 2021), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This search engine consists of several important components: crawling, indexing, Search Engine and ranking. Each of these are explained in this report successively as each next component requires the previous for its functioning. This crawler starts from https://cs.uic.edu/ and obtains the information that is useful for identifying the web-page. After page has been indexed, vectors representing the document is obtained and stored (We examine several word embedding methods). The search engine then retrieves a list of web-pages most similar to the given query based on cosine similarity measure(Other measures are explored) between query embedding vector and the embedding vector of all documents. Obtained results are ranked according to PageRank. Finally we will see evaluation of all methods. It was found that a Scrapy architecture with BeautifulSoup text extraction methods for crawling along with TF-IDF for data-specific tokens and BERT for semantically similar sentences performed the best. For more details: https://github.com/ashwinkd/Search-Engine

2 CRAWLING

For our project we have implemented two crawlers. One using BeautifulSoup and another using Scrapy. But the best performing crawler is given by a combination of the two. In the first crawler queuing web-pages, filtering and robots exclusion protocol, page deduplication was implemented from scratch. This approach gives fine-grained control over the crawling process. The URL canonicalization process was implemented by checking the content of the webpage. Given a URL this process fetches first 1024*1024 bytes of webpage data and saves the md5 hash key in a list. If a new URL's hash key is already present in our visited URLs list we do not scrape this page for more data. This method is accurate and allows for re-indexing web-pages that have been changes. However, this method is time consuming since

Author's address: Ashwin Deshpande, adeshp27@uic.edu, University of Illinois at Chicago, 851 S Morgan St, Chicago, Illinois, USA, 60607.

Manuscript submitted to ACM

it is possible to implement a reasonable dedup using the URL itself. Breadth-First Search approach is implemented in this crawler. Scrapy framework implements the underlying crawler architecture with basic features. Scrapy was chosen because it is possible to run crawling concurrently on multiple pages. To scrape data from the page. During the crawling all outlink from a page is saved as a graph for Topic Specific Page Rank Algorithm. The text surrounding the outlink is saved and indexed for the outlink page. By the end of crawling 7640 pages were scraped and 98,267 page links were stored in the graph.

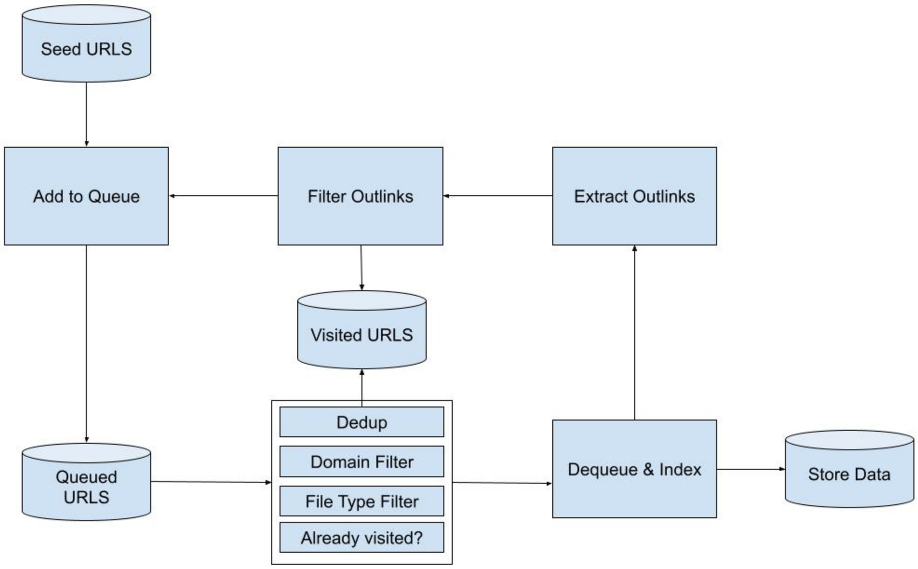


Fig. 1. Crawling Process

3 DATA REPRESENTATION

After crawling process we have obtained following data on each page:

- URL
- Title
- Body
- Out-links

We use the Title and Body for running our search Algorithm. Title and Body are cleaned and tokenized using NLTK word tokenizer. These tokens and their occurrences are saved in an inverted index along with their inverse docucment frequency. These are later used for TF-IDF algorithm.

We also develop vector space representation of each document(web-page) by encoding each text item into a 768 length BERT embedding. Both TF-IDF and BERT embeddings are calculated once and stored. These can be re-used in successive runs.

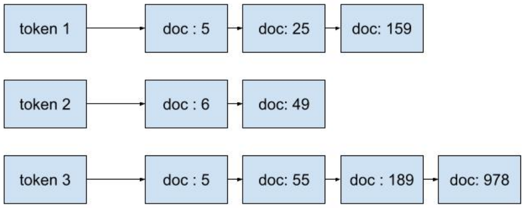


Fig. 2. TF-IDF Index

4 SIMILARITY AND RANKING

Given a query results are obtained in TF-IDF algorithm by using cosine similarity measure. Similarly query text is encoded into a similar 768 length long vector and ranked according to its similar with all document vectors. Other similarity measures such as euclidean distance as well manhattan were explored. Evaluation against gold-standard cosine simlarity performed the best with average Spearman's Rank Correlation: $\rho = 0.74$

5 SEARCH ALGORITHM

TF-IDF algorithm was best at obtain results that consisted of exact query words. Therefore webpages with unique tokens such as names or numbers were obtained using TF-IDF. However, TF-IDF does not consider the context in which the words occur or their order of occurence. For example a query with text *Natural Language Processing* obtained results on Natural Gas. To solve this problem BERT embeddings were used. Since these embeddings encode semantic meaning in vector space the results were good at obtaining semantically similar documents. BERT embeddings only encode english words therefore are not best at query consisting of only names or numbers. Therefore the best solution was to use both algorithms to obtain top 20 most similar documents half of which are obtained by either algorithms. Using simlar BERT similarity we obtain 5 similar pages for each of the 20 documents obtained earlier.

5.1 Topic Specific PageRank Algorithm

Since PageRank Algorithm is best at obtaining a general rank for all webpages, it wouldn't obtain the best result if we ranked obtained results using their generic PageRank scores. Therefore, a topic specific PageRank algorithm is developed such that the probability of jumping from a dead-end is distributed not uniformly across all web-pages but a smaller set of webpages that was obtained earlier by our TF-IDF and BERT algorithm. Further, among the set of webpages that PageRank can jump to the probability is distributed according the cosine scores so that higher ranked pages get more importance.

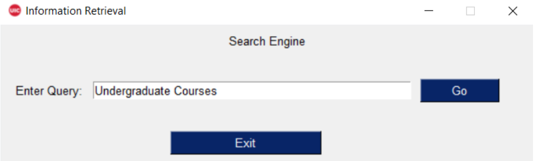
$$A_{ij} = \begin{cases} \beta M_{ij} + (1-\beta)/S & \text{if } i \in S \\ \beta M_{ij} & otherwise \end{cases} \tag{1}$$

6 RESULTS

To evaluate the results Spearman's Ranked Correlation Coefficient ρ and recall is calculated. For a gold-standard manually built target dataset following were the results:

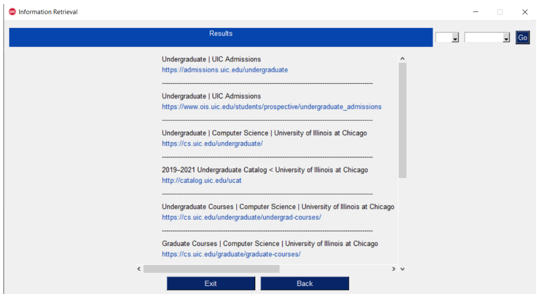
query	ρ	p	Recall
architecture	0.65	0.0425	0.5
coronavirus covid-19 pandemic	0.58	0.074	0.9
undergraduate research computer science college	0.1	0.85	0.7
mathematics department professors and faculty	0.94	0.0006	0.7
jobs internship employment and other opportunities	0.71	0.022	0.7

We see that queries with more descriptive and longer length give better results.



Enter Query: Undergraduate Courses

Fig. 3. Screenshot main page: query: Undergraduate courses



Undergraduate | UIC Admissions
https://admissions.uic.edu/undergraduate

Undergraduate | UIC Admissions
https://www.ois.uic.edu/students/prospective/undergraduate_admissions

Undergraduate | Computer Science | University of Illinois at Chicago
https://cs.uic.edu/undergraduate/

2019–2021 Undergraduate Catalog < University of Illinois at Chicago
http://catalog.uic.edu/ucat

Undergraduate Courses | Computer Science | University of Illinois at Chicago
https://cs.uic.edu/undergraduate/undergrad-courses/

Graduate Courses | Computer Science | University of Illinois at Chicago
https://cs.uic.edu/graduate/graduate-courses/

Fig. 4. Screenshot main page: query: Undergraduate courses

7 CONCLUSION

System is able to fetch results consisting of both data specific tokens as well as those results that have semantically equivalent words. The method of incorporating both algorithms can be further fine tuned. Currently the results of both methods are spliced into the final result to give equal importance to both. However this can be improved such that when both algorithms output same result the ranking is upgraded. In addition, the sample size collected from the UIC domain was smaller and hence some queries do not obtain results that we may see of Google Search. Further work shall include filtering crawling as some of the webpage include authentication.